UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T X.744.1

(03/2003)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE X: RÉSEAUX DE DONNÉES ET
COMMUNICATION ENTRE SYSTÈMES OUVERTS

Gestion OSI – Fonctions de gestion et fonctions ODMA

Service RGT de gestion des logiciels en architecture CORBA

Recommandation UIT-T X.744.1

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX DE DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

RÉSEAUX PUBLICS DE DONNÉES	
Services et fonctionnalités	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalisation et commutation	X.50–X.89
Aspects réseau	X.90–X.149
Maintenance	X.150–X.179
Dispositions administratives	X.180–X.199
INTERCONNEXION DES SYSTÈMES OUVERTS	
Modèle et notation	X.200–X.209
Définitions des services	X.210–X.219
Spécifications des protocoles en mode connexion	X.220–X.229
Spécifications des protocoles en mode sans connexion	X.230–X.239
Formulaires PICS	X.240–X.259
Identification des protocoles	X.260–X.269
Protocoles de sécurité	X.270–X.279
Objets gérés des couches	X.280–X.289
Tests de conformité	X.290–X.299
INTERFONCTIONNEMENT DES RÉSEAUX	
Généralités	X.300–X.349
Systèmes de transmission de données par satellite	X.350–X.369
Réseaux à protocole Internet	X.370–X.399
SYSTÈMES DE MESSAGERIE	X.400–X.499
ANNUAIRE	X.500–X.599
RÉSEAUTAGE OSI ET ASPECTS SYSTÈMES	
Réseautage	X.600–X.629
Efficacité	X.630–X.639
Qualité de service	X.640–X.649
Dénomination, adressage et enregistrement	X.650–X.679
Notation de syntaxe abstraite numéro un (ASN.1)	X.680–X.699
GESTION OSI	
Cadre général et architecture de la gestion-systèmes	X.700–X.709
Service et protocole de communication de gestion	X.710–X.719
Structure de l'information de gestion	X.720–X.729
Fonctions de gestion et fonctions ODMA	**X.730–X.799**
SÉCURITÉ	X.800–X.849
APPLICATIONS OSI	
Engagement, concomitance et rétablissement	X.850–X.859
Traitement transactionnel	X.860–X.879
Opérations distantes	X.880–X.899
TRAITEMENT RÉPARTI OUVERT	X.900–X.999

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

Recommandation UIT-T X.744.1

Service RGT de gestion des logiciels en architecture CORBA

Résumé

La présente Recommandation fait partie d'une série de Recommandations qui précisent les prescriptions d'interface à architecture CORBA pour les communications entre un système d'exploitation et un élément de réseau, entre un élément de réseau et un dispositif de médiation, entre un élément de réseau et un adaptateur de Bus Q, et entre systèmes d'exploitation dans un réseau de gestion de télécommunication. La présente Recommandation propose une implémentation de gestion de logiciel fondée sur une architecture CORBA, qui s'appuie sur la Rec. UIT-T X.744. La présente Recommandation a pour objet de définir un modèle CORBA/IDL semblable à celui défini dans la Rec. UIT-T X.744 au moyen de l'élément de service commun de transfert des informations de gestion (CMISE). La présente Recommandation définit les interfaces à granularité fine et grossière (par ex. façades) pour les fonctions de gestion de logiciels. La présente Recommandation est conforme aux normes de modélisation CORBA indiquées dans les Recommandations UIT-T X.780, X.780.1, Q.816, Q.816.1 et M.3120.

Source

La Recommandation X.744.1 de l'UIT-T, élaborée par la Commission d'études 4 (2001-2004) de l'UIT-T, a été approuvée le 29 mars 2003 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.

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

Recommandation UIT-T X.744.1

Service RGT de gestion des logiciels en architecture CORBA

1 Domaine d'application

La présente Recommandation fait partie d'une série de Recommandations qui indiquent les prescriptions d'interface à architecture CORBA pour les communications entre un système d'exploitation et un élément de réseau, entre un système d'exploitation et un dispositif de médiation, entre un système d'exploitation et un adaptateur de bus Q et entre des systèmes d'exploitation à l'intérieur d'un réseau de gestion des télécommunications (RGT) [1].

La fonction de gestion de logiciel comprend la gestion d'un système pour la livraison du logiciel, ainsi que la gestion du logiciel au sein du système.

Il faut distinguer deux aspects du logiciel, décrits comme l'aspect "dormant" et l'aspect "actif" du logiciel.

L'aspect dormant est lié aux données stockées dans un système géré et à la manière dont elles sont livrées et installées. Les données sont en général des informations stockées, telles que des fichiers de données ou des tables, mais il peut s'agir également de fichiers contenant des codes exécutables. Le domaine d'application de la présente Recommandation concerne l'aspect dormant du logiciel.

L'aspect actif du logiciel est lié aux ressources qui l'utilisent. Il n'existe pas de différence réelle entre cet aspect et la vue normale de la gestion des ressources. La présente Recommandation ne concerne pas l'aspect actif du logiciel. Toutefois, les relations entre les objets gérés qui représentent des ressources utilisant le logiciel et les objets gérés représentant le logiciel utilisé, c'est-à-dire l'aspect dormant du logiciel, font partie du domaine d'application de la présente Recommandation.

Le domaine d'application de la présente Recommandation englobe les points suivants:

- initialisation du transfert du logiciel;
- gestion du logiciel après le transfert;
- activation du logiciel, y compris la gestion des versions et des corrections;
- désactivation du logiciel;
- inversion de changements apportés au logiciel;
- validation du logiciel;
- interrogation du logiciel;
- sauvegarde du logiciel;
- restauration du logiciel.

Le domaine d'application de la présente Recommandation ne comprend pas les points suivants:

- mécanisme de transfert du logiciel;
- stockage physique du logiciel (mappage entre le logiciel et le stockage physique du fichier sur disquette, disque dur, etc.);
- formatage du logiciel;
- nomenclature des produits logiciels;
- l'ordonnancement de commandes de gestion de logiciel;
- supervision du logiciel;
- gestion des processus exécutés sur un système.

Dans la présente Recommandation, les interfaces fines et grossières (par exemple, façade) sont définies du point de vue des fonctions de gestion de logiciels. Les interfaces à granularité fine et les interfaces façades fournissent le même soutien de gestion de logiciel. L'une et l'autre peut être utilisées avec des objets gérés à granularité fine et grossière (par exemple ceux définis dans la Rec. UIT-T M.3120 [2]).

Les réseaux de télécommunication actuels sont constitués essentiellement d'un nombre important et de plus en plus élevé de systèmes d'exploitation et d'éléments de réseau provenant de différents fournisseurs. La multiplication et la diversification des réseaux desservis ont suscité des besoins de gestion variés. Cette évolution a entraîné la prolifération d'interfaces de télécommunication spécifiques entre systèmes d'exploitation et éléments de réseau. L'industrie des télécommunications est appelée à bénéficier de la normalisation de ces interfaces, conçues pour assurer l'interopérabilité entre une vaste gamme de systèmes d'exploitation et d'éléments de réseau/adaptateurs de bus Q, au moyen de dispositifs de médiation si nécessaire, et entre systèmes d'exploitation.

La présente Recommandation a principalement pour objet de définir une série de messages d'application et d'objets supports associés pour la prise en charge des télécommunications entre interfaces CORBA. Compte tenu de l'intérêt de proposer des solutions RGT communes, ces messages et ces objets supports sont censés être applicables à d'autres interfaces du RGT ou liés à celui-ci.

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. 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.3120 (2001), *Modèle générique informationnel d'architecture CORBA des réseaux et éléments de réseau*.

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

[4] Recommandation UIT-T Q.816 (2001), *Services RGT à architecture CORBA*, plus Amendement 1 (2001): *Profil des services OMG*.

[5] Recommandation UIT-T Q.816 (2001), *Services RGT à architecture CORBA*, plus Amendement 2 (2002): *Guide de résolution des noms locaux*.

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

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

[8] Recommandation UIT-T Q.822.1 (2001), *Service RGT de gestion de la qualité de fonctionnement en architecture CORBA*.

[9] Recommandation UIT-T X.701 (1997), *Technologies de l'information – Interconnexion des systèmes ouverts – Aperçu général de la gestion-systèmes*.

[10] Recommandation UIT-T X.720 (1992), *Technologies de l'information – Interconnexion des systèmes ouverts – Structure des informations de gestion: modèle d'information de gestion*.

[11] Recommandation UIT-T X.721 (1992), *Technologies de l'information – Interconnexion des systèmes ouverts – Structure des informations de gestion: définition des informations de gestion.*

[12] Recommandation UIT-T X.722 (1992), *Technologies de l'information – Interconnexion des systèmes ouverts – Structure des informations de gestion: directives pour la définition des objets gérés*.

[13] Recommandation UIT-T X.723 (1993), *Technologies de l'information – Interconnexion des systèmes ouverts – Structure des informations de gestion: informations génériques de gestion*.

[14] Recommandation UIT-T X.731 (1992), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de gestion d'états*.

[15] Recommandation UIT-T X.740 (1992), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de piste de vérification de sécurité*.

[16] Recommandation UIT-T X.741 (1995), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: objets et attributs de contrôle d'accès*.

[17] Recommandation UIT-T X.742 (1995), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de comptage d'utilisation aux fins de comptabilité*.

[18] Recommandation UIT-T X.744 (1996), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de gestion de logiciel*.

[19] Recommandation UIT-T X.744 (1996), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de gestion de logiciel*, plus Corrigendum technique 1 (1998).

[20] Recommandation UIT-T X.744 (1996), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de gestion de logiciel*, plus Corrigendum technique 2 (2000).

[21] Recommandation UIT-T X.744 (1996), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de gestion de logiciel*, plus Corrigendum technique 3 (2001).

[22] Recommandation UIT-T X.745 (1993), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion des systèmes: fonction de gestion des tests*.

[23] Recommandation UIT-T X.746 (2000), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de programmation*.

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

[25] Recommandation UIT-T X.780 (2001), *Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA*, plus Corrigendum 1 (2001).

[26] Recommandation UIT-T X.780 (2001), *Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA*, plus Corrigendum 2 (2002).

[27] Recommandation UIT-T X.780 (2001), *Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA*, plus Amendement 1 (2002): *Objets systèmes et guide d'extraction en bloc d'attributs*.

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

[29] Recommandation UIT-T X.780.1 (2001), *Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA à granulation grossière*, plus Amendement 1 (2002): *Façades de système et guide d'utilisation pour l'extraction en bloc d'attributs*.

[30] Recommandation UIT-T X.780.1 (2001), *Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA à granulation grossière*, plus Corrigendum 1 (2002).

3 Définitions

La présente Recommandation utilise les termes suivants employés dans d'autres Recommandations:

3.1 La présente Recommandation utilise les termes suivants définis dans la Rec. UIT-T M.3010 [1]:

a) Modèle d'information de gestion

b) Gestionnaire

3.2 La présente Recommandation utilise le terme suivant défini dans la Rec. UIT-T X.701 [9]:

a) Classe d'objets gérés

3.3 La présente Recommandation utilise les termes suivants définis dans la Rec. UIT-T X.744 [18]:

a) Sauvegarde

b) Livraison

c) Exécution

d) Installation

e) Restauration

f) Désinstallation

g) Utilisation

h) Validation

3.4 La présente Recommandation utilise les termes suivants définis dans la Rec. UIT-T X.780 [24]:

a) Hiérarchie d'héritage

b) Arbre de dénomination

c) Objets subordonnés

d) Objet supérieur

4 Abréviations

La présente Recommandation utilise les abréviations suivantes:

CORBA architecture de courtier commun de requête sur des objets (*common object request broker architecture*)

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

MD dispositif de médiation (*mediation device*)

NE élément de réseau (*network element*)

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

ORB courtier de requêtes sur des objets (*object request broker*)

OS système d'exploitation (*operations system*)

QA adaptateurs de bus Q (*Q adapter*)

RGT réseau de gestion des télécommunications

UID identificateur universel (*universal identifier*)

5 Prescriptions en matière de gestion du logiciel

La gestion du logiciel doit pouvoir répondre aux prescriptions suivantes, compte tenu de commandes ou de conditions pouvant être imposées:

a) capacité de demander une livraison de logiciel à un système de gestion spécifié;

b) capacité de gérer l'installation de logiciel sur un système géré, y compris l'installation de corrections (par exemple, des améliorations) et la possibilité de revenir à une version antérieure du logiciel;

c) capacité de lancer l'exécution d'un programme logiciel;

d) capacité d'interrogation de la valeur des attributs pour tous les logiciels détenus par un système géré;

e) capacité de création et de suppression de logiciel détenu par un système géré;

f) capacité de validation de logiciel détenu par un système géré afin d'en vérifier l'intégrité, ainsi que la capacité de mettre fin à la validation;

g) capacité de restreindre, à des fins administratives, l'utilisation de ressources logicielles d'un système géré;

h) capacité de sauvegarder un item logiciel et de restaurer un item logiciel sauvegardé précédemment.

Le modèle ne fera pas obstacle à l'utilisation de procédures d'archivage, de comptabilité, de vérification et de gestion de licence lors de chaque étape de ce processus.

La réussite ou l'échec de l'opération doit dans tous les cas faire l'objet d'un compte rendu au système gestionnaire.

La manière dont des ressources logicielles utilisent d'autres ressources logicielles est représentée par des relations entre objets logiciels gérés. Ce type de relation, appelé "utilisation", est employé pour indiquer quelles sont les versions de logiciel devant être utilisées à un instant donné. La gestion de cette relation appelle une étude ultérieure.

6 Modèle concernant la fonction de gestion de logiciel

La fonction de gestion de logiciel s'intéresse en premier lieu à la gestion de composants du logiciel. Dans ce contexte, le logiciel comprend des données, de l'information de commande et des instructions exécutables. La vue de gestion du logiciel peut être représentée par un objet géré de la classe "Software Unit" (*unité de logiciel*); la gestion d'un logiciel représentant des instructions exécutables possède des caractéristiques supplémentaires, de sorte qu'une autre classe "Executable Software Unit" (*unité de logiciel exécutable*) est définie comme sous-classe de la classe unité de logiciel.

Un autre aspect de la gestion de logiciel est la gestion de la livraison de logiciel au système géré. Le logiciel n'est pas nécessairement géré et livré dans les mêmes unités. Il est par exemple possible qu'un certain nombre d'unités de logiciel soient livrées ensemble à un système géré (par exemple au moyen d'un CD-ROM). De même, si une unité de logiciel de taille importante doit être livrée au moyen d'un réseau de communication, il peut être nécessaire de la découper en parties plus petites

pour effectuer la livraison. Un autre cas est celui d'une fourniture limitée à la livraison de modifications exigées par un objet logiciel existant, par exemple une correction. La gestion de la livraison est faite au moyen de l'objet géré "Software Distributor" (*distributeur de logiciel*).

6.1 Fonctions de gestion de logiciel

6.1.1 Create (*création*)

Une unité de logiciel peut être créée sur un système géré au moyen de l'opération "Create". Une unité de logiciel peut être créée dans l'état "créé" ou dans l'état "livré" compte tenu du comportement. Une unité de logiciel peut également être créée comme résultat d'une opération de livraison effectuée sur un objet géré "distributeur de logiciel" ou comme résultat d'une action locale. Un objet géré "unité de logiciel" peut émettre une notification de création d'objet à la suite de sa création.

6.1.2 Delete (*suppression*)

Une unité de logiciel peut être supprimée sur un système géré au moyen de l'opération "Delete". L'opération de suppression d'une unité de logiciel a pour effet de bord la suppression des ressources sous-jacentes associées. Un objet géré "unité de logiciel" peut émettre une notification de suppression d'objet à la suite de sa suppression.

6.1.3 Deliver (*livraison*)

Il est possible de demander la livraison d'un ensemble coordonné d'unités de logiciel. Le résultat de la livraison indique un succès ou un échec. La livraison est effectuée en émettant l'action "Deliver" à destination de l'objet géré "distributeur de logiciel". Le résultat d'une action de livraison est la fourniture d'une copie des items de logiciel cible, se trouvant dans l'état "livré", à destination du système cible. Un effet de bord de cette opération peut être la création d'un ou de plusieurs objets associés (par exemple des unités logicielles).

Le conditionnement des unités logicielles et le choix du mécanisme de transfert sont du ressort local et sortent du cadre de la présente Recommandation. Cette information peut, par exemple, être préconfigurée ou spécifiée dans l'action de remise en même temps que les autres informations associées.

6.1.4 Execute Program (*exécution de programme*)

Un système gestionnaire peut demander l'exécution de logiciel exécutable au moyen de l'opération "execute program". La présente Recommandation ne spécifie pas comment une telle exécution du logiciel peut être gérée par la suite, mais se borne à la fourniture d'un mécanisme de lancement de l'exécution du logiciel exécutable.

6.1.5 Get (*recherche*)

L'opération "Get" fournit la possibilité de trouver une information concernant le logiciel, par exemple de déterminer quel logiciel est présent, quel logiciel est disponible, quelles sont les relations entre logiciels, etc. En cas de succès, le résultat de l'opération de recherche fournit l'information demandée.

6.1.6 Install (*installation*)

L'opération d'installation adapte le logiciel pour une utilisation particulière. Cette opération peut nécessiter le cas échéant un volume et un temps de traitement importants. Elle peut entraîner la vérification de l'existence de tous les composants de la version du logiciel cible sur le système géré (comme partie constituante d'une mise à jour ou préexistants sur le système), ainsi que leur assemblage pour les préparer en vue d'une utilisation. L'installation est réalisée par l'envoi de l'action "Install" à destination de l'objet géré "unité de logiciel" dont elle constitue une caractéristique optionnelle.

Une correction est une modification d'un logiciel pouvant être représentée par un objet géré logiciel. En conséquence, une correction peut être livrée, installée, utilisée et copiée au moyen de la gestion du logiciel.

L'installation d'une correction représente un cas spécial, dans la mesure où la correction est livrée et où l'application de la correction consiste dans son installation, et a pour résultat la production d'une version mise à jour du logiciel, prête à l'utilisation. La version mise à jour reçoit à des fins d'identification un numéro de version mis à jour qui est fourni en même temps que la correction. Le résultat d'une installation en indique le succès ou l'échec.

L'installation peut commencer après une livraison complète du logiciel. Il est possible que l'installation puisse nécessiter une coordination qui est toutefois en dehors du domaine d'application de la présente Recommandation. Des exemples de coordination sont:

– l'obligation de terminer la livraison de plusieurs items de logiciel avant qu'un autre item de logiciel puisse être installé;
– la vérification que la version actuelle du logiciel n'est pas utilisée avant l'activation d'une nouvelle version;
– la vérification que des versions particulières de versions d'autres logiciels sont installées ou non;
– la synchronisation de l'installation avec d'autres systèmes ouverts, la possibilité d'installations simultanées mais indépendantes sur plus d'un système ouvert, la possibilité de lier des installations sur plusieurs systèmes ouverts pour constituer une seule unité de travail, etc.

6.1.7 Revert (*désinstallation*)

L'opération "Revert" est utilisée pour faire revenir un objet géré "unité de logiciel" dans un état où il n'est pas installé ou pour effectuer la suppression de l'application d'une ou de plusieurs corrections. L'opération de désinstallation est réalisée par l'envoi d'une action à destination de l'objet géré "unité de logiciel" dont elle constitue une caractéristique optionnelle.

Il peut être nécessaire de gérer une information supplémentaire indiquant de quelle manière un objet géré "unité de logiciel" peut être désinstallé. Ce point est toutefois en dehors du domaine d'application de la présente Recommandation.

6.1.8 Set Administrative State (*positionnement de l'état administratif*)

Une fois installé, c'est-à-dire personnalisé pour une utilisation particulière, le logiciel peut être rendu disponible pour une utilisation par l'envoi aux objets gérés "unité de logiciel" de l'opération "positionnement de l'état administratif sur "déverrouillé". Il est possible de modifier l'état de disponibilité du logiciel en positionnant l'état administratif de l'objet géré "unité de logiciel" sur "fermeture en cours" ou sur "verrouillé".

6.1.9 Validation (*validation*)

L'intégrité d'un logiciel peut être vérifiée au moyen de l'opération "Validation". On peut également valider le fait qu'un logiciel livré précédemment reste dans un état permettant son utilisation. Le résultat de la validation indique si le logiciel a été validé ou non. L'opération de validation est réalisée par l'envoi d'une action à destination de l'objet géré "unité de logiciel" dont elle constitue une caractéristique optionnelle.

