UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T X.780.1

(08/2001)

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

Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA à granularité grossière

Recommandation UIT-T X.780.1

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.780.1

Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA à granularité grossière

Résumé

La présente Recommandation spécifie les extensions apportées à un ensemble de directives de modélisation d'objets gérés CORBA du RGT afin que les interfaces à granularité grossière puissent être prises en charge. Elle spécifie la façon dont ces interfaces CORBA du RGT doivent être définies. Elle contient également des directives concernant la conversion d'interfaces à granularité fine en des interfaces à granularité grossière. Un module IDL CORBA définissant les types d'interfaces de base à élargir y est défini.

Source

La Recommandation X.780.1 de l'UIT-T, élaborée par la Commission d'études 4 (2001-2004) de l'UIT-T, a été approuvée le 13 août 2001 selon la procédure définie dans la Résolution 1 de l'AMNT.

Mots clés

Architecture commune de courtage pour les requêtes sur des objets (CORBA), directives pour la définition des objets gérés (GDMO), interfaces du RGT, langage de définition d'interface (IDL), objets gérés, traitement décentralisé.

AVANT-PROPOS

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

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

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

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

NOTE

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

DROITS DE PROPRIÉTÉ INTELLECTUELLE

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

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Recommandation UIT-T X.780.1

Directives concernant le RGT pour la définition d'interfaces d'objets gérés CORBA à granularité grossière

1 Domaine d'application

L'architecture du RGT définie dans la Rec. UIT-T M.3010 (2000) utilise les concepts découlant du traitement réparti et inclut l'utilisation de plusieurs protocoles de gestion. Les Rec. UIT-T Q.816 et X.780 définissent dans cette architecture un cadre d'application pour l'architecture de courtier commun de requêtes d'objets (CORBA) comme étant l'un des protocoles de gestion du RGT.

La présente Recommandation, ainsi que la Rec. UIT-T Q.816.1, vient compléter les spécifications de ce cadre afin de lui permettre de prendre en charge un style d'interaction entre systèmes gérants et systèmes gérés légèrement différents de celui spécifié dans le ou dans les documents traitant du cadre original. Ce style d'interaction présente certains avantages dont le principal est qu'il dispense un système gérant de devoir extraire une adresse de logiciel orientée objet pour chaque ressource gérable à laquelle il souhaite accéder. Le nombre de ces adresses logicielles peut atteindre des millions dans les grands systèmes. Il modifie quelque peu la façon dont le logiciel est structuré sur les systèmes gérés, ce que certains fournisseurs de systèmes gérés peuvent préférer.

Le domaine d'application de la présente Recommandation est le même que le cadre CORBA du RGT d'origine. Le cadre et ses extensions couvrent toutes les interfaces dans le RGT où l'architecture CORBA peut être utilisée. Il est probable toutefois que toutes les capacités et services définis ici ne seront pas tous nécessaires dans toutes les interfaces du RGT. De ce fait, le cadre peut être utilisé pour les interfaces entre des systèmes de gestion à tous les niveaux d'abstraction (inter et intra-administration) ainsi qu'entre tous les systèmes de gestion et tous les éléments de réseau.

1.1 Objectif

L'objet de la présente Recommandation est d'étendre le cadre CORBA du RGT pour lui permettre d'être utilisé dans un plus large éventail d'applications. Les extensions permettent un mode légèrement différent d'interaction entre les systèmes gérants et les systèmes gérés, mode que l'on peut préférer d'utiliser dans de nombreuses situations. La présente Recommandation est donc destinée à être utilisée par divers groupes qui spécifient des interfaces de gestion de réseau.

1.2 Application

L'approche utilisée dans les Recommandations relatives au cadre CORBA du RGT consiste à modéliser les ressources de réseau gérables sous forme d'objets logiciels accessibles en utilisant l'architecture CORBA. Les modèles d'information écrits dans le langage de définition d'interface CORBA (IDL, *interface definition language*) décrivent les interfaces objets.

L'architecture CORBA assure une transparence de lieu, ce qui permet à un objet logiciel d'interagir avec un autre indépendamment de sa localisation. L'accès à un objet logiciel se fait en utilisant ce que CORBA désigne par référence d'objet interopérable (IOR, *interoperable object reference*).

Le cadre CORBA du RGT original modélise chaque ressource gérable par un objet CORBA indépendant, auquel est associée une référence IOR unique. Cette approche permet, de manière souple, à chaque objet de se trouver en un lieu quelconque. Elle impose toutefois au système gérant d'avoir à disposition une référence IOR pour chaque objet auquel il souhaite accéder. Il s'agit d'une charge supplémentaire que de nombreuses entreprises et administrations du secteur des télécommunications auraient souhaité éviter. Elle pourrait également exiger qu'un système géré prenne en charge un grand nombre de références IOR, ce que les fournisseurs de systèmes gérés

voudraient aussi éviter. La présente Recommandation, ainsi que la Rec. UIT-T Q.816.1, définit la manière dont le cadre CORBA du RGT doit être élargi afin d'éviter qu'il soit nécessaire de disposer d'un grand nombre de références IOR.

Les interfaces CORBA utilisant cette approche où chaque ressource gérable est adressable avec une référence IOR unique, sont devenues ce que l'on appelle des interfaces à granularité fine. En revanche, celles pour lesquelles une référence IOR n'est pas assignée à chaque ressource gérable sont appelées interfaces à granularité grossière.

Etant donné que la présente Recommandation définit une approche légèrement différente de modélisation des ressources gérées sur des interfaces à granularité grossière, les spécifications des modèles d'interface seront légèrement différentes pour l'approche à granularité fine et pour l'approche à granularité grossière.

1.3 Structure de la Recommandation

La présente Recommandation a la structure suivante:

Paragraphe 1	Introduction, structure et mise à jour
Paragraphe 2	Références
Paragraphes 3 et 4	Définitions et abréviations utilisées dans l'ensemble de la présente Recommandation
Paragraphe 5	Conventions
Paragraphe 6	Les considérations relatives à la conception qui doivent être étudiées pour la prise en charge des interfaces à granularité grossière sont ajoutées au cadre
Paragraphe 7	Descriptions du cadre CORBA du RGT et conditions associées à la granularité grossière
Paragraphe 8	Interface de façade pour l'accès aux objets gérés. Ce paragraphe traite des interfaces spécifiques aux modèles qui doivent être mis en œuvre dans les interfaces à granularité grossière
Paragraphe 9	Directives pour la définition des interfaces CORBA à granularité grossière
Paragraphe 10	Directives pour la transposition des spécifications d'interfaces CORBA à granularité fine en spécifications d'interfaces à granularité grossière
Paragraphe 11	Directives relatives à la conformité
Annexe A	Module IDL concernant la spécification des directives de modélisation à granularité grossière. Cette annexe est normative.

2 Références normatives

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

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

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

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

[4] Document OMG/99-10-07, *Architecture commune de courtage pour les requêtes sur des objets et spécifications*, Révision 2.3.1.

3 Définitions

3.1 Définitions reprises de la Rec. UIT-T X.701

Les termes suivants, utilisés dans la présente Recommandation, sont définis dans la Rec. UIT-T X.701:

– classe d'objets gérés;

– gestionnaire;

– agent.

3.2 Définitions reprises de la Rec. UIT-T X.703

Le terme suivant, utilisé dans la présente Recommandation, est défini dans la Rec. UIT-T X.703, Architecture de gestion répartie ouverte:

– notification.

3.3 Définition additionnelle

3.3.1 façade: interface d'objet définie pour permettre l'accès à un ensemble d'objets gérés appartenant tous à la même classe.