6.1.10 Notifications

Toutes les opérations applicables à une unité de logiciel peuvent exiger que des résultats soient envoyés au système gestionnaire en même temps que la confirmation de l'exécution de l'opération. Il peut également être nécessaire de notifier à un autre système gestionnaire la fin de l'exécution d'une de ces opérations lorsque la demande concernant cette opération n'a pas été émise par le système gestionnaire.

6.1.11 Backup (*sauvegarde*)

L'opération "Backup" peut servir à un système gestionnaire pour demander l'exécution de la sauvegarde d'un objet cible. Lorsqu'elle s'applique à un objet géré logiciel, l'opération de sauvegarde a pour effet de faire réaliser une copie des ressources sous-jacentes. Elle n'a pas d'effet direct sur les ressources logicielles originales.

6.1.12 Restore (*restauration*)

L'opération "Restore" peut être utilisée par un système gestionnaire pour demander l'exécution de la restauration d'un objet cible précédemment sauvegardé.

6.2 Relations entre objets gérés

La Figure 1 représente la hiérarchie d'héritage pour les objets gérés gestion de logiciel et la Figure 2 la hiérarchie d'arbre de dénomination relative à la gestion de logiciel. Il est à noter que les objets gérés système peuvent avoir d'autres corrélations de nom par rapport à un objet géré élément géré.

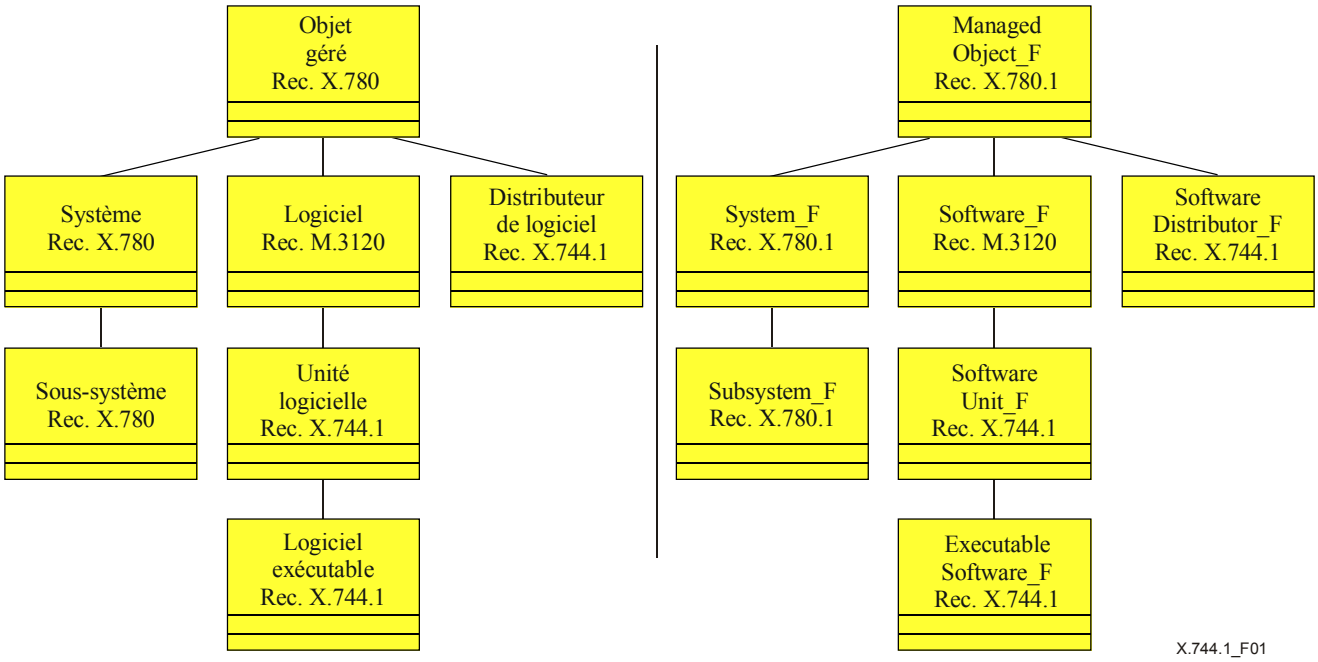

Figure 1/X.744.1 – Hiérarchie d'héritage de gestion de logiciel

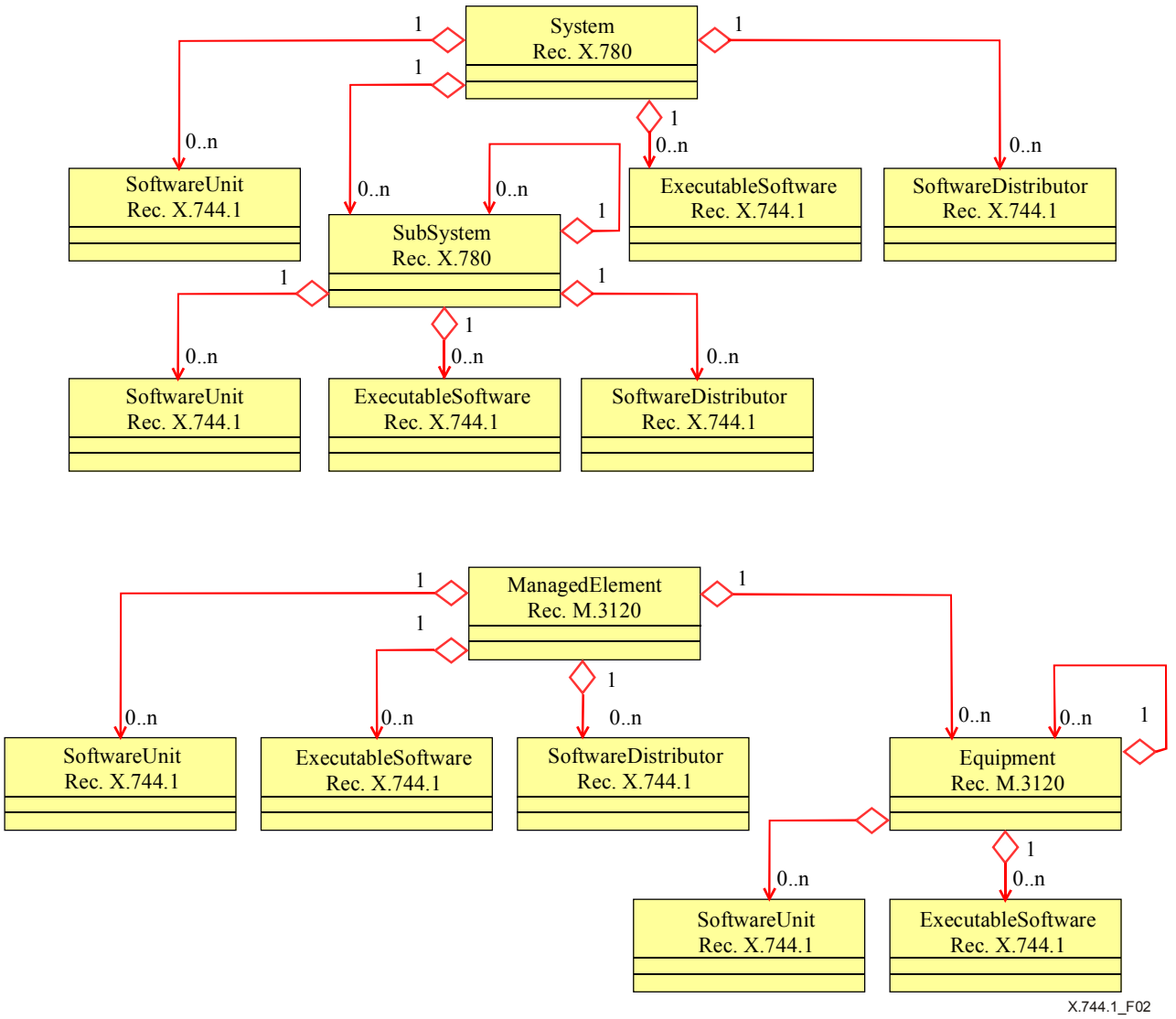

Figure 2/X.744.1 – Hiérarchies d'arbre de dénomination de gestion de logiciel

6.3 Objet géré d'unité de logiciel

6.3.1 Création d'objets gérés "Software Unit" (*unité de logiciel*)

Lorsqu'un objet géré "Software Unit" (*unité de logiciel*) est créé, il représente une ressource qui est disponible pour accepter la livraison d'un composant de logiciel.

Il existe trois mécanismes ou opérations permettant de créer un objet géré unité de logiciel:

1) le logiciel peut être reproduit sur un système géré cible en l'absence de toute opération, par exemple par un moyen local;

2) l'utilisation de l'opération "Create" (*création*) dans l'atelier unité de logiciel permet de créer un objet géré "Software Unit" dans l'état "créé" ou "livré";

3) l'utilisation de l'opération "Deliver" (*livraison*) émise à destination de l'objet géré "Software Distributor" (*distributeur de logiciel*) du système source fournissant la possibilité de créer des objets gérés "unité de logiciel" au sein de la destination cible dans l'état "livré". L'opération de livraison appliquée à un objet géré "distributeur de logiciel" a pour effet la livraison du logiciel à la destination cible. Lorsque l'ensemble du logiciel exigé a été livré avec succès, une notification de résultat de livraison est émise par l'objet géré "distributeur de logiciel" pour indiquer que la livraison a été réalisée.

L'opération de création a pour effet la création d'un objet géré "unité de logiciel" sur la cible destinataire. La réussite de la création sera indiquée par l'existence d'un objet géré "unité de logiciel" sur la destination cible (une notification de création d'objet peut également être émise).

S'il a été créé précédemment mais non livré, l'objet géré "unité de logiciel" peut émettre une notification de changement d'état lorsque la livraison est terminée, ce qui indique que son état est passé de l'état "créé" à "livré". Un objet géré "unité de logiciel" peut également être instancié dans l'état "livré".

Une fois que le logiciel se trouve dans l'état "livré", la prochaine étape pour ce logiciel est de passer dans l'état "installé". L'installation comporte la personnalisation du logiciel à des fins d'utilisation. L'installation peut par exemple impliquer la préparation et l'adaptation de l'unité de logiciel pour une amélioration, éventuellement en réalisant une copie des ressources logicielles en vigueur et en appliquant les modifications à la copie. L'installation peut impliquer la mise en place de relations de configuration et de dépendance entre l'unité de logiciel et d'autres objets logiciels.

Une fois le logiciel installé, il peut être utilisé par d'autres objets gérés. Il peut être nécessaire de rendre le logiciel disponible en positionnant l'état administratif sur "déverrouillé" si ce dernier se trouvait dans l'état "verrouillé". Cette opération est implémentée en appliquant l'opération de positionnement de l'attribut "Administrative State" (*état administratif*).

Le logiciel est rendu indisponible pour une utilisation par verrouillage de l'état administratif, c'est-à-dire en le positionnant sur "verrouillé" ou "fermeture en cours". Ceci interdit à tout nouveau processus l'utilisation du logiciel. Si l'état administratif est positionné sur "verrouillé", tous les processus actifs à l'instant donné sont abandonnés. Si l'état administratif est positionné sur "fermeture en cours", les processus actifs peuvent continuer à s'exécuter, mais le lancement de nouveaux processus n'est plus autorisé. L'état administratif est automatiquement positionné sur verrouillé lorsque tous les processus actifs ont terminé leur exécution. Cette opération est implémentée en appliquant l'opération de remplacement à l'attribut "Administrative State".

L'opération "Revert" (*désinstallation*) peut être utilisée pour inverser l'effet d'une opération d'installation antérieure ou pour supprimer l'application d'une correction.

L'opération "Validate" (*validation*) effectue une vérification de l'intégrité du logiciel, par exemple en exécutant un algorithme de contrôle par sommation ou une détection de virus, etc.

La sauvegarde du logiciel au moyen de l'opération "Backup" est considérée comme la réalisation d'une copie du logiciel en exploitation en vue d'une utilisation en cas de faute de la version en vigueur. Le logiciel sauvegardé peut être restauré au moyen de l'opération "Restore" (*restauration*).

Il est à noter que l'attribut optionnel Applied Patches (*corrections appliqués*) peut devoir être maintenu à l'intérieur du système géré, lorsqu'il n'est pas fourni dans la classe objet géré. Les corrections internes doivent être conservées en cas de prise en charge des services "Install" ou "Revert" (même si l'attribut corrections appliquées ne figure pas dans la classe objet géré).

6.3.2 Etats de l'objet géré "Software Unit"

Un objet géré "unité de logiciel" peut se trouver dans l'un des états du cycle de vie en fonction de la dernière opération qui lui a été appliquée. Les états possibles et leurs significations sont:

– créé (*created*) – La livraison du logiciel n'est pas terminée, toutefois certaines ressources arbitraires du système géré ont été allouées à l'unité de logiciel;

– livré (*delivered*) – Le logiciel a été livré avec succès au système géré;

– installé (*installed*) – Le logiciel a été installé avec succès sur le système géré.

Le Tableau 1 donne le mappage entre les états d'un objet géré "unité de logiciel" et les valeurs d'état et de statut définies dans la Rec. UIT-T X.731 [14].

Tableau 1/X.744.1 – Etats internes de l'unité de logiciel

Etat de l'unité de logiciel	Valeur du statut procédural {Initialization Required}	Valeur du statut de disponibilité {Not Installed}	Etat d'utilisation résultant
Créé	Présent	Présent	Hors service
Livré	Absent	Présent	Hors service
Installé	Absent	Absent	Hors service ou en service

Pour l'objet géré "unité de logiciel", les attributs d'état "Administrative State", "Operational State", "Procedural Status" et "Availability Status" (*respectivement, état administratif, état opérationnel, statut procédural et statut de disponibilité*) sont obligatoires, alors que l'attribut "Usage State" (*état d'utilisation*) est facultatif.

Les états "créé", "livré" et "installé" s'excluent mutuellement, c'est-à-dire qu'une unité de logiciel doit être à un instant donné dans un seul de ces états à la fois. Seules les transitions d'états indiquées au Tableau 2 sont autorisées. Le Tableau 2 se rapporte uniquement aux opérations susceptibles de modifier les états, tandis que les autres opérations n'entraînant pas de changement d'état sont exclues.

Tableau 2/X.744.1 – Matrice de transitions d'état interne d'unité de logiciel

Matrice de transitions d'état		Etat résultant			
		(Inexistant)	Créé	Livré	Installé
Etat initial	(Inexistant)	Changement d'état impossible	Créé ou moyen local[a)]	Créé, moyens locaux ou livré sur distributeur de logiciel[a)]	Changement d'état impossible
	Créé	Supprimé	–	Moyens locaux[a)]	Changement d'état impossible
	Livré	Supprimé	Changement d'état impossible	–	Installé ou moyens locaux [a)]
	Installé	Supprimé	Changement d'état impossible	Désinstallé[a)]	Installé, désinstallé ou toute operation n'entraînant pas un changement d'état (soit: sauvegarde, validation et restauration)[a)]

[a)] Fonction du comportement de l'objet ou des moyens locaux.

Les états de validation et de faute sont indépendants des états "créé", "livré" et "installé".

Les états "validation en cours" et "en faute" sont des états secondaires qui peuvent également exister dans le cycle de vie. L'état "validation en cours" correspond à un statut de disponibilité qui contient la valeur {In Test} (en cours de test). Un objet géré "unité de logiciel" peut entrer dans l'état "validation en cours" ou le quitter indépendamment de la valeur de ses autres états. Une unité de logiciel peut par exemple se trouver dans l'état "validation en cours" pendant une faute (voir Tableau 3).

Tableau 3/X.744.1 – Etat de validation de l'unité de logiciel

Etat de l'unité de logiciel	Valeur du statut de disponibilité {In Test}	Etat opérationnel résultant
Validation en cours (*validating*)	Présent	Hors service ou en service

Une unité de logiciel qui se trouve dans l'état "créé", "installé" ou "validé" peut également se trouver dans l'état "en faute". L'état "en faute" correspond à un statut de disponibilité qui contient la valeur {Failed}. Les causes spécifiques d'entrée ou de sortie de l'état "en faute" sont une question locale. Un objet géré "unité de logiciel" peut entrer dans l'état "en faute" ou le quitter indépendamment de la valeur de ses autres états. Une opération de validation peut par exemple avoir pour effet de bord de faire passer l'unité de logiciel dans l'état "en faute" tout en restant dans l'état "validation en cours", ou une unité logicielle peut quitter l'état "en faute" à la suite d'une opération de désinstallation (voir Tableau 4).

Tableau 4/X.744.1 – Etat de faute de l'unité de logiciel

Etat de l'unité de logiciel	Valeur du statut de disponibilité {Failed}	Etat opérationnel résultant
En faute (*failed*)	Présent	Hors service

6.3.3 Opérations sur l'objet géré "Software Unit"

Un certain nombre d'opérations associées à un objet géré "unité de logiciel" sont utilisées pour en modifier l'état.

Ces opérations sont:

– Backup (*sauvegarde*) – Sauvegarde le logiciel vers la destination cible.

– Create (*création*) – Crée un nouvel objet géré "unité de logiciel" sur le système géré.

– Delete (*suppression*) – Supprime l'objet géré "unité de logiciel" sur le système géré et peut également avoir pour effet la suppression des ressources associées.

– Install (*installation*) – Prépare le logiciel pour une utilisation.

– Restore (*restauration*) – Restaure un logiciel sauvegardé à partir de la destination cible.

– Revert (*désinstallation*) – Inverse l'application d'une correction ou une installation.

– Validate (*validation*) – Vérifie l'intégrité du logiciel.

Les opérations "Create" et "Delete" correspondent aux opérations normalisées Create et Delete définies dans la Rec. UIT-T X.780 [24], les autres correspondent à des méthodes. En outre les valeurs des attributs peuvent être demandées et modifiées en utilisant les opérations "Get" et "Set" définies dans la Rec. UIT-T X.780.

6.3.3.1 Service Backup (*sauvegarde*)

Le service "backup" est utilisé par un système gestionnaire pour demander l'exécution d'une sauvegarde à destination de l'information représentée par l'instance de l'objet cible (c'est-à-dire l'objet géré représentant le logiciel en cours de sauvegarde).

Ce service utilise l'opération Backup dans la classe d'objet géré unité de logiciel. L'opération Backup peut être effectuée indépendamment de l'état de la classe d'objets gérés unité de logiciel. Le paramètre "Destination de sauvegarde" indiquera la destination vers laquelle l'information sera sauvegardée. Des destinations possibles sont les suivantes:

– un objet géré local – L'opération de sauvegarde sera effectuée dans ce cas d'une manière interne au sein du système géré; l'information sera sauvegardée vers l'instance fournie d'objet géré.
– un système distant – L'information de sauvegarde sera transférée dans ce cas en temps différé vers le système distant par un moyen local.

Les exceptions suivantes sont possibles:

– Rec. UIT-T X.780 [24] **Erreur d'application**.
– **NOinformationBackupPackage** – Paquetage Information Backup (*sauvegarde d'informations*) non pris en charge dans cette instance.
– **BackupSoftwareProcessingFailure** – Paramètre Backup Destination (*destination de sauvegarde*) non valide pour l'opération Backup.
– **ConcurrentOperationRequestFailure** – Cette classe d'objets unité de logiciel fait déjà l'objet d'une requête d'opération Backup (*sauvegarde*) ou Restore (*restauration*) (ou d'une autre opération). Les critères conditionnant la prise en charge de requêtes simultanées concernant telle classe d'objets gérés unité de logiciel sont propres au système considéré.

Puisqu'une requête d'opération Backup est susceptible de prendre un certain temps, l'opération Backup renverra immédiatement une valeur. Une notification de rapport d'opération Backup Report sera émise après accomplissement de la sauvegarde d'objet cible.

6.3.3.2 Service Install (*installation*)

Le service "Install" est utilisé par un système gestionnaire pour indiquer à un système géré d'installer une instance d'objet unité de logiciel livrée ou installée. Le cas échéant, le service "Install" mettra à jour la valeur de l'attribut Corrections appliquées.

L'attribut logiciel cible indiquera la source du logiciel (d'origine) du logiciel à installer. L'origine doit être unique, du point de vue des corrections appliquées. Il peut s'agir de l'une des origines suivantes, ou de plusieurs d'entre elles:

– identificateur de correction – Identificateur propre au système.
– Pointeur de correction – Classe d'objets gérés (ou sous-classe) unité de logiciel.

L'opération "Install" renverra la valeur de l'attribut corrections appliquées de l'instance d'objet unité de logiciel à laquelle le service est destiné.

Les exceptions suivantes sont possibles:

– Rec. UIT-T X.780 [24] **Erreur d'application**.
– **NOinstallPackage** – Paquetage Install non pris en charge dans cette instance.
– **InstallSoftwareProcessingFailure** – Paramètre Install Info (*informations d'installation*) non valide pour l'opération Install.
– **OperationStateMismatch** – En cas d'impossibilité d'installer l'unité de logiciel en raison d'un état de non validité. Seuls peuvent être installés les objets gérés unité de logiciel dans l'état livré ou installé. En outre, l'état opérationnel doit être mis à la valeur "en service", l'état administratif à la valeur "déverrouillé" et l'état d'utilisation, à la valeur "actif" ou "repos".
– **ConcurrentOperationRequestFailure** – Cette classe d'objets unité de logiciel fait déjà l'objet d'une requête d'opération Install (*installation*) ou Revert (*désintallation*) (ou d'une autre opération). Les critères conditionnant la prise en charge de requêtes simultanées concernant telle classe d'objets gérés unité de logiciel sont propres au système considéré.

6.3.3.3 Service Restore (*restauration*)

Le service "Restore" est utilisé par un système gestionnaire pour demander l'exécution d'une restauration de l'information représentée par l'instance de l'objet cible. Ce service restaurera une sauvegarde précédente.

Ce service utilise l'opération "Restore" dans l'objet de classe géré unité de logiciel. L'opération de restauration peut être exécutée indépendamment de l'état de la classe d'objets gérés de l'unité de logiciel. Le paramètre Restore Source (source de restauration) doit indiquer la destination à partir de laquelle l'information sera restaurée. Les destinations possibles sont les suivantes:

– un objet géré local – Objet géré local de la même classe que celle à laquelle cette opération est appliquée. L'opération de restauration sera effectuée dans ce cas d'une manière interne au sein du système géré.

– Un système distant. Dans ce cas, l'information de restauration sera transférée en temps différé vers les systèmes distants par un moyen local.

La comportement de l'instance spécifique d'unité de logiciel ou le moyen local détermineront l'état de l'unité de logiciel une fois la restauration effectuée.

Les exceptions suivantes sont possibles:

– Rec. UIT-T X.780 [24] **Erreur d'application**.

– **NOinformationRestorePackage** – Si le paquetage Information Backup (*sauvegarde* d'information) n'est pas pris en charge dans cette instance.

– **RestoreSoftwareProcessingFailure** – Non-validité du paramètre Resource Source pour l'opération Restore (*restauration*).

– **ConcurrentOperationRequestFailure** – Cette classe d'objet unité de logiciel fait déjà l'objet d'une requête d'opération Backup (*sauvegarde*) ou Restore (*restauration*) (ou d'une autre opération). Les critères conditionnant la prise en charge de requêtes simultanées concernant telle classe d'objet géré unité de logiciel sont propres au système considéré.

Puisqu'une requête de restauration est susceptible de prendre un certain temps, l'opération restauration renverra immédiatement une valeur. Une notification de rapport de restauration sera émise suite à l'accomplissement de la restauration d'objets cibles.

6.3.3.4 Service Revert (*désinstallation*)

Le service désinstallation permet à un système gestionnaire (par exemple, à un système d'exploitation) de demander à un système géré de désinstaller une correction appliquée ou une série de corrections du logiciel représenté par l'objet géré unité de logiciel.

Ce service utilise l'opération "Revert" dans la classe d'objets gérés unité de logiciel. Ce service utilise l'opération désinstallation dans la classe d'objets gérés unité de logiciel. Le paramètre "information de désinstallation" doit indiquer une ou plusieurs corrections appliquées précédemment devant faire l'objet d'une désinstallation. Chaque identificateur de correction appliquée est soit un identificateur propre au système, soit une instance d'objet unité de logiciel, en fonction des valeurs fournies initialement dans la précédente ou les précédentes opérations d'installation à l'instance d'objet unité de logiciel.

L'opération "Revert" renverra la valeur de l'attribut corrections appliquées de l'instance d'objet unité de logiciel à laquelle le service est destiné.

Si le service désinstallation réussit à désinstaller toutes les corrections mises en place jusqu'à présent (c'est-à-dire, si l'attribut corrections appliquées est vide), l'état interne unité de logiciel sera modifié et passera de "installé" à "livré".

Les exceptions suivantes sont possibles:

– Rec. UIT-T X.780 [24] **Erreur d'application**.

– **NOrevertPackage** – Si le paquetage Revert (*désinstallation*) n'est pas pris en charge dans cette instance.

– **OperationStateMismatch** – L'unité de logiciel se trouve dans un état de non validité pour l'opération Revert. Pour exécuter l'opération Validate, l'unité de logiciel doit se trouver dans l'état interne "installé"; en outre, l'état administratif doit être positionné sur la valeur "déverrouillé", l'état opérationnel sur la valeur "en service", et l'état d'utilisation sur la valeur "actif" ou "repos".

– **RevertSoftwareProcessingFailure** – Non-validité du paramètre Revert Info pour l'opération Revert (*désintallation*).

– **ConcurrentOperationRequestFailure** – Cette classe d'objets unité de logiciel fait déjà l'objet d'une requête d'opération Install (*installation*) ou Revert (*désinstallation*) (ou d'une autre opération). Les critères conditionnant la prise en charge de requêtes simultanées concernant telle classe d'objets gérés unité de logiciel sont propres au système considéré.

6.3.3.5 Service Validate (*validation*)

Ce service utilise l'opération validation dans la classe d'objets gérés unité de logiciel. "Information de validation" peut indiquer le type de validation à effectuer. Parmi les types de validation possible, figurent:

– validation enregistrée – Dans ce cas, l'information de validation est fournie via un autre objet géré spécifié.

– Validation par défaut – Dans ce cas on utilisera la validation par défaut relative à cette instance d'objet géré spécifique.

Les exceptions suivantes sont possibles:

– Rec. UIT-T X.780 [24]. **Erreur d'application**.

– **NOvalidationPackage** – Si le paquetage Validation (*validation*) n'est pas pris en charge dans cette instance.

– **ValidationSoftwareProcessingFailure** – Non-validité d'un ou plusieurs arguments pour l'opération Validate.

– **OperationStateMismatch** – L'unité de logiciel est dans un état non valide pour l'opération Validate. Pour exécuter l'opération Validate, l'unité de logiciel doit se trouver dans l'état interne "livré" ou "installé".

– **ConcurrentOperationRequestFailure** – Cette classe d'objets unité de logiciel fait déjà l'objet d'une requête d'opération Validate (*validation*) (ou d'une autre opération). Les critères conditionnant la prise en charge de requêtes simultanées concernant telle classe d'objets gérés unité de logiciel sont propres au système considéré.

Du fait qu'une requête de validation peut prendre un certain temps, l'opération de validation enverra immédiatement une valeur. Une notification de rapport de validation sera émise suite à l'accomplissement de la validation d'objets cibles.

6.3.4 Notifications sur l'objet géré unité de logiciel

L'objet géré unité de logiciel comporte les notifications suivantes, notamment celles héritées de l'objet géré logiciel [2], (voir Tableau 5):

Tableau 5/X.744.1 – Notifications d'unité de logiciel

Notification	Référence	Paquetage conditionnel (si conditionnel)
Modification de valeur d'attribut	[24]	"itut_m3120::attributeValueChangeNotificationPackage"
Rapport de sauvegarde	6.3.4.1	"itut_x744d1::informationBackupPackage" et/ou "itut_x744d1::informationAutoBackupPackage"
Création d'objet	[24]	"itut_m3120::createDeleteNotificationsPackage"
Suppression d'objet	[24]	"itut_m3120::createDeleteNotificationsPackage"
Erreur de traitement	[24]	Obligatoire
Rapport de restauration	6.3.4.2	"itut_x744d1::informationRestorePackage" et/ou "itut_x744d1::informationAutoRestorePackage"
Changement d'état	[24]	"itut_m3120::stateChangeNotificationPackage"
Rapport de validation	6.3.4.3	"itut_x744d1::validationPackage"

Les modifications des attributs suivants (une fois définis) entraîneront l'émission des notifications de modification de valeur d'attribut (lorsqu'elles sont prises en charge):

– objets affectés;
– état d'alarme;
– corrections appliquées;
– état de disponibilité;
– liste des problèmes actuels;
– destination de sauvegarde automatique future;
– seuil de déclenchement de sauvegarde automatique future;
– autorisation de restauration automatique future;
– source de restauration automatique future;
– état de procédure;
– étiquette utilisateur;
– version.

Les modifications des états ci-dessous (s'il sont définis) entraîneront l'émission des notifications suivantes de changement d'état (s'il sont pris en charge):

– état opérationnel;
– état administratif;
– état d'utilisation.

6.3.4.1 Notification de rapport de sauvegarde

La notification de rapport de sauvegarde est émise afin de notifier une sauvegarde d'objet géré. La sauvegarde peut avoir été déclenchée automatiquement (conformément aux critères définis dans le seuil de déclenchement de sauvegarde automatique future) et aux attributs de destination de sauvegarde automatique future) par une requête de gestion (via l'opération sauvegarde) ou déclenché par le système géré.

La destination de sauvegarde peut être locale (c'est-à-dire sauvegarde vers un autre objet unité de logiciel, à l'intérieur du système géré local) ou en temps différé vers un système distant au moyen d'un protocole particulier de transfert de fichier (par exemple, FTAM). Le résultat de la sauvegarde sera indiqué dans cette notification.

6.3.4.2 Notification de rapport de restauration

La notification de rapport de restauration est émise afin de signaler une restauration d'objet géré à partir d'une sauvegarde précédente. La restauration peut avoir été déclenchée de façon automatique (selon le type de source de restauration automatique future, les attributs de type d'autorisation de restauration automatique future, et les critères spécifiques du système), au moyen d'une requête de gestion (par l'intermédiaire de l'opération restauration) ou déclenchée par le système géré.

La source de restauration peut être locale (par exemple, restauration à partir d'un autre objet unité de logiciel, à l'intérieur du système géré local) ou en différé à partir d'un système distant, au moyen d'un protocole particulier de transfert de fichier (par exemple, FTAM). Le résultat de la restauration sera indiqué dans cette notification.

6.3.4.3 Notification de rapport de validation

La notification de rapport de validation est émise afin d'indiquer les résultats d'une opération de validation.

6.4 Objet géré "Executable Software " (*logiciel exécutable*)

La classe d'objets gérés "Executable software" est une sous-classe d'objets gérés "Software unit" (*Unité de logiciel*) (voir § 6.3.1) possédant des caractéristiques supplémentaires qui décrivent les fonctionnalités liées à la possibilité d'exécution. Un logiciel exécutable est un logiciel pouvant être exécuté, par exemple à l'aide des moyens locaux ou au moyen de l'opération d'exécution. L'exécution de logiciels exécutables est possible par une commande de gestion, mais peut exiger des moyens locaux.

Bien que cet aspect soit en dehors du domaine d'application de la présente Recommandation, le modèle n'exclut pas la possibilité de sous-classes, définissant une information telle que la possibilité d'utilisateur unique ou multiple, les conditions d'activité ou d'occupation du logiciel, ou la possibilité d'autoriser un nombre maximal d'utilisateurs multiples.

6.4.1 Etats supplémentaires de l'objet géré "Executable Software"

L'état d'utilisation, définit dans la Rec. UIT-T X.731 [14], indique si le logiciel exécutable est en cours d'utilisation. Dans le cas de la classe d'objets gérés "Executable software", l'état d'utilisation est un attribut obligatoire qui s'ajoute aux autres attributs d'objet géré "Software unit" (voir § 6.3.2). les valeurs "repos", "actif", et "occupé" sont autorisées pour l'état d'utilisation de la classe d'objets gérés "logiciel exécutable".

La valeur "repos" de l'état d'utilisation signifie l'absence d'opérations en cours d'exécution. La signification des valeurs "actif" et "occupé" est particulière à l'implémentation et ne relève pas du domaine d'application de la présente Recommandation. Par exemple, la valeur "actif" pourrait signifier que certaines opérations sont en cours d'exécution, tandis que la valeur "occupé" signifie que le programme a atteint sa capacité maximale et toute autre demande d'exécution (en utilisant le service Execute Program ou un autre moyen local) peut être rejetée ou mise en attente en vue d'une exécution ultérieure. Les spécialisations peuvent exiger l'inclusion d'un seuil pour le nombre d'utilisateurs nécessaire afin d'obtenir la capacité maximale.

6.4.2 Opérations supplémentaires pour l'objet géré "Executable Software"

Les opérations suivantes s'appliquent à un objet géré "logiciel exécutable" en plus de celles définies pour l'objet géré "unité de logiciel" (voir § 6.3.3):

– Execute Program – (*Exécution de programme*): cause l'exécution de l'objet géré logiciel.

L'opération d'exécution de programme est utilisée pour lancer l'exécution du programme logiciel représenté par "Executable software". Il faut installer le logiciel exécutable préalablement à son exécution.

L'état d'un logiciel après son exécution sera déterminé par le comportement de l'instance spécifique du logiciel exécutable ou par un moyen local.

6.4.2.1 Service de programme

Le service exécution de programme est utilisé par un système de gestion afin de lancer l'exécution d'un programme représenté par un objet "Executable Software".

Les paramètres informations supplémentaires doit indiquer les paramètres utilisés lors de l'exécution du logiciel.

Suite au lancement réussi d'une opération, la réponse à l'exécution d'un programme contiendra:

– l'identificateur du processus;

– le propriétaire du processus;

– l'instant de lancement initial;

– les paramètres d'information supplémentaires fournis.

Les exceptions suivantes sont possibles:

– Rec. UIT-T X.780 [24] **Erreur d'application**.

– **NOexecuteProgramPackage** – Si le paquetage Execute Program n'est pas pris en charge dans cette instance.

– **OperationStateMismatch** – L'unité de logiciel Executable Software se trouve dans un état de non validité pour l'opération Execute Program. Pour exécuter cette opération, Executable Software doit se trouver dans l'état interne "installé"; en outre, l'état administratif doit être positionné sur la valeur "déverrouillé", l'état opérationnel sur la valeur "en service", et l'état d'utilisation sur la valeur "actif" ou "repos".

– **ExecuteProgramSoftwareProcessingFailure** – Non-validité d'un ou plusieurs arguments pour l'opération Execute Program.

6.5 Objet géré Software Distributor

Un objet géré "Software Distributor" est un objet statique représentant le ou les mécanismes de livraison d'un système géré. Il s'agit d'un objet géré qui fournit du logiciel à un système géré cible lorsqu'il reçoit une opération de livraison d'un système gestionnaire. Les paramètres de l'opération de livraison peuvent être utilisés pour indiquer l'ensemble de logiciels à livrer, la cible destinataire de la livraison et le choix du mécanisme de transfert. Quoique cette classe d'objets puisse servir à initialiser la livraison au moyen de divers mécanismes de transfert, elle ne modélise aucune de ces classes, ceci étant l'objet de spécialisations.

Cet objet géré émet une notification contenant le résultat de la distribution lorsque celle-ci est terminée.

6.5.1 Opérations sur l'objet géré "Software Distributor"

Les opérations suivantes s'appliquent à un objet géré "distributeur de logiciel":

– Create (*création*) – Crée un nouvel objet géré "distributeur de logiciel". Une notification de création d'objet est émise.

– Deliver (*livraison*) – Incite l'objet géré "distributeur de logiciel" à créer (par une méthode qui est en dehors du domaine d'application de la présente Recommandation) le logiciel spécifié sur le système géré cible et, par effet de bord sur ce même système, entraîne la création de toute ressource devant être associée à cet objet géré "unité de logiciel".

– Delete (suppression) – Provoque la suppression de l'objet géré "distributeur de logiciel" sur le système géré. Une notification de suppression d'objet est émise.

6.5.1.1 Service Deliver (*livraison*)

Le service "Deliver" est utilisé par un système gestionnaire pour demander la distribution d'un logiciel ou d'un ensemble de logiciels. L'information sur l'opération livraison identifie le logiciel à distribuer. Une opération Deliver a pour résultat de livrer une copie des éléments logiciels cibles au système cible dans l'état interne "livré".

Le conditionnement du logiciel et le choix du mécanisme de transfert sont du ressort local et ne relèvent pas du domaine d'application de la présente Recommandation. Par exemple cette information peut être préconfigurée ou spécifiée dans l'opération "Deliver" avec toute autre information connexe.

L'exécution avec succès du service a pour résultat de copier le logiciel à distribuer sur le système cible; il peut en résulter la création d'objets "Software Unit" et/ou "Executable Software". Une fois la commande exécutée, une notification du résultat de livraison est émise.

Les paramètres suivants sont utilisés pour l'opération livraison:

– Identificateur de livraison – Ces paramètres facultatifs désignent un identificateur univoque de cette opération livraison.

– Logiciel cible – Désigne l'origine du logiciel à livrer.

– Système cible – Désigne la destination cible optionnelle concernant le logiciel à livrer. Si cette indication ne correspond pas à une indication cible, le système utilise les moyens locaux afin de déterminer la destination cible.

– Information de transfert – Mécanisme de transfert propre à l'application.

– Information supplémentaire – Information supplémentaire propre à l'application.

Les exceptions suivantes peuvent se présenter:

– Rec. UIT-T X.780 [24] **Erreur d'application**.

– **OperationStateMismatch** – L'objet Software Distributor se trouve dans un état de non-validité pour l'opération "Deliver" (*livraison*). Pour exécuter l'opération Deliver, il faut que l'attribut "état administratif" soit positionné sur "déverrouillé", et l'état opérationnel sur "en service".

– **DeliverSoftwareProcessingFailure** – Non-validité d'un ou plusieurs arguments pour l'opération "Deliver".

6.5.2 Notifications concernant l'objet géré "Software Distributor" (*Distributeur de logiciel*)

L'objet géré Distributeur de logiciel comporte les notifications suivantes (voir Tableau 6):

Tableau 6/X.744.1 – Notifications d'objet Software Distributor

Notification	Référence	Paquetage conditionnel (si conditionnel)
Résultat de livraison	6.5.2.1	Obligatoire
Création d'objet	[24]	Obligatoire
Suppression d'objet	[24]	Obligatoire
Changement d'état	[24]	Obligatoire

Les modifications des états suivants entraîneront l'émission de notifications de changement d'état:

– état opérationnel;

– état administratif.

6.5.2.1 Notification de résultats de livraison

L'objet géré émet une notification de résultat de livraison lorsque l'opération livraison est effectuée. Elle contient les résultats finaux de l'opération et peut indiquer un état de conformité ou de non conformité.

6.6 Utilisation d'identificateurs universels

La présente Recommandation utilise des identificateurs universels (UID, *universal identifier*) tels qu'ils sont définis dans la Rec. UIT-T X.780 [24]. Les identificateurs universels autorisent des extensions locales de valeur constante et de valeur de données. Parmi les exemples de possibilité d'utilisation figure l'autorisation d'arguments localement définis pour une opération d'exécution de programmes et l'extension de la liste des types de fichiers "File Type".

Les identificateurs universels peuvent utiliser le type UID, le type extension de gestion et les types d'attributs de types d'ensemble d'informations supplémentaires. La syntaxe IDL correspondant à ces différents types (définie dans la Rec. UIT-T X.780) est la suivante:

```
struct UIDType {
     string moduleName; // module where value is defined
     short value; // constant within the module
};

struct ManagementExtensionType {
     UIDType id; // identifies the type of info
     any info; // type will depend on id
};

typedef sequence <ManagementExtensionType> AdditionalInformationSetType;
```

Pour obtenir une extension locale d'un identificateur universel, il faut définir de nouveaux modules constants UID dans une nouvelle expression IDL. Les modules constants UID contiendront une constante correspond à chaque constante prise en charge (avec un ou deux types UID) ou deux types de données (avec un type d'extension de gestion et un type d'ensemble d'informations supplémentaires). Des exemples de modules constants UID utilisés dans ces modèles figurent aux § 9.19 et 9.20.

Une description complète des identificateurs universels figure dans la Rec. UIT-T X.780.

6.7 Relations

Les relations suivantes ont été identifiées entre objets gérés logiciels ainsi qu'entre objets gérés logiciels et d'autres objets gérés:

– Dependency (*dépendance*) – Cette relation peut être utilisée pour modéliser le fait qu'un objet logiciel géré dépend d'une certaine manière de la présence d'un autre objet logiciel géré. Une telle relation peut être utilisée pour apporter des corrections à un modèle.

– Configuration (*configuration*) – Cette relation peut être utilisée pour modéliser le fait qu'un objet géré "unité de logiciel" peut affecter le comportement d'un autre objet géré "unité de logiciel". Une police de caractères supplémentaire peut par exemple être modélisée par une relation de configuration. Cette relation peut également être utilisée pour apporter des améliorations ou des corrections à un modèle.