4 Abréviations

Les abréviations suivantes sont utilisées dans la présente Recommandation:

CMIP protocole commun d'informations de gestion (*common management information protocol*)

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

COS services communs aux objets (*common object services*)

DN nom distinctif (*distinguished name*)

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

GDMO directives pour la définition d'objets gérés (*guidelines for the definition of managed objects*)

ID identificateur

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

IIOP protocole d'interopérabilité Internet (*Internet interoperability protocol*)

IOR référence d'objet interopérable (*interoperable object reference*)

MO objet géré (*managed object*)

NE élément de réseau (*network element*)

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

OAM&P exploitation, administration, maintenance et fourniture (*operations, administration, maintenance, and provisioning*)

OID identificateur d'objet (*object identifier*)

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

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

OSI interconnexion des systèmes ouverts (*open systems interconnection*)

PDU unité de données protocolaire (*protocol data unit*)

POA adaptateur d'objet portable (*portable object adapter*)

QS qualité de service

RDN nom distinctif relatif (*relative distinguished name*)

RGT réseau de gestion des télécommunications

UID identificateur universel (*universal identifier*)

UIT-T Union internationale des télécommunications – Secteur de la normalisation des télécommunications

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

UTC temps universel coordonné (*universal time coordinated*)

5 Conventions

5.1 Conventions de la présente Recommandation

Quelques conventions sont utilisées dans la présente Recommandation pour permettre au lecteur d'être informé de l'objet du texte. Bien qu'une grande partie de la Recommandation ait un caractère normatif, les paragraphes qui, succinctement, énoncent des conditions obligatoires à remplir par un système de gestion (gérant ou géré), sont précédés par un **(R)** en gras entre parenthèses, suivi d'un nom bref indiquant l'objet de la condition, et par un numéro, par exemple:

(R) EXEMPLE-1 Exemple d'une condition obligatoire.

Les conditions qui peuvent être mises en œuvre facultativement par un système de gestion sont précédées d'un **(O)** en gras à la place du **R**, par exemple:

(O) OPTION-1 Exemple d'une condition optionnelle.

Les déclarations de prescription sont utilisées pour créer des profils de conformité.

La présente Recommandation contient de nombreux exemples de langage IDL CORBA, et de langage IDL définissant les services spécifiques du RGT, et les types de données de prise en charge, inclus dans l'Annexe A. Le langage IDL est écrit avec une police de caractères de type courrier de 9 points.

```
// Exemple IDL
interface foo {
     void operation1 ();
};
```

5.2 Compilation de l'IDL

Un avantage lié à l'utilisation de l'IDL pour spécifier les interfaces de gestion d'un réseau est que ce langage peut être compilé en un code de programmation par des outils qui accompagnent un courtier ORB. Cela permet d'automatiser effectivement le développement de certains codes nécessaires à l'interopérabilité des applications de gestion de réseau. L'Annexe A contient un code que les réalisateurs voudront extraire et compiler. L'Annexe A a un caractère normatif et doit être utilisée par les développeurs mettant en œuvre des systèmes conformes à la présente Recommandation. L'IDL dans la présente Recommandation peut être vérifié en utilisant deux compilateurs pour s'assurer de son caractère correct. Il faut utiliser un compilateur prenant en charge la version CORBA spécifiée dans la Rec. UIT-T Q.816.

L'Annexe A a été formattée de manière à pouvoir les couper et les coller dans des fichiers en format texte compilables. Nous donnons dans ce qui suit quelques indications pratiques sur la façon de le faire.

1) Le coupage/collage semble mieux fonctionner avec la version Microsoft® Word® utilisée dans la présente Recommandation. La même double opération à partir du format de fichier Adobe® Acrobat® semble inclure les en-têtes et les bas de page qui ne peuvent pas être compilés.
2) Dans toute l'Annexe A, ce qui commence avec la ligne "/* Ce code IDL ..." doit être stocké dans un fichier appelé "itut_x780_1.idl" dans un annuaire où il sera retrouvé par le compilateur IDL.
3) Il n'est pas nécessaire de supprimer les en-têtes intégrés dans l'Annexe A. Elles ont été encapsulées dans les commentaires IDL et seront ignorées par le compilateur.
4) Les commentaires qui commencent par la séquence spéciale "/**" sont reconnus par les compilateurs qui convertissent le langage IDL en langage HTML. Ces commentaires ont souvent des instructions spéciales de formatage pour ces compilateurs. Les personnes qui travailleront en IDL voudront peut-être générer le HTML étant donné que les fichiers HTML résultant ont des liens qui permettent une navigation rapide à travers les fichiers.
5) L'Annexe A a été formatée avec des espaces de tabulation à intervalles de 8 espaces et des retours à la ligne réels qui permettent d'utiliser la plupart des éditeurs de textes.

6 Considérations relatives à la conception des interfaces à granularité grossière

Le présent paragraphe expose plusieurs considérations concernant la conception et qui doivent être prises en compte par le cadre afin de pouvoir prendre en charge les interfaces à granularité grossière.

6.1 Création et suppression d'objet à granularité grossière

Il doit être possible de créer et de supprimer des représentations à granularité grossière de ressources gérées. La possibilité d'inclure l'opération créée sur l'interface à granularité grossière doit être explorée.

6.2 Attributs

Le cadre doit prendre en charge les attributs d'association avec les ressources gérées auxquelles l'accès s'effectue par une interface à granularité grossière.

6.3 Notification

Le cadre doit prendre en charge les notifications d'événements provenant des ressources gérées auxquelles l'accès s'effectue via des interfaces à granularité grossière.

6.4 Accès à granularité grossière de toutes les ressources gérées

Le cadre doit permettre, et exiger des implémentations pour qu'il soit possible aux systèmes gérants d'accéder à toutes les ressources gérées via des interfaces à granularité grossière.

6.5 Exceptions

Les interfaces à granularité grossière doivent permettre aux ressources gérées de lever une exception lors de l'invocation d'une opération. Ces exceptions doivent explicitement être clarifiées pour chaque opération.

6.6 Prise en charge de toutes les opérations

Une interface à granularité grossière doit prendre en charge toutes les opérations applicables à une ressource gérée.

6.7 Mappage prescriptif

Le mappage entre des modèles d'information à granularité fine et des modèles d'information à granularité grossière doit être prescriptif. Le mappage doit pouvoir être effectué de manière algorithmique. Si un modèle d'information IDL est élaboré par traduction d'un modèle d'information GDMO, toutes les optimisations exécutées manuellement pendant la traduction doivent apparaître à la fois dans les modèles à granularité fine et les modèles à granularité grossière.

6.8 Extraction d'attributs depuis plusieurs objets

Il est nécessaire d'extraire des attributs d'instances d'objets gérés multiples en une seule opération fortement typée. Etant donné que l'accès à tous ces objets se ferait en utilisant la même façade, cette opération peut être exécutée au moyen d'une opération unique exécutée par la façade.

7 Description générale du cadre et des conditions associées

Le paragraphe 6 décrivait les considérations relatives à la conception pour lesquelles on devait trouver des solutions pour pouvoir prendre en charge les interfaces à granularité grossière, est ajoutée au cadre. Le présent paragraphe et la suite de la présente Recommandation contiennent des détails sur la façon dont le cadre sera élargi pour traiter du sujet. Dans la Rec. UIT-T Q.816.1 l'accent est mis sur les services de prise en charge du cadre pour les interfaces à granularité grossière, alors que dans la présente Recommandation sont définies les directives de développement de modèles d'information associés aux interfaces à granularité grossière. Dans un premier temps sera présentée une description générale du cadre actuel, elle sera suivie des extensions qui lui ont été apportées.

7.1 Description générale du cadre

Le cadre applicable aux interfaces au RGT fondé sur l'architecture CORBA est constitué d'un ensemble de capacités. L'élément central de ce cadre est un ensemble de services d'objets communs OMG. Le cadre définit leur rôle dans les interfaces de gestion de réseau ainsi que les conventions associées à leur utilisation. Le cadre définit également les services de prise en charge qui n'ont pas été normalisés sous forme de services d'objets communs OMG, mais dont on peut prévoir qu'ils deviennent une norme sur les interfaces de gestion de réseau conformes au cadre.