– Utilisation (*utilisation*) – Cette relation peut être utilisée pour indiquer quels sont les autres objets gérés qui utilisent des objets gérés logiciels. De tels objets représenteront probablement des processus en activité sur le système géré.

Une faute d'un objet géré peut avoir pour effet de provoquer une faute pour tout objet géré qui dépend de lui. Une telle faute ne provoquera toutefois pas de faute pour les objets qui lui sont liés par une relation de configuration, quoique le comportement de ces objets puisse être modifié. La détection de logiciel fautif peut entraîner comme résultat l'exigence d'installer une nouvelle copie du logiciel ou de livrer et d'installer une version mise à jour.

La spécification de l'une quelconque de ces relations est en dehors du domaine d'application de la présente Recommandation compte tenu du fait que les relations dépendent des applications.

7 Relations avec d'autres fonctions

Les fonctions suivantes sont assurées par les autres fonctions de gestion-systèmes:

– exécution de logiciel, traitée par la Rec. UIT-T Q.822.1 [8];

– piste de vérification de logiciel, traitée par la fonction de piste de vérification de sécurité (voir Rec. UIT-T X.740 [15]);

– prise en charge de la sécurité de logiciel, traitée par la fonction Objets et Attributs de contrôle d'accès (voir Rec. UIT-T X.741 [16]);

– comptabilité de l'utilisation des logiciels, traitée par la fonction de comptage d'utilisation aux fins de comptabilité (voir Rec. UIT-T X.742 [17]);

– essai de logiciel (y compris installation de l'environnement d'essai, exécution test du logiciel, établissement des points de rupture et suspension et reprise de l'environnement d'essai logiciel traitée par la fonction de gestion des tests (voir Rec. UIT-T X.745 [22]);

– programmation des fonctions de logiciel et des opérations, traitée par la fonction de programmation (voir Rec. UIT-T X.746 [23]).

8 Conformité

Le présent paragraphe définit les critères que doivent respecter les autres normes déclarées conformes à la présente Recommandation, ainsi que les fonctions qui doivent être implémentées par les systèmes déclarés conformes à la présente Recommandation.

8.1 Conformité du système

8.1.1 Points de conformité

Le présent paragraphe décrit les points de conformité qui doivent être pris en charge par les systèmes déclarés conformes aux présentes spécifications.

1) Une implémentation revendiquant la conformité aux présentes exigences doit:
   - Prendre en charge:
     - le profil de conformité de base défini dans la Rec. UIT-T Q.816 [3]. Dans ce cas, chaque instance d'objet géré sera un objet CORBA instancié;
     - le profil de conformité de base définit dans la Rec. UIT-T Q.816.1 [7]. Dans ce cas, chaque classe d'objets gérés prise en charge comportera un objet CORBA de façade instancié (voir Recommandations UIT-T Q.816.1 et X.780.1 [28]).
   - Prendre en charge soit:
     - les prescriptions d'objets "Software Unit" (sans prise en charge "Executable Software" ou "Software Distributor");
     - les prescriptions d'objets "Executable Software" (sans prise en charge d'objets "Software Unit" ou "Software Distributor");
     - les prescriptions d'objet "Software Distributor" (sans prise en charge d'objet "Software Unit" ou "Executable Software"). Cette prise en charge peut intervenir avant livraison du logiciel;
     - l'ensemble des prescriptions concernant l'objet "Software Unit" et l'objet "Executable Software" (sans prise en charge de l'objet "Software Distributor");
     - l'ensemble des prescriptions concernant l'objet "Software Unit" et "Software Distributor" (sans prise en charge de l'objet "Executable Software");
     - l'ensemble des prescriptions concernant l'objet "Executable Software" et l'objet "Software Distributor" (sans prise en charge de l'objet "Software Unit");
     - l'ensemble des prescriptions concernant les objets " Software Unit", "Executable Software" et "Software Distributor".
   - Utiliser le code IDL indiqué au paragraphe 9.
2) Une implémentation revendiquant sa conformité aux prescriptions d'objets "Software Unit":
   - prendre en charge l'objet géré "Software Unit" spécifié au § 6.3;
   - prendre en charge la création d'au moins un objet géré de la classe d'objets gérés "Software Unit".
3) Une implémentation revendiquant sa conformité aux prescriptions d'objet "Executable Software" doit:
   - prendre en charge l'objet géré "Executable Software" spécifié au § 6.4.
   - prendre en charge la création d'au moins un objet géré de la classe objets gérés "Executable Software".
4) Une implémentation revendiquant sa conformité aux prescriptions d'objet "Software Distributor" doit:
   - prendre en charge l'objet géré "Software Distributor" spécifié au § 6.5;
   - prendre en charge la création d'au moins un objet géré de la classe objets gérés "Software Distributor".

8.2 Directives de déclaration de conformité

Les utilisateurs du présent cadre général doivent, lors de la rédaction des déclarations de conformité, veiller à ce qui suit. Etant donné que les modules IDL vont être utilisés comme des espaces nominatifs, ils pourront, comme permis par les règles IDL du groupe OMG, être subdivisés entre plusieurs fichiers. Lorsqu'un module sera étendu, son nom ne change pas mais un nouveau fichier IDL sera simplement ajouté. Le simple fait de déclarer le nom d'un module dans une déclaration de conformité ne suffira donc pas pour désigner un ensemble d'interfaces en langage IDL. La déclaration de conformité doit désigner un document et son année de publication afin de garantir que la version correcte de la spécification IDL est bien désignée.

9 Code IDL Rec. UIT-T X.744.1

```
#ifndef _itut_x744_1_idl_
#define _itut_x744_1_idl_

#include <itut_x780.idl>
#include <itut_x780_1.idl>
#include <itut_x780ct.idl>
#include <itut_m3120.idl>

#pragma prefix "itu.int"

/**
Ce code IDL (à partir de la ligne "#ifndef … " jusqu'à la fin de cette section)
doit être mémorisé dans un fichier nommé "itut_x744_1.idl" situé sur le trajet
de recherche utilisé par le compilateur IDL compiler de votre système. Il faut
utiliser un compilateur prenant en charge la version CORBA spécifiée dans la
Rec. UIT-T Q.816.
*/

/**
Ce module, itut_x744d1, contient la définition d'interface IDL pour la
Rec. UIT-T X.744. Les définitions IDL contenues dans ce fichier sont les
interfaces d'objet.
*/

module itut_x744d1
{

/**
```

9.1 Importations

```
*/
     /**
     Types importés de la Rec. UIT-T X.780
     */

     typedef itut_x780::AdditionalInformationSetType AdditionalInformationSetType;
     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::ApplicationErrorInfoType ApplicationErrorInfoType;
     typedef itut_x780::AvailabilityStatusSetType AvailabilityStatusSetType;
     typedef itut_x780::DeletePolicyType DeletePolicyType;
     typedef itut_x780::ExternalTimeType ExternalTimeType;
     typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;
     typedef itut_x780::Istring Istring;
     typedef itut_x780::ManagementExtensionType ManagementExtensionType;
     typedef itut_x780::MONameType MONameType;
     typedef itut_x780::NameBindingType NameBindingType;
     typedef itut_x780::NullType NullType;
```

```
typedef itut_x780::OperationalStateType OperationalStateType;
typedef itut_x780::ProceduralStatusSetType ProceduralStatusSetType;
typedef itut_x780::StringSetType StringSetType;
typedef itut_x780::UIDType UIDType;
typedef itut_x780::UsageStateType UsageStateType;

/**
Types importés de la Rec. UIT-T M.3120
*/

typedef itut_m3120::AlarmStatusType AlarmStatusType;
typedef itut_m3120::AlarmSeverityAssignmentProfileNameType
             AlarmSeverityAssignmentProfileNameType;
typedef itut_m3120::ArcProbableCauseSetType ArcProbableCauseSetType;
typedef itut_m3120::ArcIntervalProfileNameType ArcIntervalProfileNameType;
typedef itut_m3120::ArcTimeType ArcTimeType;

/**
```

9.2 Déclarations aval

```
*/
     /**
     Déclarations aval Interface
     */

     interface ExecutableSoftware;
     interface ExecutableSoftware_F;
     interface ExecutableSoftwareFactory;
     interface SoftwareDistributor;
     interface SoftwareDistributor_F;
     interface SoftwareDistributorFactory;
     interface SoftwareUnit;
     interface SoftwareUnit_F;
     interface SoftwareUnitFactory;

     /**
     Déclarations aval valuetype
     */

     valuetype ExecutableSoftwareValueType;
     valuetype SoftwareDistributorValueType;
     valuetype SoftwareUnitValueType;

/**
```

9.3 Structures et définitions de types

```
*/

     /**
     Patch ::= CHOICE {
          patchId  GraphicString, -- identificateur propre au système --
          patchPointer  ObjectInstance } -- de la classe objets Software Unit –
     AppliedPatches ::= SEQUENCE OF Patch
     */

     enum PatchChoice
     {
          patchIdChoice, // identificateur propre au système
          patchPointerChoice // de la classe objets Software Unit (ou Executable
                             // Software)
     };
```

```
union PatchType switch (PatchChoice)
{
     case patchIdChoice:
          Istring patchId;
     case patchPointerChoice:
          MONameType patchPointer;
};

/**
Type d'attribut Applied Patches (corrections appliquées)
*/

typedef sequence <PatchType> AppliedPatchesSeqType;

/**
BackupDestination ::= CHOICE {
     localObject ObjectInstance,
     inLine NULL, -- en ligne dans la notification, dans additionalInfo --
     offLine GraphicString -- système distant, par ex., FTAM --}

Le choix inLine (en ligne) n'est pas pris en charge dans la Rec. UIT-T
X.744.1
*/

enum BackupDestinationChoice
{
     localObjectChoice,
     offLineChoice // système distant, par ex. FTAM
};

union BackupDestinationType switch (BackupDestinationChoice)
{
     case localObjectChoice:
          MONameType localObject;
     case offLineChoice:
          Istring offLine;
};

/**
CheckSum ::= BIT STRING

Type d'attribut Check Sum. L'algorithme de calcul de check sum est
déterminé localement
*/

typedef long CheckSumType;

/**
Type d'attribut Date Of Creation
*/

typedef GeneralizedTimeType DateOfCreationType;

/**
Date ::= CHOICE {
     time GeneralizedTime ,
     noSuchInformationNULL}
*/

enum DateChoice
{
     timeChoice,
     noSuchInformationChoice
```

```
};

union DateType switch (DateChoice)
{
     case timeChoice:
          GeneralizedTimeType time;
     case noSuchInformationChoice:
          NullType noInformation;
};

/**
Type d'attribut File Location (emplacement de fichier)
*/

typedef sequence <Istring> FileLocationSetType;

/**
FileType ::= INTEGER{
     unstructuredText (0), -- FTAM-1
     unstructuredBinary (1), -- FTAM-3
     blockSpecial (2)}

File Type a été converti en UIDType, initialement d'après le module
FileTypeConst. Cela permet aux applications d'ajouter leurs propres types
de fichiers

Type d'attribut File Type
*/

typedef UIDType FileTypeType;

/**
Type d'attribut Future Auto Backup Trigger Threshold

Noter que l'attribut de type flottant est censé correspondre aux types de
seuils de déclenchement de la Rec. UIT-T Q.822.1, mais il ne s'agit pas
d'un véritable seuil Q.822.1.
*/

typedef float FutureAutoBackupTriggerThresholdType;

/**
Type d'attribut Future Auto Restore Allowed – TRUE signifie qu'ils sont
autorisés
*/

typedef boolean FutureAutoRestoreAllowedType;

/**
AutoRestoreSource ::= CHOICE {
     localObject ObjectInstance,
  remoteSystem GraphicString -- en temps différé à partir du système
                             -- distant
}
*/

enum AutoRestoreSourceChoice
{
     autoRestoreSourceLocalObjectChoice,
     autoRestoreSourceRemoteSystemChoice // en différé à partie du système
                                         // distant
};

union AutoRestoreSourceType switch (AutoRestoreSourceChoice)
```

```
{
     case autoRestoreSourceLocalObjectChoice:
          MONameType localObject;
     case autoRestoreSourceRemoteSystemChoice:
          Istring offLine;
};

/**
Type d'attribut Future Auto Restore Source
*/

typedef AutoRestoreSourceType FutureAutoRestoreSourceType;

/**
Type d'attribut Identity Of Creator
*/

typedef Istring IdentityOfCreatorType;

/**
Type d'attribut Identity Of Last Modifier
*/

typedef Istring IdentityOfLastModifierType;

/**
InformationSize ::=     CHOICE {
     numberOfBits  [0] INTEGER,
     numberOfBytes [1] INTEGER}
*/

enum InformationSizeChoice
{
     numberOfBitsChoice,
     numberOfBytesChoice
};

union InformationSizeType switch (InformationSizeChoice)
{
     case numberOfBitsChoice:
          long bits;
     case numberOfBytesChoice:
          long bytes;
};

/**
LastBackupDestination ::= CHOICE {
     notBackedUp NULL,
     localObject ObjectInstance,
     managingSystem AE-title,
     remoteSystem GraphicString}

Le choix inLine n'est pas pris en charge dans la Rec. UIT-T X.744.1
*/

enum LastBackupDestinationChoice
{
     lastBackupDestinationLocalObjectChoice,
     lastBackupDestinationOffLineChoice,
     lastBackupDestinationNotBackedUpChoice
};

/**
Type d'attribut Last Backup Destination
```

```
*/

union LastBackupDestinationType switch (LastBackupDestinationChoice)
{
     case lastBackupDestinationLocalObjectChoice:
          MONameType localObject;
     case lastBackupDestinationOffLineChoice:
          Istring offLine;
     case lastBackupDestinationNotBackedUpChoice:
          NullType noInformation;
};

/**
LastRestoreSource ::= CHOICE {
     notRestored NULL,
     localObject ObjectInstance,
     managingSystem AE-title,
     remoteSystem GraphicString}

Le choix inline n'est pas pris en charge dans la Rec. UIT-T X.744.1
*/

enum LastRestoreSourceChoice
{
     lastRestoreSourceLocalObjectChoice,
     lastRestoreSourceOffLineChoice,
     lastRestoreSourceNotRestoredChoice
};

/**
Type d'attribut Last Restore Source
*/

union LastRestoreSourceType switch (LastRestoreSourceChoice)
{
     case lastRestoreSourceLocalObjectChoice:
          MONameType localObject;
     case lastRestoreSourceOffLineChoice:
          Istring offLine; // en différé à partir du système distant
     case lastRestoreSourceNotRestoredChoice:
          NullType noInformation;
};

/**
Type d'attribut Note Field
*/

typedef Istring NoteFieldType;

/**
Type d'attribut Date Delivered
*/

typedef DateType DateDeliveredType;

/**
Type d'attribut Date Installed
*/

typedef DateType DateInstalledType;

/**
Type d'attribut Date Of Last Modification
*/
```

```
typedef DateType DateOfLastModificationType;

/**

Type d'attribut File Size
*/

typedef InformationSizeType FileSizeType;

/**
Type d'attribut Future Auto Backup Destination
*/

typedef BackupDestinationType FutureAutoBackupDestinationType;

/**
Type d'attribut Last Backup Time
*/

typedef DateType LastBackupTimeType;

/**
Type d'attribut Last Restore Time
*/

typedef DateType LastRestoreTimeType;

/**
BackupResult ::= CHOICE {
     inLine [0]     CHOICE {
          success   BIT STRING,
          fail-pduSizeLimitation  [3]  NULL,
          fail-securityLicensing  [4]  NULL,
          fail-unknown   [5]  NULL},
     local [1] SEQUENCE {
          destination ObjectInstance, -- dans le système géré --
          success BOOLEAN -- TRUE pour succès --
          },
     offLine [2] SEQUENCE {
          destination GraphicString, -- valeur CHOICE résultant du système
                                     -- distant {
               succès    [6]  NULL,
               échec-securityLicensing [7]  NULL,
               échec-inconnu [8]  NULL}
}}

Le choix inLine n'est pas pris en charge dans la Rec. UIT-T X.744.1
*/

enum BackupResultChoice
{
     backupResultFailureChoice, // un certain type d'erreur s'est produit
                                // pendant la sauvegarde de l'objet
     backupResultLocalChoice,
     backupResultOffLineChoice
};

union BackupResultType switch (BackupResultChoice)
{
     case backupResultFailureChoice:
          ApplicationErrorInfoType error; // d'après la Rec. UIT-T X.780
     case backupResultLocalChoice:
          MONameType localObject; // dans le système géré
```

```
     case backupResultOffLineChoice:
          Istring offLine; // dans le système géré
};

/**
DeliverId ::= CHOICE {
     globalValue   OBJECT IDENTIFIER,
     localValue    INTEGER}
*/

typedef long DeliverIdType;

/**
DeliverIdTypeOpt est de type optionnel. Si le discriminateur est "Vrai"
alors la valeur est "présent", sinon la valeur est NullType.
*/

union DeliverIdTypeOpt switch (boolean)
{
     case TRUE:
          DeliverIdType value;
     case FALSE:
          NullType noInformation;
};

/**
DeliverResult ::= INTEGER {
     effectué (0),
     communicationError (1),
     equipmentError (2),
     qosError (3),
     accessDenied (4),
     notFound (5),
     insufficientSpace (6),
     alreadyDelivered (7),
     inProgress (8),
     unknown (9) }

Deliver Result a été converti en UIDType, initialement en se fondant sur le
module DeliverResultConst module. Cela permet aux applications d'ajouter
leurs propres résultats d'opération Deliver.
*/

typedef UIDType DeliverResultType;

/**
Destination ::= CHOICE {
     unique        AE-title,
     multiple  SET OF AE-title}
-- Noter qu'utiliser la syntaxe AE-title indiquée dans la Rec. UIT-T X.227
-- | ISO/CEI 8650-1 Amendement 1 et non "ANY".
*/

typedef sequence <MONameType> MONameSetType;

enum DestinationChoice
{
     singleChoice,
     multipleChoice
};

union DestinationType switch (DestinationChoice)
{
     case singleChoice:
```

```
          MONameType single;
     case multipleChoice:
          MONameSetType multipleValues;
};

/**
ExecuteProgramReply ::= SEQUENCE {
     processId INTEGER,
     processOwner  Identity,
     startTime GeneralizedTime,
     additionalInfoSET OF ManagementExtension OPTIONAL }
*/

typedef Istring IdentityType;

struct ExecuteProgramReplyType
{
     long processId;
     IdentityType processOwner;
     GeneralizedTimeType startTime;
     AdditionalInformationSetType additionalInfo;
};

typedef AppliedPatchesSeqType InstallReplyType;

/**
Qui a émis la requête?
*/

enum RequestType
{
     automaticRequest,
     managementRequest, // i.e., exécution de la méthode
     managedSystemRequest
};

/**
Quels ont été les résultats de l'opération Restore (restauration)?
*/

enum RestoreResultChoice
{
     restoreResultFailureChoice, // un certain type d'erreur s'est produit
                                 // pendant la restauration de l'objet
     restoreResultLocalChoice,
     restoreResultOffLineChoice
};

union RestoreResultType switch (RestoreResultChoice)
{
     case restoreResultFailureChoice:
          ApplicationErrorInfoType error; // d'après la Rec. UIT-T X.780
     case restoreResultLocalChoice:
          MONameType localObject; // dans le système géré
     case restoreResultOffLineChoice:
          Istring offLine; // dans le système géré
};
```

```
/**
RestoreSource ::= CHOICE {
     localObject ObjectInstance,
     inLine BIT STRING,
     offLine GraphicString
          -- système distant via un autre protocole de transfert, par ex.,
          FTAM --
}

Le choix inline n'est pas pris en charge dans la Rec. UIT-T X.744.1
*/

enum RestoreSourceChoice
{
     restoreSourceLocalObjectChoice,
     restoreSourceOffLineChoice
          // système distant via un autre protocole de transfert, par ex.,
          // FTAM
};

union RestoreSourceType switch (RestoreSourceChoice)
{
     case restoreSourceLocalObjectChoice:
          MONameType localObject;
     case restoreSourceOffLineChoice:
          Istring offLine;
};

/**
RevertInfo ::= SEQUENCE OF CHOICE {
     patchId GraphicString, -- identificateur propre au système --
     patchPointer ObjectInstance } --classe d'objets Executable Software --
*/

enum RevertChoice
{
     revertPatchIdChoice, // identificateur propre au système
     revertPatchPointerChoice // classe objets (ou sous-classe Software
                              // Unit
};

union RevertType switch (RevertChoice)
{
     case revertPatchIdChoice:
          Istring patchId;
     case revertPatchPointerChoice:
          MONameType patchPointer;
};

typedef sequence <RevertType> RevertInfoSetType;

/**
RevertReply ::= SEQUENCE {
     revertedPatches    [0] AppliedPatches,
     additionalInfo     [1] SET OF ManagementExtension OPTIONAL }
*/

typedef AppliedPatchesSeqType RevertReplyType;

/**
DistributedSoftware ::= CHOICE {
     distibutedSoftwareId    GraphicString,
     distributedSoftwarePointer  ObjectInstance }
```

```
*/

enum DistributedSoftwareChoice
{
     distributedSoftwareIdChoice,
     distributedSoftwarePointerChoice
};

union DistributedSoftwareType switch (DistributedSoftwareChoice)
{
     case distributedSoftwareIdChoice:
          Istring patchId; // identificateur propre au système
     case distributedSoftwarePointerChoice:
          MONameType patchPointer; // classe objets (ou sous-classe)
                                   // SoftwareUnit
};

typedef sequence <DistributedSoftwareType> TargetSoftwareSetType;

/**
Il y a lieu de noter que AdditionalInformationSetType est un ensemble
ManagementExtensionType
*/

typedef ManagementExtensionType TransferInfoType;

/**
ValidateInfo ::= CHOICE {
     instanceDefaultValidationType    [0] NULL, -- question locale --
     registeredValidationType         [1] OBJECT IDENTIFIER }
*/

enum ValidateInfoChoice
{
     registeredValidationTypeChoice,
     instanceDefaultValidationTypeChoice // question locale
};

union ValidateInfoType switch (ValidateInfoChoice)
{
     case registeredValidationTypeChoice:
          MONameType instanceDefaultValidationType;
     case instanceDefaultValidationTypeChoice:
          NullType noInformation;
};

/**
ValidateReply ::= CHOICE {
     validationTerminated    [0] NULL,
     passValidation[1] NULL,
     passValidationWithResult     [2] SET OF ManagementExtension,
     failValidation[3] NULL,
     failValidationWithResult     [4] SET OF ManagementExtension }

L'opération Fin de validation (Terminate Validation) n'est pas prise en
charge dans la Rec. UIT-T X.744.1
*/

enum ValidateResultChoice
{
     passValidationWithResultChoice,
     failValidationWithResultChoice,
     passValidationChoice,
     failValidationChoice
```

```
};

union ValidateResultType switch (ValidateResultChoice)
{
     case passValidationWithResultChoice:
          AdditionalInformationSetType passValidationWithResult;
     case failValidationWithResultChoice:
          AdditionalInformationSetType failValidationWithResult;
     case failValidationChoice:
          ApplicationErrorInfoType error; // d'après la Rec. UIT-T X.780
     case passValidationChoice:
          NullType noInformation;
};

/**
BackupReply ::= SEQUENCE {
     reply     [0] CHOICE {
          success   NULL, -- pour sauvegarde locale ou en différé
          inLine    BIT STRING },
     additionalInfo[1] SET OF ManagementExtension OPTIONAL }

Dans la Rec. UIT-T X.744.1, l'opération Sauvegarde (Backup) émet une
notification des résultats au lieu de les renvoyer
*/

/**
TerminateValidationInfo ::= ENUMERATED {
     cancel (0),     -- rejet du résultat de la vérification partielle --
     truncate (1) } -- notification du résultat de la vérification
                     -- partiellement effectuée audit --

TerminateValidationReply ::= CHOICE {
     noOutStandingValidation [0] NULL,
     validationCancelled     [1] NULL,
     resultOfPartialValidation    [2] ValidateReply}

L'opération Fin de Validation (Terminate Validation) n'est pas prise en
charge dans la Rec. UIT-T X.744.1
*/

/**
BackupDestinationTypeOpt est un aiguillage de type optionnel. Si le
discriminateur prend la valeur "Vrai", alors la valeur est "présent", sinon
la valeur est NullType.
*/

union BackupDestinationTypeOpt switch (boolean)
{
     case TRUE:
          BackupDestinationType value;
     case FALSE:
          NullType noInformation;
};

/**
RestoreSourceTypeOpt est un aiguillage de type optionnel. Si le
discriminateur prend la valeur "Vrai", alors la valeur est "présent", sinon
la valeur est NullType.
*/

union RestoreSourceTypeOpt switch (boolean)
{
     case TRUE:
          RestoreSourceType value;
```

```
     case FALSE:
          NullType noInformation;
};

/**
TargetSoftwareSetTypeOpt est un aiguillage de type optionnel. Si le
discriminateur prend la valeur "Vrai", alors la valeur est "présent", sinon
la valeur est NullType.
*/

union TargetSoftwareSetTypeOpt switch (boolean)
{
     case TRUE:
          TargetSoftwareSetType value;
     case FALSE:
          NullType noInformation;
};

/**
RevertInfoSetTypeOpt est un aiguillage de type optionnel. Si le
discriminateur prend la valeur "Vrai", alors la valeur est "présent", sinon
la valeur est NullType.
*/

union RevertInfoSetTypeOpt switch (boolean)
{
     case TRUE:
          RevertInfoSetType value;
     case FALSE:
          NullType noInformation;
};

/**
ValidateInfoTypeOpt est un aiguillage de type optionnel. Si le
discriminateur prend la valeur "Vrai", alors la valeur est "présent", sinon
la valeur est NullType.
*/

union ValidateInfoTypeOpt switch (boolean)
{
     case TRUE:
          ValidateInfoType value;
     case FALSE:
          NullType noInformation;
};

struct ValidateSoftwareProcessingErrorType
{
     ValidateInfoTypeOpt validateInfo;
};

/**
DestinationTypeOpt est un aiguillage de type optionnel. Si le
discriminateur prend la valeur "Vrai", alors la valeur est "présent", sinon
la valeur est NullType.
*/

union DestinationTypeOpt switch (boolean)
{
     case TRUE:
          DestinationType value;
     case FALSE:
          NullType noInformation;
};
```

```
/**
TransferInfoTypeOpt est un aiguillage de type optionnel. Si le
discriminateur prend la valeur "Vrai", alors la valeur est "présent", sinon
la valeur est NullType.
*/

union TransferInfoTypeOpt switch (boolean)
{
     case TRUE:
          TransferInfoType value;
     case FALSE:
          NullType noInformation;
};

struct DeliverSoftwareProcessingErrorType
{
     DeliverIdTypeOpt deliverId;
     TargetSoftwareSetTypeOpt targetSoftware;
     DestinationTypeOpt targetSystem;
     TransferInfoTypeOpt transferInfo;
     AdditionalInformationSetType additionalInfo;
};

const string administrativeStatePackage =
     "itut_x744d1::administrativeStatePackage";
const string appliedPatchPackage = "itut_x744d1::appliedPatchPackage";
const string checkSumPackage = "itut_x744d1::checkSumPackage";
const string createDeleteNotificationsPackage =
     "itut_x744d1::createDeleteNotificationsPackage";
const string executeProgramPackage = "itut_x744d1::executeProgramPackage";
const string fileInformationPackage =
     "itut_x744d1::fileInformationPackage";
const string filePackage = "itut_x744d1::filePackage";
const string informationAutoBackupPackage =
     "itut_x744d1::informationAutoBackupPackage";
const string informationAutoRestorePackage =
     "itut_x744d1::informationAutoRestorePackage";
const string informationBackupPackage =
     "itut_x744d1::informationBackupPackage";
const string informationRestorePackage =
     "itut_x744d1::informationRestorePackage";
const string installPackage = "itut_x744d1::installPackage";
const string noteFieldPackage = "itut_x744d1::noteFieldPackage";
const string revertPackage = "itut_x744d1::revertPackage";
const string stateChangeNotificationPackage =
     "itut_x744d1::stateChangeNotificationPackage";
const string usageStatePackage = "itut_x744d1::usageStatePackage";
const string validationPackage = "itut_x744d1::validationPackage";

/**
```

9.4 Exceptions

```
*/

     exception NOadministrativeStatePackage {};
     exception NOappliedPatchPackage {};
     exception NOcheckSumPackage {};
     exception NOexecuteProgramPackage {};
     exception NOfileInformationPackage {};
     exception NOfilePackage {};
     exception NOinformationAutoBackupPackage {};
     exception NOinformationAutoRestorePackage {};
```

```
    exception NOinformationBackupPackage {};
    exception NOinformationRestorePackage {};
    exception NOinstallPackage {};
    exception NOnoteFieldPackage {};
    exception NOrevertPackage {};
    exception NOusageStatePackage {};
    exception NOvalidationPackage {};

    exception BackupSoftwareProcessingFailure
    {
        BackupDestinationTypeOpt backupDestionation;
    };

    /**
    Utilisé par différentes méthodes pour indiquer que l'échec de cette
    opération est imputable à d'autres opérations en attente ou en cours
    d'exécution.
    */

    exception ConcurrentOperationRequestFailure {};

    exception RestoreSoftwareProcessingFailure
    {
        RestoreSourceTypeOpt attributes;
    };

    exception InstallSoftwareProcessingFailure
    {
        TargetSoftwareSetTypeOpt attributes;
    };

    exception RevertSoftwareProcessingFailure
    {
        RevertInfoSetTypeOpt attributes;
    };

    exception ValidateSoftwareProcessingFailure
    {
        ValidateSoftwareProcessingErrorType attributes;
    };

    exception ExecuteProgramSoftwareProcessingFailure
    {
        AdditionalInformationSetType additionalInfo;
    };

    exception DeliverSoftwareProcessingFailure
    {
        DeliverSoftwareProcessingErrorType attributes;
    };

    /**
    Utilisé par différentes méthodes pour indiquer que l'opération ne peut être
    effectuée suite à une condition d'état non valide pour cette opération
    */

    exception OperationStateMismatch {};

/**
```

9.5 Software Unit

```
*/
     /**
     Type de valeur valuetype utilisé pour extraire tous les attributs
     */

     valuetype SoftwareUnitValueType : truncatable itut_m3120::SoftwareValueType
     {
          public AvailabilityStatusSetType availabilityStatus;
               // GET
          public ProceduralStatusSetType proceduralStatus;
               // GET
          public AppliedPatchesSeqType appliedPatches;
               // GET
               // appliedPatchPackage
          public CheckSumType checkSum;
               // GET
               // checkSumPackage
          public DateOfCreationType dateOfCreation;
               // GET
               // fileInformationPackage
          public IdentityOfCreatorType identityOfCreator;
               // GET
               // fileInformationPackage
          public DateOfLastModificationType dateOfLastModification;
               // GET
               // fileInformationPackage
          public IdentityOfLastModifierType identityOfLastModifier;
               // GET
               // fileInformationPackage
          public DateDeliveredType dateDelivered;
               // GET
               // fileInformationPackage
          public DateInstalledType dateInstalled;
               // GET
               // fileInformationPackage
          public FileLocationSetType fileLocation;
               // GET
               // filePackage
          public FileSizeType fileSize;
               // GET
               // filePackage
          public FileTypeType fileType;
               // GET
               // filePackage
          public FutureAutoBackupTriggerThresholdType
                                   futureAutoBackupTriggerThreshold;
               // GET-REPLACE
               // informationAutoBackupPackage
          public FutureAutoBackupDestinationType futureAutoBackupDestination;
               // GET-REPLACE
               // informationAutoBackupPackage
          public FutureAutoRestoreSourceType futureAutoRestoreSource;
               // GET-REPLACE
               // informationAutoRestorePackage
          public FutureAutoRestoreAllowedType futureAutoRestoreAllowed;
               // GET-REPLACE
               // informationAutoRestorePackage
          public LastBackupTimeType lastBackupTime;
               // GET
               // informationBackupPackage
          public LastBackupDestinationType lastBackupDestination;
               // GET
```

```
            // informationBackupPackage
        public LastRestoreTimeType lastRestoreTime;
            // GET
            // informationRestorePackage
        public LastRestoreSourceType lastRestoreSource;
            // GET
            // informationRestorePackage
        public NoteFieldType noteField;
            // GET-REPLACE
            // noteFieldPackage
        public UsageStateType usageState;
            // GET
            // usageStatePackage

    }; // valuetype SoftwareUnitValueType

    /**
    La classe objet Unité de logiciel (Software Unit) est une classe d'objets
    gérés qui fournit une information pouvant être administrée, associée au
    logiciel (le logiciel peut se trouver sous forme de fichier exécutable
    telle qu'un programme logiciel, ou sous forme de fichier non exécutable
    telle que des données ou une table de références croisées). Le type de
    fichier, son emplacement et sa taille figurent parmi les attributs
    identifiés dans cette classe d'objets. En présence du paquetage
    fileInformationPackage, la valeur initiale de l'attribut dateOfCreation est
    obligatoirement l'instance de création de l'objet géré. S'il faut prendre
    en charge des opérations de sauvegarde, alors la prise en charge des
    opérations de restauration est obligatoire. Les paquetages Information
    Backup et Information Auto Backup existeront uniquement dans les objets
    gérés Unité de logiciel également dotés du paquetage Information Restore ou
    du paquetage Information Auto Restore. Les paquetages Information Restore
    et Information Auto Restore existeront uniquement dans les objets gérés
    Unité de logiciel également dotés du paquetage Information Backup ou du
    paquetage Information Auto Backup.
    En présence du paquetage notification de changement de valeur d'attribut
    (attribute value change notification package) (hérité du logiciel
    hyperclasse), la notification attributeValueChange définie dans la Rec.
    UIT-T X.780 doit être émise lors d'un changement de valeur de l'un des
    attributs suivants:
    - Affected Objects
    - Alarm Status
    - Applied Patches
    - Availability Status
    - Current Problem List
    - Future Auto Backup Destination
    - Future Auto Backup Trigger Threshold
    - Future Auto Restore Allowed
    - Future Auto Restore Source
    - Procedural Status
    - User Label
    - Version

    Etant donné que certains des attributs sumentionnés sont contenus dans des
    paquetages conditionnels, le comportement en matière d'émission d'une
    notification attributeValueChange notification s'applique uniquement en
    présence des paquetages conditionnels correspondants dans l'objet géré.
    */

/**
```