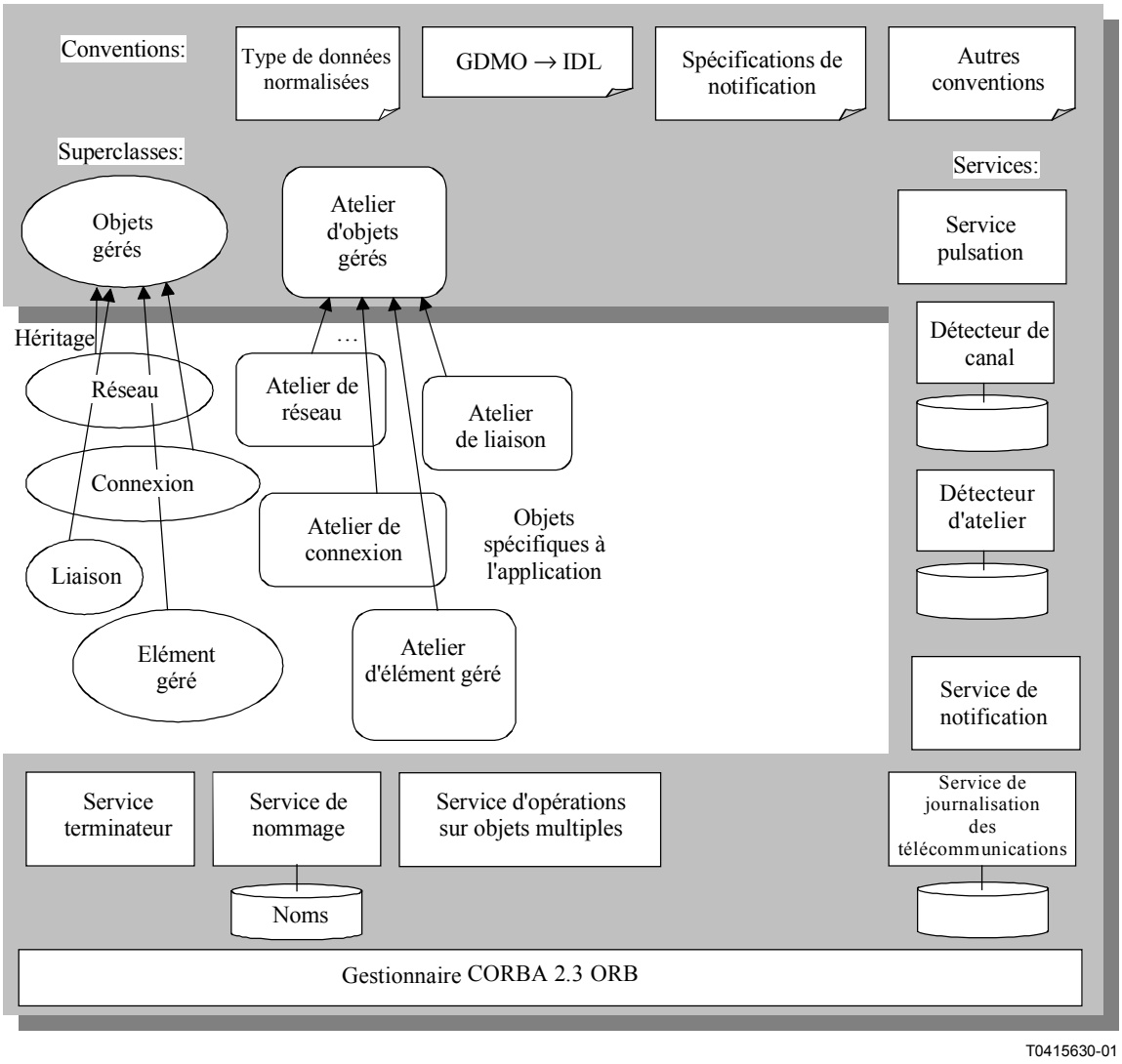

Figure 1/X.780.1 – Description générale du cadre

Le cadre est représenté de manière graphique à la Figure 1. Cette figure montre le cadre en gris. Au centre de la figure se trouvent les objets applicatifs qui sont pris en charge par le cadre. Sur le bas de la figure se trouve une case représentant le gestionnaire ORB CORBA. Au-dessus se trouve un certain nombre de cases dont les noms représentent les services composant le cadre général. (Certaines cases ont également des icônes qui décrivent les bases de données dont elles devront conserver la maintenance afin d'exécuter leurs fonctions.) Ces services, ainsi que les conditions associées à la version de l'ORB, sont définis dans la Rec. UIT-T Q.816. En haut de la figure, des icônes représentent deux superclasses, l'une pour les objets gérés, l'autre pour les ateliers d'objets gérés. Chacun des objets gérés et des ateliers d'objets gérés correspondants, pris en charge par ce cadre, doivent finalement hériter de ces superclasses respectives. La figure montre également des icônes de pages aux coins repliés, qui représentent des conventions normalisées de modélisation d'objet. Ces conventions et les superclasses sont définies dans la Rec. UIT-T X.780.

7.2 Description générale des extensions associées à la granularité grossière

Le présent paragraphe contient une description des extensions qu'il est nécessaire d'apporter au cadre pour pouvoir prendre en charge les interfaces à granularité grossière.

7.2.1 Séquence de conception de la façade

La modification la plus importante qui est apportée au cadre afin de prendre en charge des interfaces à granularité grossière porte sur les modalités d'accès aux objets gérés. Contrairement aux références IOR prises en charge par le système, le nombre d'objets gérés dans un système géré doit pouvoir augmenter. Il est toutefois toujours souhaitable que l'accès aux objets gérés reste fortement typé. Cela conduit à l'utilisation d'une séquence de conception appelée ici la séquence "façade". On peut représenter une façade comme étant une fausse devanture ou un portail. En utilisant la séquence de conception de la façade, un système géré prendra en charge un petit nombre d'interfaces de façade, une au minimum, mais en général, un nombre assez faible pour chaque type d'objet géré du système. Un système gérant invoquera alors une opération sur l'objet géré en invoquant réellement l'opération sur une façade pour ce type d'objet géré dans le système. Dans la séquence de conception de la façade, les objets gérés ne doivent pas exposer d'interface CORBA et par conséquent ils peuvent ne pas avoir de référence IOR individuelle. Cela signifie qu'il n'est pas nécessaire pour un système géré prenant en charge l'approche façade de mettre en œuvre des interfaces d'objet géré à granularité fine.

Il vaut mieux ne pas considérer une façade comme un objet géré, mais plutôt comme un objet intermédiaire qui permet à un système géré d'accéder à des objets gérés. L'objet façade a une interface CORBA et est accessible en utilisant le CORBA. Les objets gérés toutefois, peuvent éventuellement ne pas disposer d'interfaces CORBA et n'être pas directement accessibles en utilisant le CORBA. La façade elle-même ne représente pas une ressource de réseau gérable; elle a pour objet de permettre une interaction avec les objets qui représentent des ressources gérables. Tous les objets de façade sont créés automatiquement par le système géré, et existent aussi longtemps que les objets sont accessibles via la façade. Plusieurs façades pour le même type d'objet géré peuvent exister sur une interface à granularité grossière, mais un objet géré doit être accessible via une seule façade. Voir la Figure 2 ci-dessous.