9.6 Interface SoftwareUnit

```
*/
     /**
     Objet géré Software Unit
     */

     interface SoftwareUnit : itut_m3120::Software
     {
          /**
          OperationalState et AdministrativeState de l'objet géré logiciel
          doivent maintenant être indiqués avec l'objet géré logiciel géré.
          Autrement dit l'attribut Packages (paquetages) de l'unité de logiciel
          contiendra toujours la chaîne
          "itut_m3120::administrativeOperationalStatesPackage"
          */

          /**
          La Rec. UIT-T X.731 contient la description de l'état de disponibilité
          (Availability Status). Des indications supplémentaires de comportement
          figurent au § 6.3.2
          */

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

          /**
          La Rec. UIT-T X.731 contient la description de l'état procédural
          (Procedural Status). Des indications supplémentaires de comportement
          figurent au § 6.3.2.
          */

          ProceduralStatusSetType proceduralStatusGet ()
               raises (itut_x780::ApplicationError);

          /**
          Applied Patches identifie les corrections qui ont été appliquées et
          qui restent en vigueur dans l'unité de logiciel représentée par
          l'instance de l'objet unité de logiciel. Les corrections sont des
          mises à jour du logiciel. La valeur de cet attribut est définie en
          lecture seulement et il est mis à jour d'une manière automatique
          lorsqu'une correction est effectuée sur le logiciel. La syntaxe de cet
          attribut est une liste d'identificateurs de correction, un
          identificateur de correction pouvant être un choix entre une instance
          d'objet si la correction est représentée sous la forme d'un objet géré
          "unité de logiciel" et une chaîne de caractères graphiques dans le cas
          contraire.

          Il est à noter que l'attribut optionnel Applied Patches (Corrections
          appliqués) peut devoir être maintenu à l'intérieur du système géré,
          lorsqu'il n'est pas fourni dans la classe objets gérés. Les
          corrections internes doivent être conservées en cas de prise en charge
          des services "Install" ou "Revert" (même si l'attribut corrections
          appliquées ne figure pas dans la classe objets gérés).

          PRESENT SI l'instance prend en charge les corrections de logiciel.
          */

          AppliedPatchesSeqType appliedPatchesGet ()
               raises (itut_x780::ApplicationError,
               NOappliedPatchPackage);

          /**
```

```
L'attribut "CheckSum" identifie la somme de contrôle de l'information
du logiciel représentée par l'instance de l'objet unité de logiciel.
PRESENT SI l'instance prend en charge la validation par somme de
contrôle.
*/

CheckSumType checkSumGet ()
     raises (itut_x780::ApplicationError,
     NOcheckSumPackage);

/**
L'attribut "dateOfCreation" identifie la date de création de l'objet
géré "unité de logiciel". La syntaxe est celle du type GeneralizedTime
temps ASN.1 généralisé. PRESENT SI l'instance prend en charge les
informations de fichier.
*/

DateOfCreationType dateOfCreationGet ()
     raises (itut_x780::ApplicationError,
               NOfileInformationPackage);

/**
L'attribut identityOfCreator identifie l'entité créatrice de l'objet
géré. Il peut s'agir d'une chaîne vide si cette identité est inconnue.
PRESENT SI l'instance prend en charge les informations de fichier.
*/

IdentityOfCreatorType identityOfCreatorGet ()
     raises (itut_x780::ApplicationError,
               NOfileInformationPackage);

/**
L'attribut dateOfLastModification identifie la date de la dernière ou
de la plus récente modification (par exemple du raccordement, du
retour, de la création ou de la désinstallation) de l'information
représentée par le logiciel. Les valeurs valides pour cet attribut
sont un temps généralisé ou "Nul" si l'information n'a pas été
fournie. PRESENT SI l'instance prend en charge les informations de
fichier.
*/

DateOfLastModificationType dateOfLastModificationGet ()
     raises (itut_x780::ApplicationError,
               NOfileInformationPackage);

/**
L'attribut identityOfLastModifier identifie le dernier ou le plus
récent auteur d'une modification de l'information représentée par
l'instance de l'objet unité de logiciel. Il peut s'agir d'une chaîne
vide si cette identité est inconnue. PRESENT SI l'instance prend en
charge les informations de fichier.
*/

IdentityOfLastModifierType identityOfLastModifierGet ()
     raises (itut_x780::ApplicationError,
               NOfileInformationPackage);

/**
L'attribut dateDelivered identifie la date à laquelle l'information
représentée par l'instance de l'objet unité de logiciel a été fournie
au système géré. Les valeurs valides pour cet attribut sont un temps
généralisé ou "Nul" si l'information n'a pas été fournie. PRESENT SI
l'instance prend en charge les informations de fichier.
*/
```

```
DateDeliveredType dateDeliveredGet ()
     raises (itut_x780::ApplicationError,
               NOfileInformationPackage);

/**
L'attribut dateInstalled identifie la date à laquelle l'information
représentée par l'instance de l'objet unité de logiciel a été
installée. Les valeurs valides pour cet attribut sont un temps
généralisé ou "Nul" si l'information n'a pas été fournie. PRESENT SI
l'instance prend en charge les informations de fichier.
*/

DateInstalledType dateInstalledGet ()
     raises (itut_x780::ApplicationError,
               NOfileInformationPackage);

/**
L'attribut fileLocation identifie l'adresse physique ou logique
complète du ou des fichiers de logiciel représentés par l'instance de
l'objet unité de logiciel. Le format d'une adresse dépend de
l'implémentation compte tenu des conventions d'adressage du système
géré considéré, de sorte que la représentation syntactique est une
chaîne de caractères graphiques. La valeur nulle de cet attribut
indique que l'information à laquelle l'objet géré "unité de logiciel"
s'applique n'a pas encore été fournie au système géré. PRESENT SI
l'instance prend en charge la représentation d'un fichier.
*/

FileLocationSetType fileLocationGet ()
     raises (itut_x780::ApplicationError,
               NOfilePackage);

/**
L'attribut fileSize indique la taille de l'objet géré unité de
logiciel. PRESENT SI l'instance prend en charge la représentation d'un
fichier.
*/

FileSizeType fileSizeGet ()
     raises (itut_x780::ApplicationError,
               NOfilePackage);

/**
L'attribut "fileType" indique le type de l'unité de logiciel. Les
types d'unités de logiciel possibles sont un fichier binaire non
structuré (par exemple un fichier exécutable), un fichier texte non
structuré (par exemple un fichier non exécutable), un fichier de blocs
spéciaux, etc. PRESENT SI l'instance prend en charge la représentation
d'un fichier.
*/

FileTypeType fileTypeGet ()
     raises (itut_x780::ApplicationError,
               NOfilePackage);

/**
L'attribut futureAutoBackupTriggerThreshold spécifie le seuil de
déclenchement d'une sauvegarde automatique de l'information
représentée par l'instance de l'objet. Le seuil est défini par le
comptage du nombre de modifications de l'information. Une sauvegarde
automatique sera faite une fois ce seuil atteint. La destination de la
sauvegarde est définie par l'attribut
futureAutomaticBackupDestination. Les sauvegardes de ce type sont
```

```
faites en plus des sauvegardes périodiques planifiées. Une
notification autoBackupReport sera émise par l'objet une fois la
sauvegarde automatique effectuée. PRESENT SI l'instance prend en
charge la sauvegarde automatique.
*/

FutureAutoBackupTriggerThresholdType
              futureAutoBackupTriggerThresholdGet ()
     raises (itut_x780::ApplicationError,
              NOinformationAutoBackupPackage);

void futureAutoBackupTriggerThresholdSet
     (in FutureAutoBackupTriggerThresholdType
              futureAutoBackupTriggerThreshold)
     raises (itut_x780::ApplicationError,
              NOinformationAutoBackupPackage);

/**
L'attribut futureAutoBackupDestination spécifie la destination vers
laquelle sera sauvegardée l'information représentée par l'instance de
l'objet. Le critère de sauvegarde est défini par l'attribut
futureAutoBackupTriggerThreshold de l'instance de l'objet. La
destination peut être une autre instance d'objet de la même classe
d'objets existant sur le même système géré local, un système ouvert
distant accédé au moyen d'un protocole de transfert de fichier tel que
FTAM, ou le système gestionnaire accédé en ligne au moyen de la
notification autoBackupReport. PRESENT SI l'instance prend en charge
la sauvegarde automatique.
*/

FutureAutoBackupDestinationType futureAutoBackupDestinationGet ()
     raises (itut_x780::ApplicationError,
              NOinformationAutoBackupPackage);

void futureAutoBackupDestinationSet
     (in FutureAutoBackupDestinationType futureAutoBackupDestination)
     raises (itut_x780::ApplicationError,
              NOinformationAutoBackupPackage);

/**
L'attribut futureAutoRestoreSource spécifie la source de l'information
utilisée pour restaurer l'information représentée par l'instance de
l'objet géré. La source est un objet géré local ou un système distant.
Les critères de déclenchement de la restauration automatique sont
propres au système. PRESENT SI l'instance prend en charge la
restauration automatique.
*/

FutureAutoRestoreSourceType futureAutoRestoreSourceGet ()
     raises (itut_x780::ApplicationError,
              NOinformationAutoRestorePackage);

void futureAutoRestoreSourceSet
     (in FutureAutoRestoreSourceType futureAutoRestoreSource)
     raises (itut_x780::ApplicationError,
              NOinformationAutoRestorePackage);

/**
L'attribut futureAutoRestoreAllowed spécifie si la restauration de
l'information représentée par cette instance de l'objet géré est
autorisée. La syntaxe de cet attribut est un type BOOLEAN, la valeur
"Vrai" signifiant une autorisation et "Faux" signifiant une
interdiction. Les critères de déclenchement de la restauration
```

```
automatique sont propres au système. PRESENT SI l'instance prend en
charge la restauration automatique.
*/

FutureAutoRestoreAllowedType futureAutoRestoreAllowedGet ()
     raises (itut_x780::ApplicationError,
               NOinformationAutoRestorePackage);

void futureAutoRestoreAllowedSet
     (in FutureAutoRestoreAllowedType futureAutoRestoreAllowed)
     raises (itut_x780::ApplicationError,
               NOinformationAutoRestorePackage);

/**
L'attribut lastBackupTime identifie la date de la dernière sauvegarde
de l'information représentée par l'instance de l'objet géré. Les
valeurs valides pour cet attribut sont un temps généralisé ou "Nul" si
aucune sauvegarde n'a été effectuée. PRESENT SI l'instance prend en
charge l'opération sauvegarde.
*/

LastBackupTimeType lastBackupTimeGet ()
     raises (itut_x780::ApplicationError,
               NOinformationBackupPackage);

/**
L'attribut lastBackupDestination identifie la destination, si elle
existe, vers laquelle l'information représentée par l'instance de
l'objet géré a été sauvegardée. PRESENT SI l'instance prend en charge
l'opération sauvegarde.
*/

LastBackupDestinationType lastBackupDestinationGet ()
     raises (itut_x780::ApplicationError,
               NOinformationBackupPackage);

/**
Le service "backup" est utilisé par un système gestionnaire pour
demander l'exécution d'une sauvegarde à destination de l'information
représentée par l'instance de l'objet cible (c'est-à-dire l'objet géré
représentant le logiciel en cours de sauvegarde). Une fois l'argument
dument validé, l'opération backup renverra immédiatement une valeur.
Une notification de rapport de sauvegarde sera émise suite à
l'exécution de la sauvegarde de l'objet cible.

@param backupDestination    Indique la destination vers laquelle
                            l'information sera sauvegardée. Des
                            destinations possibles sont les suivantes:
                            - un objet géré local. L'opération de
                            sauvegarde sera effectuée dans ce cas
                            d'une manière interne au sein du système
                            géré; l'information sera sauvegardée vers
                            l'instance fournie de l'objet géré;
                            - un choix en différé. L'information de
                            sauvegarde sera transférée dans ce cas en
                            temps différé vers le système distant par
                            un moyen local.
PRESENT SI l'instance prend en charge l'opération sauvegarde.
*/

void backup
     (in BackupDestinationType backupDestionation)
     raises (itut_x780::ApplicationError,
               NOinformationBackupPackage,
```

```
                BackupSoftwareProcessingFailure,
                ConcurrentOperationRequestFailure);

/**
L'attribut lastRestoreTime identifie la date de la dernière
restauration de l'information représentée par l'instance de l'objet
géré. Les valeurs valides pour cet attribut sont un temps généralisé
ou "Nul" si aucune restauration n'a été effectuée. PRESENT SI
l'instance prend en charge l'opération restauration ou restauration
automatique.
*/

LastRestoreTimeType lastRestoreTimeGet ()
     raises (itut_x780::ApplicationError,
                NOinformationRestorePackage,
                NOinformationAutoRestorePackage);

/**
L'attribut LastRestoreSource identifie la source, si elle existe, à
partir de laquelle l'information représentée par l'instance de l'objet
géré a été restaurée. PRESENT SI l'instance prend en charge
l'opération restauration ou restauration automatique.
*/

LastRestoreSourceType lastRestoreSourceGet ()
     raises (itut_x780::ApplicationError,
                NOinformationRestorePackage,
                NOinformationAutoRestorePackage);

/**
Le service "restore" est utilisé par un système gestionnaire pour
demander l'exécution d'une restauration de l'information représentée
par l'instance de l'objet cible. Une fois l'argument dument validé,
l'opération restauration renverra immédiatement une valeur. Une
notification de rapport de restauration sera émise suite à l'exécution
de la restauration de l'objet cible.

@param restoreSource         Indique la source vers laquelle
                             l'information sera restaurée. Des
                             destinations possibles sont les suivantes:
                             - un objet géré local de la même classe
                             que celui auquel l'opération est
                             appliquée. L'opération de restauration
                             sera effectuée dans ce cas d'une manière
                             interne au sein du système géré;
                             - choix en différé. L'information
                             restaurée sera transférée dans ce cas en
                             temps différé à partir du système distant
                             par un moyen local.
PRESENT SI l'instance prend en charge l'opération restauration.
*/

void restore
     (in RestoreSourceType restoreSource)
     raises (itut_x780::ApplicationError,
                NOinformationRestorePackage,
                RestoreSoftwareProcessingFailure,
                ConcurrentOperationRequestFailure);

/**
Le service "install" est utilisé par un système gestionnaire pour
indiquer à un système géré d'installer une instance d'objet unité de
logiciel livrées. Le cas échéant, le service "install" mettra à jour
la valeur de l'attribut Corrections appliquées.
```

```
@param targetSoftware       L'attribut logiciel cible indiquera la
                            source du logiciel (d'origine) du logiciel
                            à installer. L'origine doit être unique,
                            du point de vue des corrections
                            appliquées. Il peut s'agir de l'une des
                            origines suivantes, ou de plusieurs
                            d'entre elles:
                            - Identificateur de correction –
                            Identificateur propre au système.
                            - Pointer de correction – Classe d'objets
                            gérés unité de logiciel (ou logiciel
                            exécutable).
@return                     L'opération "install" renverra la valeur
                            de l'attribut corrections appliquées de
                            l'instance d'objet unité de logiciel à
                            laquelle le service est destiné.
PRESENT SI une instance prend en charge l'opération "install".
*/

InstallReplyType install
     (in TargetSoftwareSetType targetSoftware)
     raises (itut_x780::ApplicationError,
               NOinstallPackage,
               InstallSoftwareProcessingFailure,
               OperationStateMismatch,
               ConcurrentOperationRequestFailure);

/**
L'attribut noteField contient tout type d'information ou des
commentaires associés à l'objet géré, incluant toute instruction
particulière d'installation, toutes valeurs et tous paramètres
initiaux, ou toute information nécessaire pour activer les fonctions
de l'objet géré, etc. PRESENT SI une instance le prend en charge.
*/

NoteFieldType noteFieldGet ()
     raises (itut_x780::ApplicationError,
               NOnoteFieldPackage);

void noteFieldSet
     (in NoteFieldType noteField)
     raises (itut_x780::ApplicationError,
               NOnoteFieldPackage);

/**
Le service de désinstallation permet à un système gestionnaire (par
exemple, à un système d'exploitation) de demander à un système géré de
désinstaller une correction appliquée ou une série de corrections du
logiciel représenté par l'objet géré d'unités de logiciel. Si le
service désinstallation réussit à désinstaller toutes les corrections
mises en place jusqu'à présent (c'est-à-dire, si l'attribut
corrections appliquées est vide), l'état interne unité de logiciel
sera modifié et passera de "installé" à "livré".

@param revertInfo           Le paramètre "information de
                            désinstallation" doit indiquer une ou
                            plusieurs corrections appliquées
                            précédemment devant faire l'objet d'une
                            désinstallation. Chaque identificateur de
                            correction appliquée est soit un
                            identificateur propre au système, soit une
                            instance d'objet unité de logiciel, en
                            fonction des valeurs fournies initialement
```

```
                              à cette instance d'objet unité de logiciel
                              dans une précédente opération
                              d'installation.
@return                       Le service "Revert" renverra la valeur de
                              l'attribut corrections appliquées de
                              l'instance d'objet unité de logiciel à
                              laquelle le service est destiné.
PRESENT SI une instance le prend en charge
*/

RevertReplyType revert
     (in RevertInfoSetType revertInfo)
     raises (itut_x780::ApplicationError,
               NOrevertPackage,
               OperationStateMismatch,
               RevertSoftwareProcessingFailure,
               ConcurrentOperationRequestFailure);

/**
L'état Usage State est décrit dans la Rec. UIT-T X.731. PRESENT SI une
instance le prend en charge.
*/

UsageStateType usageStateGet ()
     raises (itut_x780::ApplicationError,
               NOusageStatePackage);

/**
Le service "validate" est utilisé par un système gestionnaire pour
demander l'exécution d'une validation de l'information représentée par
l'instance d'objet unité de logiciel.

@param validateInfo           Indique les types de validation possibles.
                              Parmi ces derniers figurent:
                              - Validation enregistrée - Dans ce cas,
                              l'information de validation est fournie
                              via un autre objet géré spécifié.
                              - Validation par défaut – Dans ce cas on
                              utilisera la validation par défaut
                              relative à cette instance spécifique
                              d'objet géré.
PRESENT SI une instance le prend en charge
*/

void validate
     (in ValidateInfoType validateInfo)
     raises (itut_x780::ApplicationError,
               NOvalidationPackage,
               OperationStateMismatch,
               ValidateSoftwareProcessingFailure,
               ConcurrentOperationRequestFailure);

/**
Le paramètre "Alarm Effect On Service" (effet de l'alarme sur le
service) défini dans la Rec. UIT-T X.744 a été inclus dans les
paramètres "processingErrorAlarm" (alarme d'erreur de traitement).

La notification d'alarme d'erreur de traitement est maintenant
obligatoire dans la classe d'objets gérés logiciels.
*/

MANDATORY_NOTIFICATION(
     itut_x780::Notifications, processingErrorAlarm)
```

```
        /**
        La notification de rapport de sauvegarde est émise afin de notifier
        une sauvegarde de l'information représentée par cet objet. La
        sauvegarde est déclenchée par l'opération. La destination de
        sauvegarde peut être locale (c'est-à-dire sauvegarde vers un autre
        objet unité de logiciel, à l'intérieur du système géré local) ou en
        temps diifféré vers un système distant au moyen d'un protocole
        particulier de transfert de fichier (par exemple FTAM). Le résultat de
        la sauvegarde, c'est-à-dire succès ou échec, sera indiqué dans cette
        notification. PRESENT SI une instance prend en charge l'opération
        sauvegarde ou sauvegarde automatique.
        */

        CONDITIONAL_NOTIFICATION(
             itut_x744d1::Notifications, backupReport,
             informationBackupPackage)

        CONDITIONAL_NOTIFICATION(
             itut_x744d1::Notifications, backupReport,
             informationAutoBackupPackage)

        /**
        La notification de rapport de restauration est émise afin de signaler
        une restauration d'objets gérés à partir d'une sauvegarde précédente.
        La restauration peut avoir été déclenchée de façon automatique (selon
        le type de source de restauration automatique future et les attributs
        de type d'autorisation de restauration automatique future et enfin les
        critères spécifiques du système), au moyen d'une requête de gestion
        (par l'intermédiaire de l'opération restauration) ou déclenchée par le
        système géré. La source de restauration peut être locale (par exemple,
        restauration à partir d'un autre objet unité de logiciel, à
        l'intérieur du système géré local) ou en différé à partir d'un système
        distant, au moyen d'un protocole particulier de transfert de fichier
        (par exemple, STAM). Le résultat de la restauration sera indiqué dans
        cette notification. PRESENT SI une instance prend en charge
        l'opération restauration ou restauration automatique.
        */

        CONDITIONAL_NOTIFICATION(
             itut_x744d1::Notifications, restoreReport,
             informationRestorePackage)

        CONDITIONAL_NOTIFICATION(
             itut_x744d1::Notifications, restoreReport,
             informationAutoRestorePackage)

        /**
        La notification de rapport de validation est émise afin de signaler la
        validation d'un objet géré. Cette notification en indique les
        résultats de l'opération de validation. PRESENT SI une unstance prend
        en charge l'opération de validation.
        */

        CONDITIONAL_NOTIFICATION(
             itut_x744d1::Notifications, validateReport, validationPackage)

    }; // interface SoftwareUnit

/**
```

9.7 Interface SoftwareUnit_F

```
*/
     /**
     Software Unit Facade managed object – voir Rec. UIT-T X.780.1
     */

     interface SoftwareUnit_F : itut_m3120::Software_F
     {
          /**
       OperationalState et AdministrativeState de l'objet géré logiciel doivent
       maintenant être indiqués avec l'objet géré logiciel. Autrement dit
       l'attribut Packages (paquetages) de l'unité de logiciel contiendra
       toujours la chaîne "itut_m3120::administrativeOperationalStatesPackage"
          */

          /**
       La Rec. UIT-T X.731 contient la description de l'état de disponibilité
       (Availability Status). Des indications supplémentaires de comportement
       figurent au § 6.3.2
          */

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

          /**
       La Rec. UIT-T X.731 contient la description de l'état procédural
       (Procedural Status). Des indications supplémentaires de comportement
       figurent au § 6.3.2.
          */

          ProceduralStatusSetType proceduralStatusGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          /**
       Applied Patches identifie les corrections qui ont été appliquées et qui
       restent en vigueur dans l'unité de logiciel représentée par l'instance de
       l'objet unité de logiciel. Les corrections sont des mises à jour du
       logiciel. La valeur de cet attribut est définie en lecture seulement et
       il est mis à jour d'une manière automatique lorsqu'une correction est
       effectuée sur le logiciel. La syntaxe de cet attribut est une liste
       d'identificateurs de correction, un identificateur de correction pouvant
       être un choix entre une instance d'objet si la correction est représentée
       sous la forme d'un objet géré "unité de logiciel" et une chaîne de
       caractères graphiques dans le cas contraire.

       Il est à noter que l'attribut optionnel Applied Patches (Corrections
       appliquées) peut devoir être maintenu à l'intérieur du système géré,
       lorsqu'il n'est pas fourni dans la classe objets gérés. Les corrections
       internes doivent être conservées en cas de prise en charge des services
       "Install" ou "Revert" (même si l'attribut corrections appliquées ne
       figure pas dans la classe objets gérés).

       PRESENT SI l'instance prend en charge les corrections de logiciel.
          */

          AppliedPatchesSeqType appliedPatchesGet
               (in MONameType name)
               raises (itut_x780::ApplicationError,
                         NOappliedPatchPackage);
```

```
   /**
L'attribut "CheckSum" identifie la somme de contrôle de l'information du
logiciel représentée par l'instance de l'objet unité de logiciel. PRESENT
SI l'instance prend en charge la validation par somme de contrôle.
   */

   CheckSumType checkSumGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
        NOcheckSumPackage);

   /**
L'attribut "dateOfCreation" identifie la date de création de l'objet géré
"unité de logiciel". La syntaxe de cet attribut est celle du type
GeneralizedTime. PRESENT SI l'instance prend en charge les informations
de fichier.
   */

   DateOfCreationType dateOfCreationGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfileInformationPackage);

   /**
L'attribut identityOfCreator identifie l'entité créatrice de l'objet
géré. Il peut s'agir d'une chaîne vide si cette identité est inconnue.
PRESENT SI l'instance prend en charge les informations de fichier.
   */

   IdentityOfCreatorType identityOfCreatorGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfileInformationPackage);

   /**
L'attribut dateOfLastModification identifie la date de la dernière ou de
la plus récente modification (par exemple du raccordement, du retour, de
la création ou de la désinstallation) de l'information représentée par le
logiciel. Les valeurs valides pour cet attribut sont un temps généralisé
ou "Nul" si l'information n'a pas été fournie. PRESENT SI l'instance
prend en charge les informations de fichier.
   */

   DateOfLastModificationType dateOfLastModificationGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfileInformationPackage);

   /**
L'attribut identityOfLastModifier identifie le dernier ou le plus récent
auteur d'une modification de l'information représentée par l'instance de
l'objet unité de logiciel. Il peut s'agir d'une chaîne vide si cette
identité est inconnue.
PRESENT SI l'instance prend en charge les informations de fichier.
   */

   IdentityOfLastModifierType identityOfLastModifierGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfileInformationPackage);
```

```
   /**
L'attribut dateDelivered identifie la date à laquelle l'information
représentée par l'instance de l'objet unité de logiciel a été fournie au
système géré. Les valeurs valides pour cet attribut sont un temps
généralisé ou "Nul" si l'information n'a pas été fournie. PRESENT SI
l'instance prend en charge les informations de fichier.
   */

   DateDeliveredType dateDeliveredGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfileInformationPackage);

   /**
L'attribut dateInstalled identifie la date à laquelle l'information
représentée par l'instance de l'objet unité de logiciel a été installée.
Les valeurs valides pour cet attribut sont un temps généralisé ou "Nul"
si l'information n'a pas été fournie. PRESENT SI l'instance prend en
charge les informations de fichier.
   */

   DateInstalledType dateInstalledGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfileInformationPackage);

   /**
L'attribut fileLocation identifie la ou les adresses physiques ou
logiques complètes de l'objet unité de logiciel. Le format d'une adresse
dépend de l'implémentation compte tenu des conventions d'adressage du
système géré considéré. Un ensemble vide de cet attribut indique que
l'information à laquelle l'objet géré "unité de logiciel" s'applique n'a
pas encore été fournie au système géré. PRESENT SI l'instance prend en
charge la représentation d'un fichier.
   */

   FileLocationSetType fileLocationGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfilePackage);

   /**
L'attribut fileSize indique la taille de l'objet géré unite de logiciel.
PRESENT SI l'instance prend en charge la représentation d'un fichier.
   */

   FileSizeType fileSizeGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfilePackage);

   /**
L'attribut "fileType" indique le type de l'unité de logiciel. Les types
d'unités de logiciel possibles sont un fichier binaire non structuré (par
exemple un fichier exécutable), un fichier texte non structuré (par
exemple un fichier non exécutable), un fichier de blocs spéciaux, etc.
PRESENT SI l'instance prend en charge la représentation d'un fichier.
   */

   FileTypeType fileTypeGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOfilePackage);
```

```
   /**
L'attribut futureAutoBackupTriggerThreshold spécifie le seuil de
déclenchement d'une sauvegarde automatique de l'information représentée
par l'instance de l'objet. Le seuil est défini par le comptage du nombre
de modifications de l'information. Une sauvegarde automatique sera faite
une fois ce seuil atteint. La destination de la sauvegarde est définie
par l'attribut futureAutomaticBackupDestination. Les sauvegardes de ce
type sont faites en plus des sauvegardes périodiques planifiées. Une
notification autoBackupReport sera émise par l'objet une fois la
sauvegarde effectuée. PRESENT SI l'instance prend en charge la sauvegarde
automatique.
   */

   FutureAutoBackupTriggerThresholdType
                FutureAutoBackupTriggerThresholdGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOinformationAutoBackupPackage);

   void futureAutoBackupTriggerThresholdSet
        (in MONameType name,
        in FutureAutoBackupTriggerThresholdType
                futureAutoBackupTriggerThreshold)
        raises (itut_x780::ApplicationError,
                NOinformationAutoBackupPackage);

   /**
L'attribut futureAutoBackupDestination spécifie la destination vers
laquelle sera sauvegardée l'information représentée par l'instance de
l'objet. Le critère de sauvegarde est défini par l'attribut
futureAutoBackupTriggerThreshold de l'instance de l'objet. La destination
peut être une autre instance d'objet de la même classe d'objets existant
sur le même système géré local ou un système ouvert distant (au moyen
d'un protocole de transfert de fichier tel que FTAM). PRESENT SI
l'instance prend en charge la sauvegarde automatique.
   */

   FutureAutoBackupDestinationType futureAutoBackupDestinationGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOinformationAutoBackupPackage);

   void futureAutoBackupDestinationSet
        (in MONameType name,
        in FutureAutoBackupDestinationType futureAutoBackupDestination)
        raises (itut_x780::ApplicationError,
                NOinformationAutoBackupPackage);

   /**
L'attribut futureAutoRestoreSource spécifie la source de l'information
utilisée pour restaurer l'information représentée par l'instance de
l'objet géré. La source est un objet géré local ou un système distant.
Les critères de déclenchement de la restauration automatique sont propres
au système. PRESENT SI l'instance prend en charge la restauration
automatique.
   */

   FutureAutoRestoreSourceType futureAutoRestoreSourceGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                NOinformationAutoRestorePackage);
```

```
   void futureAutoRestoreSourceSet
        (in MONameType name,
        in FutureAutoRestoreSourceType futureAutoRestoreSource)
        raises (itut_x780::ApplicationError,
                  NOinformationAutoRestorePackage);

   /**
L'attribut futureAutoRestoreAllowed spécifie si la restauration de
l'information représentée par cette instance de l'objet géré est
autorisée. La syntaxe de cet attribut est un type Boolean, la valeur
"Vrai" signifiant une autorisation et "Faux" signifiant une interdiction.
Les critères de déclenchement de la restauration automatique sont propres
au système. PRESENT SI l'instance prend en charge la restauration
automatique.
   */

   FutureAutoRestoreAllowedType futureAutoRestoreAllowedGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOinformationAutoRestorePackage);

   void futureAutoRestoreAllowedSet
        (in MONameType name,
        in FutureAutoRestoreAllowedType futureAutoRestoreAllowed)
        raises (itut_x780::ApplicationError,
                  NOinformationAutoRestorePackage);

   /**
L'attribut lastBackupTime identifie la date de la dernière sauvegarde de
l'information représentée par l'instance de l'objet géré. Les valeurs
valides pour cet attribut sont un temps généralisé ou "Nul" si aucune
sauvegarde n'a été effectuée. PRESENT SI l'instance prend en charge
l'opération sauvegarde.
   */

   LastBackupTimeType lastBackupTimeGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOinformationBackupPackage);

   /**
L'attribut lastBackupDestination identifie la destination, si elle
existe, vers laquelle l'information représentée par l'instance de l'objet
géré a été sauvegardée. PRESENT SI l'instance prend en charge l'opération
sauvegarde.
   */

   LastBackupDestinationType lastBackupDestinationGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOinformationBackupPackage);

   /**
Le service "backup" est utilisé par un système gestionnaire pour demander
l'exécution d'une sauvegarde à destination de l'information représentée
par l'instance de l'objet cible (c'est-à-dire l'objet géré représentant
le logiciel en cours de sauvegarde). Une fois l'argument dument validé,
l'opération backup renverra immédiatement une valeur. Une notification de
rapport de sauvegarde sera émise suite à l'exécution de la sauvegarde de
l'objet cible.

   @param name                Nom d'instance d'objet géré unite de
                              logiciel
```

```
   @param backupDestination     Indique la destination vers laquelle
                                l'information sera sauvegardée. Des
                                destinations possibles sont les suivantes:
                                - un objet géré local. L'opération de
                                sauvegarde sera effectuée dans ce cas
                                d'une manière interne au sein du système
                                géré; l'information sera sauvegardée vers
                                l'instance fournie de l'objet géré;
                                - un choix en différé. L'information de
                                sauvegarde sera transférée dans ce cas en
                                temps différé vers le système distant par
                                un moyen local.
   PRESENT SI l'instance prend en charge l'opération sauvegarde.
   */

   void backup
        (in MONameType name,
        in BackupDestinationType backupDestionation)
        raises (itut_x780::ApplicationError,
                  NOinformationBackupPackage,
                  BackupSoftwareProcessingFailure,
                  ConcurrentOperationRequestFailure);

   /**
L'attribut lastRestoreTime identifie la date de la dernière restauration
de l'information représentée par l'instance de l'objet géré. Les valeurs
valides pour cet attribut sont un temps ASN.1 généralisé ou "Nul" si
aucune restauration n'a été effectuée. PRESENT SI l'instance prend en
charge l'opération restauration ou restauration automatique.
   */

   LastRestoreTimeType lastRestoreTimeGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOinformationRestorePackage,
                  NOinformationAutoRestorePackage);

   /**
L'attribut LastRestoreSource identifie la source, si elle existe, à
partir de laquelle l'information représentée par l'instance de l'objet
géré a été restaurée. PRESENT SI l'instance prend en charge l'opération
restauration ou restauration automatique.
   */

   LastRestoreSourceType lastRestoreSourceGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOinformationRestorePackage,
                  NOinformationAutoRestorePackage);

   /**
Le service "restore" est utilisé par un système gestionnaire pour
demander l'exécution d'une restauration de l'information représentée par
l'instance de l'objet cible. Une fois l'argument dument validé,
l'opération restauration renverra immédiatement une valeur. Une
notification de rapport de restauration sera émise suite à l'exécution de
la restauration de l'objet cible.

   @param name                  Nom d'instance d'objet géré unité de
                                logiciel
   @param restoreSource         Indique la source vers laquelle
                                l'information sera restaurée. Des
                                destinations possibles sont les suivantes:
```

```
                                - un objet géré local de la même classe
                                que celui auquel l'opération est
                                appliquée. L'opération "restore" sera
                                effectuée dans ce cas d'une manière
                                interne au sein du système géré;
                                - choix en différé. L'information
                                restaurée sera transférée dans ce cas en
                                temps différé à partir du système distant
                                par un protocole de transfert de fichier
                                choisi localement.
   PRESENT SI l'instance prend en charge l'opération restauration.
   */

   void restore
        (in MONameType name,
        in RestoreSourceType restoreSource)
        raises (itut_x780::ApplicationError,
                  NOinformationRestorePackage,
                  RestoreSoftwareProcessingFailure,
                  ConcurrentOperationRequestFailure);

   /**
Le service "install" est utilisé par un système gestionnaire pour
indiquer à un système géré d'installer une instance d'objet unité de
logiciel livrées. Le cas échéant, le service "install" mettra à jour la
valeur de l'attribut Corrections appliquées.

   @param name                  Nom d'instance d'objet géré unité de
                                logiciel
   @param targetSoftware        L'attribut logiciel cible indiquera la
                                source du logiciel (d'origine) du logiciel
                                à installer. L'origine doit être unique,
                                du point de vue des corrections
                                appliquées. Il peut s'agir de l'une des
                                origines suivantes, ou de plusieurs
                                d'entre elles:
                                - Identificateur de correction –
                                Identificateur propre au système.
                                - Pointer de correction – Classe d'objets
                                gérés unité de logiciel (ou logiciel
                                exécutable).
   @return                      L'opération "install" renverra la valeur
                                de l'attribut corrections appliquées de
                                l'instance d'objet unité de logiciel à
                                laquelle le service est destiné.
   PRESENT SI une instance prend en charge l'opération "install".
   */

   InstallReplyType install
        (in MONameType name,
        in TargetSoftwareSetType targetSoftware)
        raises (itut_x780::ApplicationError,
                  NOinstallPackage,
                  InstallSoftwareProcessingFailure,
                  OperationStateMismatch,
                  ConcurrentOperationRequestFailure);

   /**
L'attribut noteField contient tout type d'information ou des commentaires
associés à l'objet géré, incluant toute instruction particulière
d'installation, toutes valeurs et tous paramètres initiaux, ou toute
information nécessaire pour activer les fonctions de l'objet géré, etc.
PRESENT SI une instance le prend en charge.
   */
```

```
   NoteFieldType noteFieldGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOnoteFieldPackage);

   void noteFieldSet
        (in MONameType name,
        in NoteFieldType noteField)
        raises (itut_x780::ApplicationError,
                  NOnoteFieldPackage);

   /**
Le service "revert" (désinstallation) permet à un système gestionnaire
(par exemple, à un système d'exploitation) de demander à un système géré
de désinstaller une correction appliquée ou une série de corrections du
logiciel représenté par l'objet géré d'unités de logiciel. Si le service
désinstallation réussit à désinstaller toutes les corrections mises en
place jusqu'à présent (c'est-à-dire, si l'attribut corrections appliquées
est vide), l'état interne unité de logiciel sera mùodifié et passera de
"installé" à "livré".

   @param name             Nom d'instance d'objet géré unité de logiciel
   @param revertInfo       Le paramètre "information de désinstallation"
                           doit indiquer une ou plusieurs corrections
                           appliquées précédemment devant faire l'objet
                           d'une désinstallation. Chaque identificateur de
                           correction appliquée est soit un identificateur
                           propre au système, soit une instance d'objet
                           unité de logiciel, en fonction des valeurs
                           fournies initialement à cette instance d'objet
                           unité de logiciel dans une précédente opération
                           d'installation.
   @return                 Le service "Revert" renverra la valeur de
                           l'attribut corrections appliquées de l'instance
                           d'objet unité de logiciel à laquelle le service
                           est destiné.
   PRESENT SI une instance le prend en charge
   */

   RevertReplyType revert
        (in MONameType name,
        in RevertInfoSetType revertInfo)
        raises (itut_x780::ApplicationError,
                  NOrevertPackage,
                  OperationStateMismatch,
                  RevertSoftwareProcessingFailure,
                  ConcurrentOperationRequestFailure);

   /**
L'état Usage State est décrit dans la Rec. UIT-T X.731. PRESENT SI une
instance le prend en charge.
   */

   UsageStateType usageStateGet
        (in MONameType name)
        raises (itut_x780::ApplicationError,
                  NOusageStatePackage);

   /**
Le service "validate" est utilisé par un système gestionnaire pour
demander l'exécution d'une validation de l'information représentée par
l'instance d'objet unité de logiciel.
```

```
   @param name             Nom d'instance d'objet géré unité de logiciel
   @param validateInfo     Indique les types de validation possibles.
                           Parmi ces derniers figurent:
                           - validation enregistrée - Dans ce cas,
                           l'information de validation est fournie via un
                           autre objet géré spécifié;
                           - validation par défaut – Dans ce cas on
                           utilisera la validation par défaut relative à
                           cette instance spécifique d'objets gérés.
   PRESENT SI une instance le prend en charge
   */

   void validate
        (in MONameType name,
        in ValidateInfoType validateInfo)
        raises (itut_x780::ApplicationError,
                  NOvalidationPackage,
                  OperationStateMismatch,
                  ValidateSoftwareProcessingFailure,
                  ConcurrentOperationRequestFailure);

   /**
Le paramètre "Alarm Effect On Service" (effet de l'alarme sur le service)
défini dans la Rec. UIT-T X.744 a été inclus dans les paramètres
"processingErrorAlarm" (alarme d'erreur de traitement).

La notification d'alarme d'erreur de traitement est maintenant
obligatoire dans la classe d'objets gérés logiciels.
   */

   MANDATORY_NOTIFICATION(
        itut_x780::Notifications, processingErrorAlarm)

   /**
La notification de rapport de sauvegarde est émise afin de notifier une
sauvegarde de l'information représentée par cet objet. La sauvegarde est
déclenchée par l'opération. La destination de sauvegarde peut être locale
(c'est-à-dire sauvegarde vers un autre objet unité de logiciel, à
l'intérieur du système géré local) ou en temps diifféré vers un système
distant au moyen d'un protocole particulier de transfert de fichier (par
exemple FTAM). Le résultat de la sauvegarde, c'est-à-dire succès ou
échec, sera indiqué dans cette notification. PRESENT SI une instance
prend en charge l'opération sauvegarde ou sauvegarde automatique.
   */

   CONDITIONAL_NOTIFICATION(
        itut_x744d1::Notifications, backupReport,
        informationBackupPackage)

   CONDITIONAL_NOTIFICATION(
        itut_x744d1::Notifications, backupReport,
        informationAutoBackupPackage)

   /**
La notification de rapport de restauration est émise afin de signaler une
restauration d'objets gérés à partir d'une sauvegarde précédente. La
restauration peut avoir été déclenchée de façon automatique (selon le
type de source de restauration automatique future et les attributs de
type d'autorisation de restauration automatique future et enfin les
critères spécifiques du système), au moyen d'une requête de gestion (par
l'intermédiaire de l'opération restauration) ou déclenchée par le système
géré. La source de restauration peut être locale (par exemple,
restauration à partir d'un autre objet unité de logiciel, à l'intérieur
du système géré local) ou en différé à partir d'un système distant, au
```

```
        moyen d'un protocole particulier de transfert de fichier (par exemple,
        STAM). Le résultat de la restauration sera indiqué dans cette
        notification. PRESENT SI une instance prend en charge l'opération
        restauration ou restauration automatique.
           */

           CONDITIONAL_NOTIFICATION(
                itut_x744d1::Notifications, restoreReport,
                informationRestorePackage)

           CONDITIONAL_NOTIFICATION(
                itut_x744d1::Notifications, restoreReport,
                informationAutoRestorePackage)

           /**
        La notification de rapport de validation est émise afin de signaler la
        validation d'un objet géré. Cette notification en indique les résultats
        de l'opération de validation. PRESENT SI une unstance prend en charge
        l'opération de validation.
           */

           CONDITIONAL_NOTIFICATION(
                itut_x744d1::Notifications, validateReport, validationPackage)

     }; // interface SoftwareUnit_F
```

/**

9.8 Interface SoftwareUnitFactory

```
*/
     /**
     Création et suppression d'unité de logiciel
     */

     interface SoftwareUnitFactory : 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 AdministrativeStateType adminstrativeState,
                         // conditional
                         // itut_m3120::administrativeOperationalStatePackage
                         // GET-REPLACE
               in AlarmSeverityAssignmentProfileNameType profile,
                         // conditional
                         // itut_m3120::alarmSeverityAssignmentPointerPackage
                         // GET-REPLACE
               in Istring userLabel,
                         // conditional
                         // itut_m3120::userLabelPackage
                         // GET-REPLACE
               in Istring vendorName,
                         // conditional
                         // itut_m3120::vendorNamePackage
                         // GET-REPLACE
               in Istring version,
                         // conditional
                         // itut_m3120::versionPackage
                         // GET-REPLACE
```

```
                in ArcProbableCauseSetType arcProbableCauseList,
                        // conditional
                        // itut_m3120::arcPackage
                        // GET-REPLACE, ADD-REMOVE
                in ArcIntervalProfileNameType arcIntervalProfilePointer,
                        // conditional
                        // itut_m3120::arcPackage
                        // GET-REPLACE
                in ArcTimeType arcManagementRequestedInterval,
                        // conditional
                        // itut_m3120::arcPackage
                        // GET-REPLACE
                in AvailabilityStatusSetType availabilityStatus,
                        // GET
                        // SET-BY-CREATE
                in FutureAutoBackupTriggerThresholdType
                        futureAutoBackupTriggerThreshold,
                        // GET-REPLACE
                        // informationAutoBackupPackage
                in FutureAutoBackupDestinationType futureAutoBackupDestination,
                        // GET-REPLACE
                        // informationAutoBackupPackage
                in FutureAutoRestoreSourceType futureAutoRestoreSource,
                        // GET-REPLACE
                        // informationAutoRestorePackage
                in FutureAutoRestoreAllowedType futureAutoRestoreAllowed,
                        // GET-REPLACE
                        // informationAutoRestorePackage
                in NoteFieldType noteField
                        // GET-REPLACE
                        // noteFieldPackage
                )
                raises (itut_x780::ApplicationError,
                        itut_x780::CreateError);

        }; // interface SoftwareUnitFactory

/**
```

9.9 Executable Software (*Logiciel exécutable*)

```
*/
        /**
        L'information valuetype permet d'extraire tous les attributs.
        */

        valuetype ExecutableSoftwareValueType : truncatable SoftwareUnitValueType
        {
        }; // valuetype ExecutableSoftwareValueType

        /**
        La classe objet "Executable Software" est une classe d'objets gérés qui
        fournit une information, pouvant être administrée, associée à un programme
        exécutable au sein du système géré. Le programme exécutable lui-même, qui
        peut être constitué de segments de code associés ou non à des segments de
        données, peut être dans un format non normalisé dépendant de la machine et
        qui est généralement illisible pour le système gestionnaire et pour le
        restant du monde extérieur. Une action, dénommée executeProgram
        (conditionnelle), peut être utilisée pour exécuter le programme représenté
        par l'instance de l'objet logiciel exécutable. L'attribut usageState est
        utilisé pour indiquer s'il existe des exécutions du programme en cours.
        */

/**
```

9.10 Interface ExecutableSoftware

```
*/
     /**
     Objet géré "Executable Software" (logiciel exécutable)
     */

     interface ExecutableSoftware : SoftwareUnit
     {
          /**
          UsageState de l'objet géré unité de logiciel doit maintenant être
          indiqué avec l'objet géré logiciel exécutable. Autrement dit
          l'attribut Packages (paquetages) de logiciel exécutable contiendra
          toujours la chaîne "itut_x744d1::usageStatePackage".
          */

          /**
          Le service "Execute Program" est utilisé par un système gestionnaire
          pour lancer l'exécution d'un programme représenté par un objet
          logiciel exécutable. L'exécution de l'action "Execute Program" exige
          que Executable Software soit dans l'état interne "Installed"
          (Installé), que l'état administratif soit "Unlocked" (déverrouillé),
          que l'état opérationnel soit "Enabled" (en service) et que l'état
          d'utilisation soit "Active" (actif) ou "Idle" (au repos).

          @param additionalInfo       Définit les paramètres utilisés pour
                                      exécuter le logiciel.
          @return                     Une requête aboutie sera confirmée par le
                                      renvoi d'informations comprenant
                                      l'identificateur de processus, le
                                      propriétaire de processus, la date de
                                      lancement initial et les paramètres
                                      d'information additionnelle fournis.
          PRESENT SI une instance le prend en charge
          */

          ExecuteProgramReplyType executeProgram
               (in AdditionalInformationSetType additionalInfo)
               raises (itut_x780::ApplicationError,
                         NOexecuteProgramPackage,
                         OperationStateMismatch,
                         ExecuteProgramSoftwareProcessingFailure);

     }; // interface ExecutableSoftware

/**
```