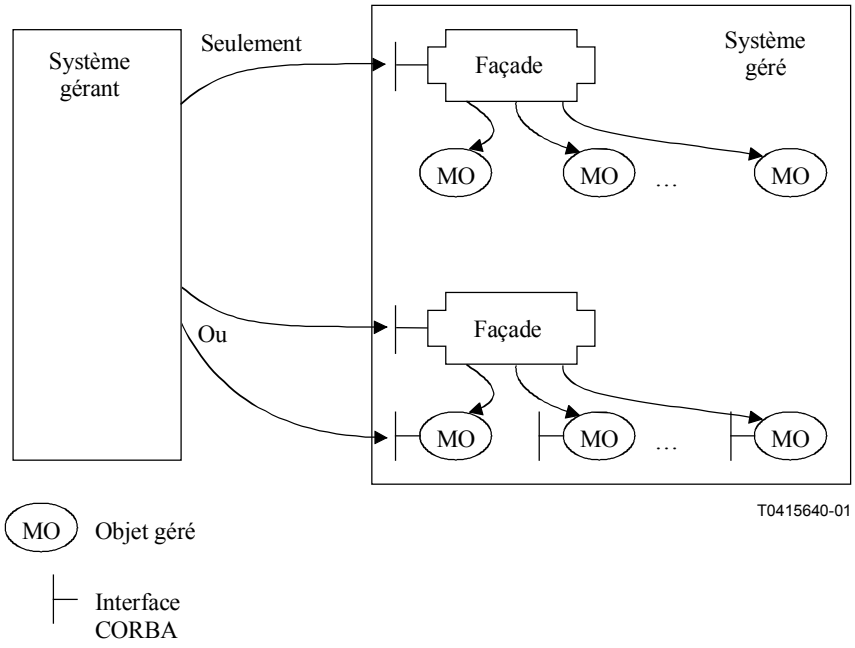

Figure 2/X.780.1 – Rôle de la façade

La figure montre un système gérant qui accède à un système géré prenant en charge l'approche granularité grossière. Le système géré dispose de deux interfaces de façade qui permettent au système gérant d'accéder à deux ensembles différents d'objets gérés. Les objets gérés en haut de la figure peuvent uniquement être accessibles via la façade. Les objets gérés au bas de la figure prennent également en charge des interfaces CORBA directes et sont accessibles via la façade ou

directement. L'accès CORBA direct est facultatif, mais un système géré qui prend en charge l'approche façade doit disposer d'interfaces de façade pour chacune de ses instances d'objets gérés.

Une façade peut utiliser une interface CORBA d'objets gérés pour invoquer une opération sur elle, ou d'autres moyens spécifiques à l'implémentation. En fait, un système géré, n'a même pas besoin de mettre en œuvre des objets gérés comme objets individuels de manière interne. En mettant en œuvre une interface fondée sur ce cadre toutefois, on aura l'illusion que les objets gérés sont mis en œuvre de manière interne comme des objets.

Lorsqu'une opération est invoquée sur un objet géré via la façade, la façade doit alors invoquer l'opération sur l'entité ou l'objet réel géré. Etant donné que l'accès à de nombreux objets gérés se fera via une seule façade, la façade doit savoir quel objet géré est la cible réelle de l'opération. Pour cela, on adoptera une convention incluant le nom de l'objet géré cible comme premier paramètre de chaque opération de façade dirigée vers un objet géré.

Tandis qu'il se peut que les objets gérés n'aient plus de référence IOR unique, ils auront toujours des noms uniques et pourront encore être considérés comme des entités individuelles représentant des ressources gérables.

7.2.2 Extension des noms d'objets gérés

Comme indiqué plus haut, les objets gérés auxquels l'accès se fait via une façade auront toujours un nom même s'ils peuvent ne pas disposer d'une interface CORBA individuelle. Il est important qu'un système gérant puisse déterminer quelle façade utiliser sur la base du nom d'objet géré. S'il ne peut le faire, il devra interroger le système géré ou associer de manière permanente une référence IOR de façade à chaque nom d'objet géré. Pour pouvoir prendre en charge la capacité à déterminer une façade d'objet géré basée uniquement sur son nom, les noms des objets gérés accessibles via une façade sont pourvus d'une petite extension après les noms des objets gérés non accessibles via la façade. Dans la composante finale de nom, qui a toujours une chaîne *ID* avec la valeur "object" (ou, étendue par la Rec. UIT-T Q.816.1 à savoir, <empty>), la chaîne *kind* (sorte) est mise à la valeur d'un identificateur de façade assigné à la façade par laquelle on peut accéder à l'objet. Pour les objets gérés non accessibles via une façade, cette chaîne *kind* est vide. La Rec. UIT-T Q.816.1 contient des détails complémentaires sur la façon dont la chaîne *kind* dans la composante nom d'objet géré final est utilisée pour identifier la façade.

7.2.3 Prise en charge des services pour les objets gérés accessibles par la façade

Les services de prise en charge du cadre fournis sur les interfaces qui utilisent l'approche par façade seront essentiellement les mêmes que ceux définis dans la Rec. UIT-T Q.816. Certains de ces services, tels le détecteur d'atelier et le détecteur de canaux, nécessitent de ne pas apporter de modification. D'autres, tels le service terminateur et le service d'opération sur objets multiples (MOO, *multiple object operation*), nécessitent de ne pas apporter de modification à leurs interfaces ou sur la façon dont ils sont utilisés par les systèmes gérants, mais peuvent nécessiter de légères modifications en ce qui concerne leur implémentation, s'ils accèdent à des objets gérés en utilisant les interfaces de façade d'objets gérés (de préférence à certaines méthodes propres à l'implémentation). La Rec. UIT-T Q.816.1 contient des détails sur les modifications au service de prise en charge du support qu'il faut apporter pour pouvoir prendre en charge les interfaces à granularité grossière.

La modification la plus importante aux services de prise en charge concerne le domaine de prise en charge pour le nommage. Les interfaces à façade sont liées aux noms dans le service de nommage, en grande partie de la même façon que les interfaces de service de prise en charge. Dans les interfaces qui utilisent les façades toutefois, les noms des objets gérés ne doivent pas nécessairement être liés aux références IOR dans le service de nommage OMG. En effet, un nouveau service est introduit comme un endroit où l'on stocke les noms d'objets gérés et l'information de relation de confinement. Ce nouveau service, le service de confinement, est défini dans la Rec. UIT-T Q.816.1.

7.2.4 Modélisation de la façade

Afin de pouvoir prendre en charge la séquence de conception des façades et la définition des façades utilisable dans le présent cadre, on a défini une nouvelle interface de base. Cette interface est appelée l'interface de façade d'objet géré. Elle joue le même rôle dans les interfaces à granularité grossière que celui joué par l'interface d'objet géré dans les interfaces à granularité fine. C'est-à-dire, qu'elle constitue l'interface de base à partir de laquelle toutes les interfaces à façade d'objet géré doivent hériter directement ou indirectement pour pouvoir fonctionner dans le cadre. L'interface de façade d'objet géré est assez analogue à l'interface d'objet géré définie dans la Rec. UIT-T X.780. On se reportera au paragraphe 8 pour la définition de l'interface de façade d'objet géré.

Les modifications apportées au cadre sont illustrées à la Figure 3 ci-dessous. Une nouvelle superclasse, la *ManagedObjectFacade*, est ajoutée à la figure, ainsi que le service de confinement. Il convient de noter qu'elle permet l'accès à une base de données de noms d'objets gérés. La base de données de noms d'objets gérés entretenue par le service de nommage est représentée avec des lignes en pointillé, indiquant qu'il n'est pas nécessaire de stocker les noms et les références IOR des objets gérés. Le service de nommage est toujours requis toutefois, pour permettre au système gérant de trouver des interfaces de façade et prendre en charge des références de service. Finalement, les objets gérés sont également représentés par des lignes en pointillé pour indiquer qu'elles n'ont pas besoin d'être directement accessibles.

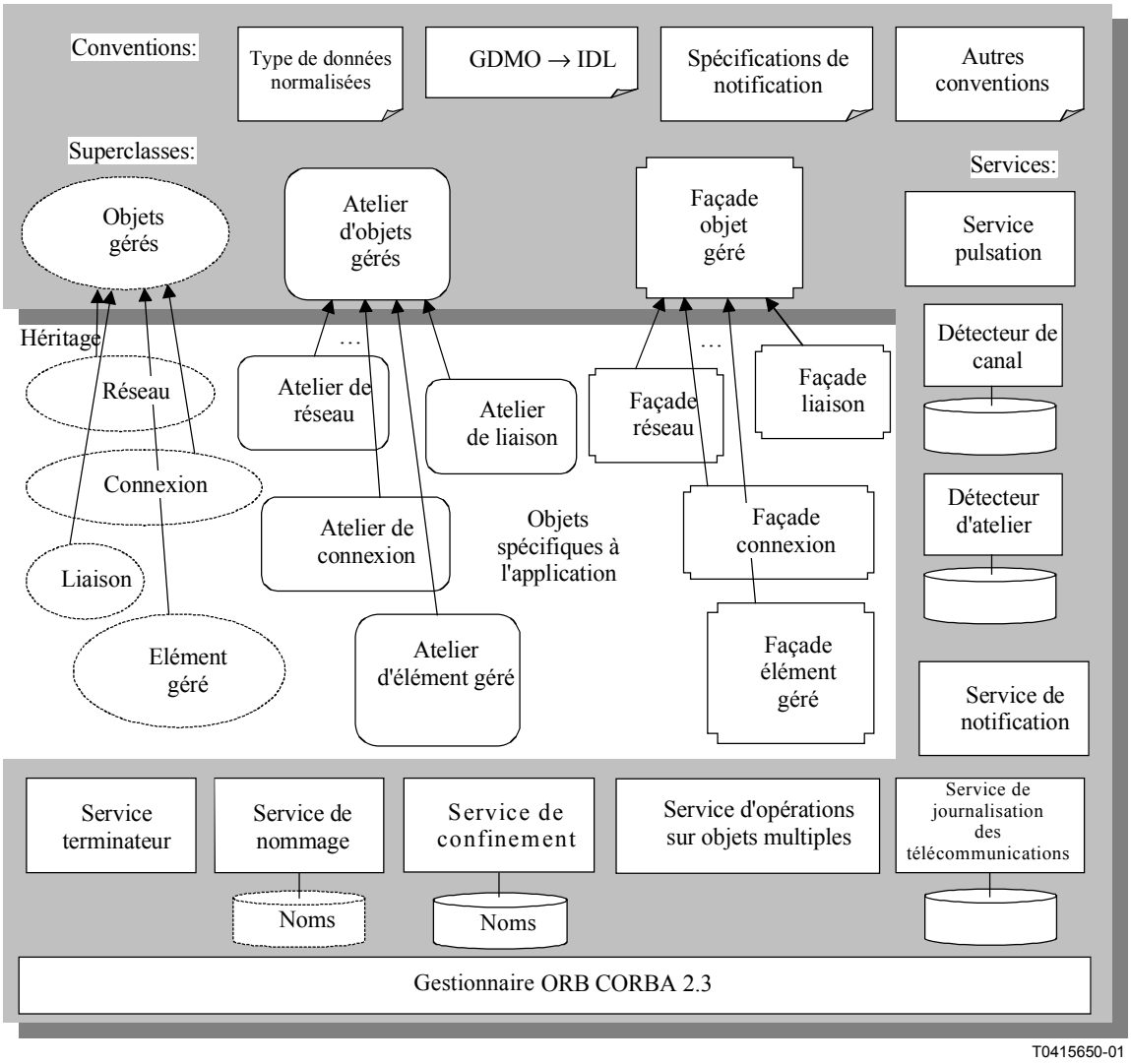

Figure 3/X.780.1 – Cadre avec extension permettant la prise en charge des interfaces à granularité grossière

8 Définition des interfaces de façade pour l'accès aux objets gérés

Comme décrit ci-dessus, les interfaces de façade sont un moyen différent d'accès aux objets gérés. Dans une interface à granularité grossière (dans laquelle les interfaces de façade sont présentes), les objets gérés peuvent ou ne peuvent pas être accessibles en utilisant des interfaces individuelles CORBA. Ainsi, un système gérant peut utiliser une façade pour accéder aux objets gérés en invoquant des opérations sur la façade. Même si le système géré ne met pas en œuvre des objets gérés sous forme d'objets séparés, s'il est compatible avec la présente Recommandation il donne l'illusion de le faire.

8.1 Instanciation de la façade

Le présent paragraphe définit les conditions que doivent respecter les systèmes gérés lorsqu'ils fournissent des interfaces de façade pour l'accès aux objets gérés.

(R) FACADE-1 – Un système géré doit offrir au moins une interface de façade pour chaque classe d'objets gérés susceptibles d'être instanciés sur lui, même si ces objets prennent en charge des interfaces CORBA directes. Les interfaces de façade pour les classes d'objets gérés qui ne peuvent pas être instanciés dans le système ne doivent pas être fournies. Cela inclut les superclasses. Ainsi, une façade pour une superclasse d'objets gérés ne doit pas nécessairement être instanciée si des objets gérés de ce type peuvent ne pas être instanciés, même si des objets gérés de la sous-classe peuvent être instanciés. Toutefois cela ne signifie pas qu'il faille définir des interfaces de superclasse. La définition des interfaces de superclasse suit la même hiérarchie d'héritage que les objets gérés. Voir les § 9 et 10 ci-dessous. Les conditions de lien de noms pour les interfaces de façade sont définies dans la Rec. UIT-T Q.816.1.

Le fait de dispenser le système géré de fournir des interfaces de façade pour des superclasses qui ne sont pas instantiables sur le système, n'empêche pas au système gérant de tirer avantage du polymorphisme. Un système gérant peut, en tout état de cause, traiter une interface de façade comme une superclasse de façade et les capacités d'accès disponibles via la superclasse de la même façon que tout client CORBA peut traiter un objet comme une superclasse. L'opération toutefois sera réellement invoquée sur la façade de sous-classe, en utilisant la référence IOR de cette façade. Le langage de programmation sur le système client gère le polymorphisme.

(R) FACADE-2 – Un système géré peut fournir plusieurs interfaces de façade pour une classe donnée d'objets gérés. Des objets gérés accessibles via une interface de façade ne doivent pas être accessibles par une autre. La Rec. UIT-T Q.816.1 donne des détails sur la manière à utiliser pour identifier la façade afin d'accéder à un objet géré en se basant sur son nom.

(R) FACADE-3 – Toutes les interfaces de façade sont créées et détruites par le système géré. Une interface de façade doit exister pendant tout le temps où existe l'un quelconque des objets gérés accessibles par elle. Aucune notification n'est envoyée lorsqu'une façade est créée ou supprimée. Par conséquent, un système gérant doit devenir informé de l'existence d'une nouvelle façade si et lorsque commence à apparaître des objets gérés avec des noms indiquant qu'ils sont accessibles via la nouvelle façade.

(R) FACADE-4 – Chaque opération de façade qui est dirigée en un objet géré spécifique contiendra le nom de cet objet dans le premier paramètre de l'opération. La façade devra invoquer cette opération sur l'objet géré nommé et renvoyer le résultat y compris les exceptions. Si l'objet géré nommé n'est pas accessible via la façade considérée, la façade déclenchera une exception *applicationError* en réponse à l'opération. Le code d'erreur correspondant à cette exception doit être émis à *objectNotFound* défini dans le langage IDL à l'Annexe A.

8.2 Classe de base d'interface de façade

Le présent paragraphe décrit une classe de base de façade d'objets gérés définie en langage IDL dans l'Annexe A. Toutes les interfaces de façade d'objets gérés fournies dans un système géré doivent être héritières, directes ou indirectes, de cette interface. Cette interface fournit un ensemble d'opérations de base que toutes les interfaces de façade doivent prendre en charge pour être utilisables dans le présent cadre.