9.11 Interface ExecutableSoftware_F

```
*/
     /**
     Objet géré Executable Unit Facade – voir Rec. UIT-T X.780.1
     */

     interface ExecutableSoftware_F : SoftwareUnit_F
     {
          /**
          UsageState de l'objet géré unité de logiciel doit maintenant être
          indiqué avec l'objet géré logiciel exécutable. Autrement dit
          l'attribut Packages (paquetages) de logiciel exécutable contiendra
          toujours la chaîne "itut_x744d1::usageStatePackage".
          */
```

```
        /**
        Le service "Execute Program" est utilisé par un système gestionnaire
        pour lancer l'exécution d'un programme représenté par un objet
        logiciel exécutable. L'exécution de l'action "Execute Program" exige
        que Executable Software soit dans l'état interne "Installed"
        (Installé), que l'état administratif soit "Unlocked" (déverrouillé),
        que l'état opérationnel soit "Enabled" (en service) et que l'état
        d'utilisation soit "Active" (actif) ou "Idle" (au repos).

        @param name             nom de l'instance objet géré Executable
                                Software
        @param additionalInfo   Définit les paramètres utilisés pour exécuter
                                le logiciel.
        @return                 Une requête aboutie sera confirmée par le
                                renvoi d'informations comprenant
                                l'identificateur de processus, le propriétaire
                                de processus, la date de lancement initial et
                                les paramètres d'information additionnelle
                                fournis.
        PRESENT SI une instance le prend en charge
        */

        ExecuteProgramReplyType executeProgram
             (in MONameType name,
             in AdditionalInformationSetType additionalInfo)
             raises (itut_x780::ApplicationError,
                       NOexecuteProgramPackage,
                       OperationStateMismatch,
                       ExecuteProgramSoftwareProcessingFailure);

    }; // interface ExecutableSoftware_F

/**
```

9.12 ExecutableSoftwareFactory Interface

```
*/
    /**
    Création et suppression d'unité Executable
    */

    interface ExecutableSoftwareFactory : 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 AdministrativeStateType adminstrativeState,
                       // conditional
                       // itut_m3120::administrativeOperationalStatePackage
                       // GET-REPLACE
             in AlarmSeverityAssignmentProfileNameType profile,
                       // conditional
                       // itut_m3120::alarmSeverityAssignmentPointerPackage
                       // GET-REPLACE
             in Istring userLabel,
                       // conditional
                       // itut_m3120::userLabelPackage
                       // GET-REPLACE
             in Istring vendorName,
                       // conditional
                       // itut_m3120::vendorNamePackage
```

```
                    // GET-REPLACE
               in Istring version,
                    // conditional
                    // itut_m3120::versionPackage
                    // GET-REPLACE
               in ArcProbableCauseSetType arcProbableCauseList,
                    // conditional
                    // itut_m3120::arcPackage
                    // GET-REPLACE, ADD-REMOVE
               in ArcIntervalProfileNameType arcIntervalProfilePointer,
                    // conditional
                    // itut_m3120::arcPackage
                    // GET-REPLACE
               in ArcTimeType arcManagementRequestedInterval,
                    // conditional
                    // itut_m3120::arcPackage
                    // GET-REPLACE
               in AvailabilityStatusSetType availabilityStatus,
                    // GET
                    // SET-BY-CREATE
               in FutureAutoBackupTriggerThresholdType
                    futureAutoBackupTriggerThreshold,
                    // GET-REPLACE
                    // informationAutoBackupPackage
               in FutureAutoBackupDestinationType futureAutoBackupDestination,
                    // GET-REPLACE
                    // informationAutoBackupPackage
               in FutureAutoRestoreSourceType futureAutoRestoreSource,
                    // GET-REPLACE
                    // informationAutoRestorePackage
               in FutureAutoRestoreAllowedType futureAutoRestoreAllowed,
                    // GET-REPLACE
                    // informationAutoRestorePackage
               in NoteFieldType noteField
                    // GET-REPLACE
                    // noteFieldPackage
               )
               raises (itut_x780::ApplicationError,
                    itut_x780::CreateError);

     }; // interface ExecutableSoftwareFactory

/**
```

9.13 Software Distributor (*Distributeur de logiciel*)

```
*/
     /**
     L'information valuetype est utilisée afin d'extraire les valeurs de tous
     les attributs.
     */

     valuetype SoftwareDistributorValueType :
               truncatable itut_x780::ManagedObjectValueType
     {
          public AdministrativeStateType administrativeState;
               // GET-REPLACE
          public OperationalStateType operationalState;
               // GET

     }; // valuetype SoftwareDistributorValueType

     /**
```

```
     Un objet géré "Software Distributor" est un objet statique représentant le
     ou les mécanismes de livraison d'un système géré. Il s'agit d'un objet géré
     qui fournit du logiciel à un système géré cible lorsqu'il reçoit une
     opération de livraison d'un système gestionnaire. Cet objet émet une
     notification contenant le résultat de la distribution lorsque celle-ci est
     terminée. La notification stateChangeNotification définie dans la Rec.
     UIT-T X.780 sera émise en cas de changement de la valeur de l'état
     administratif ou de l'état opérationnel.
     */

/**
```

9.14 Interface SoftwareDistributor

```
*/
     /**
     Objet géré "Software Distributor"
     */

     interface SoftwareDistributor : itut_x780::ManagedObject
     {
          /**
       La Rec. UIT-T X.731 contient la description de Administrative State (état
       administratif).
          */

          AdministrativeStateType administrativeStateGet ()
               raises (itut_x780::ApplicationError);

          void administrativeStateSet
               (in AdministrativeStateType administrativeState)
               raises (itut_x780::ApplicationError);

          /**
       La Rec. UIT-T X.731 contient la description de Operational State (état
       opérationnel).
          */

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

          /**
       Le service "deliver" est utilisé par un système gestionnaire pour
       demander la distribution d'un logiciel ou d'un ensemble de logiciels.
       L'information de livraison identifie le logiciel à distribuer. Le
       résultat d'une action de livraison est la fourniture d'une copie des
       items de logiciel cible, se trouvant dans l'état "livré", à destination
       du système cible.

       Le conditionnement des unités logicielles et le choix du mécanisme de
       transfert sont du ressort local et sortent du cadre de la présente
       Recommandation. Cette information peut, par exemple, être préconfigurée
       ou spécifiée dans l'action de remise en même temps que les autres
       informations associées.

       L'exécution avec succès du service a pour résultat de copier le logiciel
       à distribuer sur le système cible; il peut en résulter la création
       d'objets "Software Unit" et/ou "Executable Software". Une fois la
       commande exécutée, une notification du résultat de livraison est émise.

          @param deliverId            Ce paramètre facultatif désigne un
                                      identificateur univoque de cette opération
                                      livraison.
          @param targetSoftware       Désigne l'origine du logiciel à livrer.
```

```
        @param targetSystem         Désigne la destination cible optionnelle
                                    concernant le logiciel à livrer. Si cette
                                    indication ne correspond pas à une
                                    indication cible, le système utilise les
                                    moyens locaux afin de déterminer la
                                    destination cible.
        @param transferInfo         Mécanisme de transfert propre à
                                    l'application.
        @param additionalInfo       Information supplémentaire propre à
                                    l'application.
        */

        void deliver
             (in DeliverIdTypeOpt deliverId,
             in TargetSoftwareSetType targetSoftware,
             in DestinationType targetSystem,
             in TransferInfoType transferInfo,
             in AdditionalInformationSetType additionalInfo)
             raises (itut_x780::ApplicationError,
                       OperationStateMismatch,
                       DeliverSoftwareProcessingFailure);

        /**
      L'objet géré émet une notification de résultat de livraison lorsque
      l'opération livraison est effectuée. Elle contient les résultats finaux
      de l'opération et peut indiquer un état de conformité ou de
      non-conformité.
        */

        MANDATORY_NOTIFICATION(
             itut_x744d1::Notifications, deliverResultNotification)
        MANDATORY_NOTIFICATION(
             itut_x780::Notifications, objectCreation)
        MANDATORY_NOTIFICATION(
             itut_x780::Notifications, objectDeletion)
        MANDATORY_NOTIFICATION(
             itut_x780::Notifications, stateChange)

    }; // interface SoftwareDistributor

/**
```

9.15 Interface SoftwareDistributor_F

```
*/
    /**
    Objet géré Software Distributor Facade – voir Rec. UIT-T X.780.1
    */

    interface SoftwareDistributor_F : itut_x780::ManagedObject_F
    {
        /**
      La Rec. UIT-T X.731 contient une description de Administrative State
      (état administratif)
        */

        AdministrativeStateType administrativeStateGet
             (in MONameType name)
             raises (itut_x780::ApplicationError);

        void administrativeStateSet
             (in MONameType name,
             in AdministrativeStateType administrativeState)
             raises (itut_x780::ApplicationError);
```

```
   /**
La Rec. UIT-T X.731 contient une description de Operational State (état
opérationnel)
   */

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

   /**
Le service "Deliver" est utilisé par un système gestionnaire pour
demander la distribution d'un logiciel ou d'un ensemble de logiciels.
L'information de livraison identifie le logiciel à distribuer. Le
résultat d'une action de livraison est la fourniture d'une copie des
items de logiciel cible, se trouvant dans l'état "livré", à destination
du système cible.

Le conditionnement des unités logicielles et le choix du mécanisme de
transfert sont du ressort local et sortent du cadre de la présente
Recommandation. Cette information peut, par exemple, être préconfigurée
ou spécifiée dans l'action de remise en même temps que les autres
informations associées.

L'exécution avec succès du service a pour résultat de copier le logiciel
à distribuer sur le système cible; il peut en résulter la création
d'objets "Software Unit" et/ou "Executable Software". Une fois la
commande exécutée, une notification du résultat de livraison est émise.

   @param name                 Nom d'instance d'objet géré Software
                               Distributor
   @param deliverId            Ce paramètre facultatif désigne un
                               identificateur univoque de cette opération
                               livraison.
   @param targetSoftware       Désigne l'origine du logiciel à livrer.
   @param targetSystem         Désigne la destination cible optionnelle
                               concernant le logiciel à livrer. Si cette
                               indication ne correspond pas à une
                               indication cible, le système utilise les
                               moyens locaux afin de déterminer la
                               destination cible.
   @param transferInfo         Mécanisme de transfert propre à
                               l'application.
   @param additionalInfo       Information supplémentaire propre à
                               l'application.
   */

   void deliver
        (in MONameType name,
        in DeliverIdTypeOpt deliverId,
        in TargetSoftwareSetType targetSoftware,
        in DestinationType targetSystem,
        in TransferInfoType transferInfo,
        in AdditionalInformationSetType additionalInfo)
        raises (itut_x780::ApplicationError,
                 OperationStateMismatch,
                 DeliverSoftwareProcessingFailure);

   /**
L'objet géré émet une notification de résultat de livraison lorsque
l'opération livraison est effectuée. Elle contient les résultats finaux
de l'opération et peut indiquer un état de conformité ou de
non-conformité.
   */
```

```
          MANDATORY_NOTIFICATION(
               itut_x744d1::Notifications, deliverResultNotification)
          MANDATORY_NOTIFICATION(
               itut_x780::Notifications, objectCreation)
          MANDATORY_NOTIFICATION(
               itut_x780::Notifications, objectDeletion)
          MANDATORY_NOTIFICATION(
               itut_x780::Notifications, stateChange)

     }; // interface SoftwareDistributor_F

/**
```

9.16 Interface SoftwareDistributorFactory (*atelier distributeur de logiciel*)

```
*/
     /**
     Création et suppression pour Software Distributor
     */

     interface SoftwareDistributorFactory : 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 AdministrativeStateType administrativeState
                              // GET-REPLACE
               )
               raises (itut_x780::ApplicationError,
                              itut_x780::CreateError);

     }; // interface SoftwareDistributorFactory

/**
```

9.17 Notifications

```
*/

     interface Notifications
     {
          /**
       La notification de rapport de sauvegarde est émise pour signaler une
       sauvegarde de l'information représentée par cet objet.

          @param eventTime        Date courante du système géré.
          @param source           Objet qui émet la notification.
          @param request          Indique qui a émis la requête: processus
                                  automatique, système gestionnaire ou système
                                  géré.
          @param backupResult     L'un des résultats suivants, selon le succès ou
                                  l'échec de la sauvegarde et la valeur des
                                  paramètres de l'opération sauvegarde:
                                  - Choix échec – Indication d'erreur
                                  d'application.
                                  - Choix en différé – Emplacement de sauvegarde
                                  en différé.
                                  - Choix objet local – Instance objet de
                                  sauvegarde.
          */
```

```
   void backupReport (
        in ExternalTimeType eventTime,
        in MONameType source,
        in RequestType request,
        in BackupResultType backupResult
   );

   /**
La notification de résultat de livraison est émise à partir de l'objet
géré, une fois l'opération livraison effectuée.

   @param eventTime        Date courante du système géré.
   @param source           Objet qui émet la notification.
   @param deliverId        Identificateur unique de livraison fourni (le
                           cas échéant)avec l'opération livraison.
   @param deliver          Résultats de l'opération livraison. Puisqu'il
                           s'agit de données de type UIDType, la
                           localisation des résultats est possible. Le
                           module DeliverResultConst présente les
                           résultats définis dans la présente
                           Recommandation.
   @param additionalInfo   Information additionnelle propre à
                           l'application.
   */

   void deliverResultNotification (
        in ExternalTimeType eventTime,
        in MONameType source,
        in DeliverIdTypeOpt deliverId,
        in DeliverResultType deliver,
        in AdditionalInformationSetType additionalInfo
   );

   /**
La notification de compte rendu de restauration est émise afin de
signaler une restauration d'objet géré à partir d'une précédente
sauvegarde.

   @param eventTime        Date courante du système géré.
   @param source           Objet qui émet la notification.
   @param request          Indique qui a émis la requête: processus
                           automatique, système gestionnaire ou système
                           géré.
   @param restoreResult    L'un des résultats suivants, selon le succès ou
                           l'échec de la restauration et la valeur de
                           l'attribut source de restauration automatique
                           future:
                           - Choix échec – Indication d'erreur
                           d'application.
                           - Choix en différé – Emplacement de
                           restauration en différé.
                           - Choix objet local – Instance objet de
                           restauration.

   */

   void restoreReport (
        in ExternalTimeType eventTime,
        in MONameType source,
        in RequestType request,
        in RestoreResultType restoreResult
   );
```

```
        /**
      La notification de compte rendu de validation est émise pour signaler une
      validation d'objet géré.

        @param eventTime        Date courante du système géré.
        @param source           Objet qui émet la notification.
        @param validateInfo     Indique l'argument fourni à l'opération de
                                validation.
        @param validateResult   L'un des résultats suivants, selon le succès ou
                                l'échec de la validation:
                                - Validation de conformité (avec résultat)
                                - Validation de non-conformité (avec résultat)
                                - Validation de conformité
                                - Validation de non-conformité (avec erreur)
        */

        void validateReport (
             in ExternalTimeType eventTime,
             in MONameType source,
             in ValidateInfoType validateInfo,
             in ValidateResultType validateResult
        );
    };

    /**
    Constantes de notification
    */

    const string backupReportTypeName =
         "itut_x744d1::Notifications::backupReport";
    const string deliverResultNotificationTypeName =
         "itut_x744d1::Notifications::deliverResultNotification";
    const string restoreReportTypeName =
         "itut_x744d1::Notifications::restoreReport";
    const string validateReportTypeName =
         "itut_x744d1::Notifications::validateReport";
    const string additionalInfoName = "additionalInfo";
    const string backupResultName = "backupResult";
    const string deliverIdName = "deliverId";
    const string deliverName = "deliver";
    const string eventTimeName = "eventTime";
    const string sourceName = "source";
    const string restoreResultName = "restoreResult";
    const string requestName = "request";
    const string validateInfoName = "validateInfo";
    const string validateResultName = "validateResult";

/**
```

9.18 Corrélation de nom

```
*/

    module NameBinding
    {
        /**
      Cette corrélation de nom sert à nommer l'objet Software Distributor
      relativement à l'objet ManagedElement.
        */

        module SoftwareDistributor_ManagedElement
        {
             const string superiorClass = "itut_m3120::ManagedElement";
             const boolean superiorSubclassesAllowed = TRUE;
```

```
        const string subordinateClass =
             "itut_x744d1::SoftwareDistributor";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "SoftwareDistributor";

   }; // module SoftwareDistributor_ManagedElement

   /**
Cette corrélation de nom sert à nommer l'objet Software Distributor
relativement à l'objet Subsystem.
   */

   module SoftwareDistributor_Subsystem
   {
        const string superiorClass = "itut_x780::Subsystem";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::SoftwareDistributor";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "SoftwareDistributor";

   }; // module SoftwareDistributor_Subsystem

   /**
Cette corrélation de nom sert à nommer l'objet Software Distributor
relativement à l'objet System.
   */

   module SoftwareDistributor_System
   {
        const string superiorClass = "itut_x780::System";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::SoftwareDistributor";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "SoftwareDistributor";

   }; // module SoftwareDistributor_System

   /**
Cette corrélation de nom sert à nommer l'objet unité de logiciel
relativement à l'objet Equipment.
   */

   module SoftwareUnit_Equipment
   {
        const string superiorClass = "itut_m3120::Equipment";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::SoftwareUnit";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "SoftwareUnit";
```

```
   }; // module SoftwareUnit_Equipment

   /**
Cette corrélation de nom sert à nommer l'objet unité de logiciel
relativement à l'objet ManagedElement.
   */

   module SoftwareUnit_ManagedElement
   {
        const string superiorClass = "itut_m3120::ManagedElement";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::SoftwareUnit";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "SoftwareUnit";

   }; // module SoftwareUnit_ManagedElement

   /**
Cette corrélation de nom sert à nommer l'objet unité de logiciel
relativement à l'objet Subsystem.
   */

   module SoftwareUnit_Subsystem
   {
        const string superiorClass = "itut_x780::Subsystem";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
             "itut_x744d1::SoftwareUnit";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
             itut_x780::deleteContainedObjects;
        const string kind = "SoftwareUnit";

   }; // module SoftwareUnit_Subsystem

   /**
Cette corrélation de nom sert à nommer l'objet unité de logiciel
relativement à l'objet System.
   */

   module SoftwareUnit_System
   {
        const string superiorClass = "itut_x780::System";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::SoftwareUnit";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "SoftwareUnit";

   }; // module SoftwareUnit_System

   /**
Cette corrélation de nom sert à nommer l'objet Executable Software
relativement à l'objet Equipment.
   */
```

```
   module ExecutableSoftware_Equipment
   {
        const string superiorClass = "itut_m3120::Equipment";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::ExecutableSoftware";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "ExecutableSoftware";

   }; // module ExecutableSoftware_Equipment

   /**
Cette corrélation de nom sert à nommer l'objet Executable Software
relativement à l'objet ManagedElement.
   */

   module ExecutableSoftware_ManagedElement
   {
        const string superiorClass = "itut_m3120::ManagedElement";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::ExecutableSoftware";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "ExecutableSoftware";

   }; // module ExecutableSoftware_ManagedElement

   /**
Cette corrélation de nom sert à nommer l'objet Executable Software
relativement à l'objet Subsystem.
   */

   module ExecutableSoftware_Subsystem
   {
        const string superiorClass = "itut_x780::Subsystem";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::ExecutableSoftware";
        const boolean subordinateSubclassesAllowed = TRUE;
        const boolean managerCreatesAllowed = TRUE;
        const DeletePolicyType deletePolicy =
                 itut_x780::deleteContainedObjects;
        const string kind = "ExecutableSoftware";

   }; // module ExecutableSoftware_Subsystem

   /**
Cette corrélation de nom sert à nommer l'objet Executable Software
relativement à l'objet System.
   */

   module ExecutableSoftware_System
   {
        const string superiorClass = "itut_x780::System";
        const boolean superiorSubclassesAllowed = TRUE;
        const string subordinateClass =
                 "itut_x744d1::ExecutableSoftware";
```

```
                const boolean subordinateSubclassesAllowed = TRUE;
                const boolean managerCreatesAllowed = TRUE;
                const DeletePolicyType deletePolicy =
                          itut_x780::deleteContainedObjects;
                const string kind = "ExecutableSoftware";

           }; // module ExecutableSoftware_System

     }; // module NameBinding

/**
```

9.19 Module FileTypeConst

```
*/

     module FileTypeConst
     {
          const string moduleName = "itut_x744d1::FileTypeConst";

          const short unstructuredText = 0; // FTAM-1
          const short unstructuredBinary = 1; // FTAM-3
          const short blockSpecial = 2;

     }; // end of module FileTypeConst

/**
```

9.20 Module DeliverResultConst

```
*/

     module DeliverResultConst
     {
          const string moduleName = "itut_x744d1::DeliverResultConst";

          const short pass = 0;
          const short communicationError = 1;
          const short equipmentError = 2;
          const short qosError = 3;
          const short accessDenied = 4;
          const short notFound = 5;
          const short insufficientSpace = 6;
          const short alreadyDelivered = 7;
          const short inProgress = 8;
          const short unknown = 9;

     }; // end of module DeliverResultConst

}; // module itut_x744d1

#endif //_itut_x744_1_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

Imprimé en Suisse
Genève, 2003