L'interface de façade d'objets gérés, appelé "*ManagedObject_F*", en langage IDL dans l'Annexe A, joue le même rôle sur les interfaces à granularité grossière que l'interface *ManagedObject* joue pour les interfaces à granularité fine. Ainsi, cette interface ressemble beaucoup à l'interface *ManagedObject* définie dans la Rec. UIT-T X.780. La plupart des opérations sur l'interface *ManagedObject_F* sont pratiquement identiques aux opérations sur l'interface *ManagedObject.* Une nouvelle opération *attributesBulkGet,* est ajoutée et une autre, *nameGet,* supprimée.

8.2.1 Capacités de base des façades d'objets gérés

Les capacités que doivent prendre en charge toutes les interfaces de façade d'objets gérés sont les suivantes:

- une méthode renvoyant le nom de classe d'un objet géré nommé;
- une méthode renvoyant les paquetages conditionnels pris en charge par un objet géré nommé;
- une méthode permettant de renvoyer la source de création d'un objet géré nommé (qu'il ait été créé de manière autonome par la ressource gérée, en réponse à une opération de gestion, ou que sa source de création soit inconnue);
- une méthode permettant de renvoyer la politique de suppression d'un objet géré nommé. Il s'agit d'une valeur énumérée et elle indique si l'objet n'est pas effaçable, s'il est effaçable seulement s'il ne contient pas d'objet, ou si tous les objets contenus seront supprimés lorsqu'il sera supprimé;
- une méthode qui permet de renvoyer un objet de type valeur CORBA contenant tous les attributs lisibles pour l'objet géré nommé;
- une méthode permettant de renvoyer un objet de type valeur CORBA contenant tous les attributs lisibles pour un ensemble d'objets gérés;
- une opération de destruction.

8.2.2 Séquence IDL associée à une façade d'objets gérés

La séquence IDL décrivant l'interface *ManagedObject_F* (sans commentaire) est la suivante:

```
interface ManagedObject_F {
     ObjectClassType objectClassGet(in NameType name)
          raises (ApplicationError);

     StringSetType packagesGet (in NameType name)
          raises (ApplicationError);

     SourceIndicatorType creationSourceGet(in NameType name)
          raises (ApplicationError);

     DeletePolicyType deletePolicyGet (in NameType name)
          raises (ApplicationError);

     ManagedObjectValueType attributesGet (
          in        NameType name,
          inout     StringSetType attributeNames)
          raises    (ApplicationError);
```

```
        boolean attributesBulkGet (
             in        NameSetType                    names,
             in        StringSetType                  attributeNames,
             in        unsigned short                 howMany,
             out       AttributesGetResultSet         attributes,
             out       AttributesGetResultIterator    iterator)
             raises    (ApplicationError);

        void destroy(in NameType name)
             raises    (ApplicationError,
                        DeleteError);

    }; // end of ManagedObject_F interface
```

8.2.3 Opérations *objectClassGet*()

L'opération *objectClassGet*() renvoie le nom d'interface visé de l'objet géré nommé. L'objet géré nommé est l'objet géré dont le nom est inclus dans le premier paramètre appelé "name".

Il convient de noter que cette valeur est différente du nom de classe de la façade. Etant donné qu'il est possible pour des objets gérés sur une interface à granularité grossière, de prendre également en charge des interfaces CORBA directes, il a été décidé que cette opération devait renvoyer la même valeur qu'un système gérant obtiendrait s'il invoquait l'opération équivalente directement sur l'objet géré. Si aucune interface d'objet géré à granularité fine n'a été définie pour la classe d'objet géré accessible via l'interface, la réponse doit être le nom détecté de l'interface de la façade sans les caractères "_F". La même valeur retournée en réponse à cette opération sera incluse dans les notifications provenant de l'objet et dans le type de valeur renvoyée pour l'objet. Le type *ObjectClassType,* est une définition de type pour une chaîne.

Une référence étant donnée à une façade, un système gérant peut déterminer de quel type de façade il s'agit via des appels CORBA standards (par exemple l'appel get_interface sur l'interface CORBA::Object). C'est pourquoi, une opération distincte pour cela n'a pas été définie sur l'interface de façade de base.

8.2.4 Opérations *packagesGet*()

L'opération *packagesGet*() renvoie la liste des paquetages conditionnels pris en charge par l'objet géré nommé. Il est possible pour des objets gérés auxquels on a accès via la même façade, de prendre en charge différents paquetages conditionnels. La notion de paquetage conditionnel, comportant chacun un nom de chaîne, est définie dans la Rec. UIT-T X.780. *StringSetType* est une définition de type pour une séquence de chaîne.

8.2.5 Opération *creationSourceGet*()

L'opération *creationSourceGet*() renvoie une valeur indiquant le système qui a provoqué la création de l'objet géré nommé. *SourceIndicatorType* est un type énuméré qui peut prendre trois valeurs: *resourceOperation, managementOperation* et *unknown.* Il indique si l'objet a été créé de manière autonome par la ressource en réaction à une opération ou si l'on ne sait pas pourquoi l'objet a été créé.

8.2.6 Opération *deletePolicyGet*()

L'opération *deletePolicyGet*() renvoie la politique de suppression pour l'objet géré nommé. Il s'agit d'une valeur énumérée qui indique si l'objet n'est pas supprimable ou s'il est supprimable seulement s'il ne contient aucun objet, ou si tous les objets qu'il contient seront supprimés lorsqu'il sera supprimé. (La suppression d'un objet sans suppression des objets qu'il contient n'est pas autorisée.) La politique est fixée lorsque l'objet est créé par son atelier sur la base de l'information de corrélation de nom identifiée dans l'opération de création.

8.2.7 Opération *attributesGet*()

La méthode *attributesGet*() est utilisée pour renvoyer l'ensemble, ou un sous-ensemble de valeurs d'attribut d'objets en une opération. Pour chaque objet géré ou chaque interface de façade dans un modèle d'informations, un type de valeur *valuetype* CORBA contenant les éléments de données pour chaque attribut lisible de cette interface sera défini (les attributs lisibles sont ceux qui sont associés à une opération <attribute name>Get()). Cette méthode peut être utilisée pour extraire ce type de valeur pour tous objets gérés. Les types de valeurs seront définis en suivant la hiérarchie d'héritage des interfaces d'objets gérés (sauf que les types de valeurs ne peuvent pas prendre en charge plusieurs héritages), et chacune sera finalement déduite du *ManagedObjectValueType* défini dans la Rec. UIT-T X.780. L'objet géré doit renvoyer la sous-classe définie pour son interface en réponse à l'utilisation de cette méthode. Ainsi, lorsqu'un client invoque l'opération *attributesGet*() sur un objet géré quelconque, il recevra en retour une référence à *ManagedObjectValueType* qu'il peut ensuite diffuser (de manière restreinte) vers le type de valeur défini pour l'interface sur laquelle l'opération a été invoquée.

Une complication supplémentaire apparaît en ce qui concerne le fait qu'un client peut ne pas vouloir extraire toutes les valeurs d'attribut depuis une instance, et qu'une instance peut ne pas prendre en charge tous les attributs qui se trouvent dans des paquetages conditionnels. (Les types de valeurs incluent des attributs dans des paquetages conditionnels.) Ce cas est traité par l'utilisation du paramètre *attributeNames* in/out. Sur invocation, le client peut soumettre une liste des noms des attributs qui l'intéressent, une liste vide indiquant que tous les attributs pris en charge doivent être renvoyés. Tout nom figurant sur la liste qui n'est pas un nom d'attribut valide doit être ignoré par l'objet géré. Dans sa réponse, l'objet renverra la liste réelle des attributs pour lesquels des valeurs sont fournies. Il faut noter que cette liste peut ne pas correspondre à la liste soumise. L'objet doit toujours renvoyer une liste exacte, même si la liste soumise est vide ou contient des noms non valides. Si tous les noms de la liste soumise ne sont pas valides, l'objet doit renvoyer une liste nulle et un type de valeur vide.

Comme la structure du type de valeur est prédéfinie, l'objet doit donner une certaine valeur aux attributs non demandés ou non pris en charge. A la base, l'objet peut renvoyer les valeurs quelconques pour ces attributs, mais les valeurs doivent être aussi courtes que possible pour des questions d'efficacité. Ainsi, les valeurs nulles doivent être envoyées pour des chaînes, des références et des listes de toute sorte. Une valeur quelconque peut être renvoyée pour des entiers et des types énumérés. Le client doit considérer comme non valides toutes valeurs d'un attribut non nommé dans la liste renvoyée par l'objet.

8.2.8 Opération *attributesBulkGet*()

La méthode *attributesBulkGet*() est utilisée pour retourner plusieurs attributs de plusieurs objets gérés du même type qui sont accessibles via la façade. Le système gérant fournit une liste de noms d'attributs et une liste de noms d'objets gérés à partir desquels on peut extraire ces attributs. La liste des noms d'attributs est traitée de la même façon qu'elle l'est pour l'opération *attributesGet* précédemment décrite. Le traitement de la liste des noms d'objets gérés est décrit ci-après.

8.2.8.1 Détermination des objets à partir desquels des attributs peuvent être extraits

Si la liste de noms est vide, tous les objets accessibles via la façade sont implicitement demandés.

Si la liste inclut un élément qui n'inclut pas la composante finale avec une valeur *ID* "Object" ou <empty>, cet élément de la liste est utilisé comme nom partiel, ce qui implicitement revient à demander l'ensemble des noms gérés (accessibles via la façade) dont les noms commencent par le nom partiel. Un tel nom serait créé en supprimant une ou plusieurs composantes de noms depuis la fin d'un nom d'objets gérés.

Par exemple, on commence avec un élément géré nommé avec un *ID* "me1" et utilisant une valeur *kind* "ME" sous une racine locale définie pour acme telecom:

acme\.com/me1.ME/.façadeID1

Il faut noter que dans la composante de noms finals (après le dernier caractère /), la chaîne *ID* (qui précéde le caractère .) est vide. Par conséquent, il ne s'agit pas d'un nom partiel. Ensuite supposons qu'il y a un objet équipement "eq1", nommé sous me1 en utilisant "EQ" comme type:

acme\.com/me1.ME/eq1.EQ/.façadeID2

Enfin, supposons qu'il y a des objets gérés détenteurs d'équipement appelés sous cet objet d'équipement (utilisant le type "EH"), comme indiqué à la Figure 4 ci-dessous, tous étant accessibles via une façade de détenteurs d'équipement avec l'identificateur ID "façadeID3"

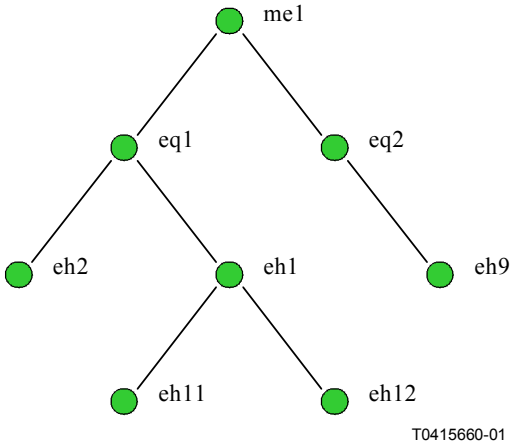

Figure 4/X.780.1 – Exemple d'arbre de nommage

Si le nom partiel "acme\.com/me1.ME/eq1.EQ" est utilisé dans la liste des paramètres des noms d'objets gérés de l'opération *attributesBulkGet* sur la façade, des valeurs d'attributs extraites des objets de maintenance d'équipement suivant doivent être renvoyées:

acme.com/me1.ME/eq1.EQ/eh1.EH/.façadeID3
acme.com/me1.ME/eq1.EQ/eh2.EH/.façadeID3
acme.com/me1.ME/eq1.EQ/eh1.EH/eh11.EH/.façadeID3
acme.com/me1.ME/eq1.EQ/eh1.EH/eh12.EH/.façadeID3

La façade peut reconnaître que le nom est un nom partiel car la valeur de l'ID dans la composante d'un nom final n'est pas "Object" ou <empty>.

Bien que cet exemple ne le démontre pas, l'objet à la racine de l'arborescence identifié par le nom partiel est inclus dans le champ d'application de l'opération.

8.2.8.2 Renvoi des résultats

Les données sont renvoyées dans des types de valeur d'objet gérés fortement typés, une depuis chaque objet géré nommé. Si la façade ne permet pas l'accès pour un nom d'objet géré fourni par le client, aucun type de valeur associée à cet objet n'est renvoyé. Etant donné qu'un système géré peut fournir plusieurs interfaces de façade du même type, le client peut éventuellement devoir invoquer cette opération sur plusieurs interfaces pour extraire les valeurs de tous les objets gérés d'un type donné dans un système.

Même si le client ne demande pas le renvoi des valeurs pour l'attribut "name", la façade doit renvoyer le nom dans chaque type de valeur d'objet géré. Elle doit le faire de sorte que le client connaîtra les valeurs à appliquer à chaque objet géré.

Parallèlement à chaque type de valeur d'objet géré renvoyé, il y a la liste des noms des attributs dans le type de valeur qui a des valeurs valides. Cette liste peut ne pas correspondre à la liste des attributs demandés, étant donné que l'instance peut ne pas prendre en charge tous les attributs demandés. Si l'instance ne prend pas en charge tous les attributs demandés, la façade doit envoyer un type de valeur d'objet géré pour cette instance avec seulement l'attribut de nom contenant une valeur valide.

Etant donné qu'il est possible de renvoyer un grand volume de données, la séquence de conception de l'itérateur est utilisée. Le client spécifie le nombre maximal de types de valeurs à renvoyer. Le reste doit être renvoyé dans un itérateur. Si la présence d'un itérateur est nécessaire, la valeur de renvoi doit être "true". Dans les autres cas, elle doit être "false" et la référence de l'itérateur doit avoir la valeur nulle.

8.2.9 Opération *destroy*()

L'opération finale sur la façade de base, l'opération *destroy*(), est utilisée pour libérer les ressources associées à l'objet géré nommé et à le supprimer. L'exception *DeleteError* est soulevée si la politique de suppression de l'objet est *NotDeletable*. L'exception *DeleteError* est également un moyen extensible de signaler les problèmes rencontrés lors de la destruction d'un objet qui dépend d'un modèle. Ainsi, une tentative de suppression d'un objet point de terminaison d'un chemin avant la suppression du chemin peut se traduire par une erreur *DeleteError*. La Rec. UIT-T Q.816 définit un service appelé le "service terminateur" pour mettre en œuvre la logique nécessaire pour imposer les politiques de suppression et à maintenir l'intégrité de l'arbre de nommage. L'opération de destruction est en réalité destinée à être utilisée par ce service et ne doit pas être directement invoquée par un système gérant. Pour de plus amples détails sur le service terminateur, on se reportera à la Rec. UIT-T Q.816.

(R) FAÇADE-5 – Les interfaces de façade définies pour une interface CORBA à granularité grossière doivent être les héritières (directes ou indirectes) de l'interface *ManagedObject_F* décrite plus haut et définie en langage CORBA IDL dans l'Annexe A. Les capacités décrites ci-dessus doivent être prises en charge.

8.3 Interface d'itérateur *AttributesBulkGet*

Comme indiqué au § 8.2.8, lorsqu'un grand nombre de résultats doivent être renvoyés en réponse à une opération *attributesBulkGet*, il peut être nécessaire d'avoir un itérateur. La structure de conception d'un itérateur est une structure de conception CORBA bien connue. Lorsque de grands volumes de données doivent être renvoyés en réponse à une opération, on renvoie à la place une référence à une interface itérateur. Le client peut alors demander à l'itérateur d'extraire les résultats dans des tronçons gérables.

La description IDL de l'itérateur "attributes get result" est présentée ci-dessous.

```
interface AttributesGetResultIterator {

     boolean getNext(in unsigned short howMany,
          out AttributesGetResultSet results)
          raises (ApplicationError);

     void destroy();

}; // end of interface AttributesGetResultIterator
```

(R) FAÇADE-6 – Le système géré doit instancier un itérateur avec une interface correspondant à la description de la définition de *AttributesGetResultIterator* en langage IDL contenue dans l'Annexe A lorsque le nombre de résultats à envoyer en réponse à une opération *attributesBulkGet* dépasse le nombre demandé par le client.

(R) FAÇADE-7 – Chaque fois qu'un client invoque une opération *getNext* sur l'itérateur, il doit renvoyer l'ensemble subséquent de résultats. L'itérateur doit garder trace du nombre de résultats qui ont déjà été extraits par le client et renvoyer l'ensemble des résultats en une fois. Les résultats initialement renvoyés en réponse à un opérateur *attributesBulkGet* ne doivent pas être renvoyés une nouvelle fois par l'itérateur. L'itérateur doit renvoyer en réponse à une opération *getNext* au plus le nombre de noms indiqué par la valeur du paramètre *howMany*. L'itérateur peut retourner un nombre inférieur à la taille du lot demandé, faisant un compromis entre l'efficacité du renvoi des résultats dans un lot important avec l'éventuelle nécessité de rester bloqué jusqu'à ce qu'un plus grand nombre de résultats soient disponibles. Si le nombre de résultats renvoyés (en plus de ceux qui sont en cours de renvoi) est le plus grand, la valeur renvoyée de l'opération *getNext* doit être "true", dans les autres cas elle est "false". L'itérateur ne doit pas renvoyer un ensemble vide de résultat à moins que le paramètre *howMany* ait été mis à zéro ou qu'il n'y ait plus de résultat à renvoyer, ce qui faisant obligerait le client à interroger l'itérateur.

(R) FAÇADE-8 – Le système géré doit contrôler le cycle de vie de l'itérateur. Une opération destroy, toutefois, peut être invoquée si le gestionnaire veut arrêter l'extraction des résultats avant d'atteindre le dernier résultat. Lorsque l'opération *destroy* est invoquée, l'itérateur doit libérer toutes les ressources qu'il est en train d'utiliser et se supprimer lui-même. Lorsqu'il a retourné le dernier résultat, l'itérateur doit se détruire. L'itérateur peut également être détruit par le système géré s'il est inutilisé pendant une période trop longue.

8.4 Instanciation d'atelier

Les ateliers sont des interfaces d'objets persistantes utilisées pour instancier d'autres objets. L'utilisation d'ateliers suit une structure de conception CORBA bien connue. Les ateliers sont utilisés sur des interfaces à granularité fine pour fournir à un système gérant une méthode de création de nouvelles instances d'objets gérés. Bien qu'elles ne soient pas strictement requises sur des interfaces à granularité grossière car la façade peut jouer le rôle de l'atelier, les interfaces d'atelier distinctes doivent être utilisées sur des interfaces à granularité grossière. Ceci permet de rendre les approches à granularité grossière et à granularité fine plus compatibles. Un autre avantage est d'empêcher l'héritage de l'opération de création pour des objets de superclasse par des sous-classes. Ce problème ne se produit pas avec des ateliers distincts car les interfaces d'atelier ne suivent pas la hiérarchie d'héritage des objets gérés. Les interfaces d'atelier utilisées pour créer des objets gérés sur des interfaces à granularité grossière sont les mêmes que celles définies pour des interfaces à granularité fine. Voir le § 9.

(R) FAÇADE-6 – Un système géré doit fournir au moins une interface d'atelier pour chaque classe d'objets gérés susceptible d'être instanciée à son niveau. Il n'est pas nécessaire de fournir des interfaces pour des classes d'objets qui ne peuvent pas être instanciées dans le système. Les interfaces d'atelier sont inscrites dans le service détecteur d'atelier, défini dans la Rec. UIT-T Q.816. Les ateliers associés aux interfaces à granularité grossière peuvent renvoyer une référence nulle en réponse à une opération create à la place d'une référence à l'objet nouvellement créé. Le nom d'objet géré renvoyé par l'atelier en réponse à une opération create doit indiquer la façade qui peut être utilisée pour accéder au nouvel objet conformément aux règles de nommage d'objets gérés définis dans la Rec. UIT-T Q.816.1.

9 Directives de modélisation CORBA à granularité grossière

Le présent paragraphe spécifie les règles de définition des interfaces IDL à granularité grossière. Une interface à granularité grossière est créée tout d'abord en définissant une interface à granularité fine conformément aux directives contenues dans la Rec. UIT-T X.780. Cette Recommandation traite à la fois de la création des interfaces IDL de A à Z ainsi que la transformation d'une interface GDMO en IDL. Après que l'interface à granularité fine ait été définie, des interfaces de façade pour chaque interface d'objet géré sont ensuite développées conformément aux règles définies au § 10 ci-dessous.

Les spécifications d'interface d'objet géré à granularité fine doivent être maintenues. L'ensemble des autres structures définies pour l'interface à granularité fine, y compris les types de données, les types de valeur, les exceptions, les notifications et les ateliers sont réutilisées sans modification sur l'interface à granularité grossière.

10 Directives de transformation de modèles à granularité fine en modèles à granularité grossière

Un modèle IDL à granularité grossière peut être créé à partir d'un modèle à granularité fine en suivant les étapes ci-dessous:

1) une interface de façade doit être créée pour chaque interface d'objet géré. Une interface d'objet géré est une interface qui découle directement ou indirectement de l'interface *ManagedObject*;

2) les interfaces de façade doivent être créées dans le même module IDL que les interfaces d'objet à granularité fine. Cela dispense la personne chargée de la modélisation d'avoir à inclure des définitions de type pour tous les types définis pour le modèle à granularité fine.

   Les interfaces de façade peuvent être créées dans un fichier distinct ou incluses dans une nouvelle version du fichier à granularité fine. Dans le cas d'un fichier distinct, le fichier contenant le modèle à granularité fine devra être inclus dans la compilation.

   La répartition d'un module sur plusieurs fichiers est autorisée par les normes OMG. A la base, un module IDL définit un espace nom. A l'intérieur d'un module, tous les noms doivent être uniques. Ainsi, un module ne peut pas contenir deux interfaces nommées "ManagedObject_F". Deux modules différents peuvent contenir des noms identiques et les modules peuvent être contenus dans d'autres modules. Lorsqu'un module est réparti sur des fichiers, la règle d'unicité s'applique toujours. La présence de noms dupliqués dans le même module mais dans des fichiers séparés n'est pas autorisée.

   L'IDL figurant dans l'Annexe A est défini à l'intérieur d'un seul module appelé "itut_x780", qui est le même module utilisé dans la Rec. UIT-T X.780. Ainsi, ce module est réparti sur des fichiers. Une des conséquences est qu'aucun des noms utilisés dans l'IDL figurant dans la Rec. UIT-T X.780 ne peut être réutilisé dans l'IDL de la présente Recommandation. Un avantage toutefois est que toute structure IDL définie dans la Rec. UIT-T X.780 peut être réutilisée dans la présente Recommandation en incluant simplement le fichier avec une directive au précompilateur. Aucune définition ou nom détecté n'est requise, comme cela serait le cas si l'IDL occupait des modules distincts. Il faut noter que la conformité est encore basée sur des documents, et non sur des modules IDL. Ainsi un système peut être conforme à la Rec. UIT-T X.780 sans être conforme à la présente Recommandation.

3) Le nom de l'interface de la façade doit être le nom de l'interface à granularité fine à partir de laquelle elle est créée, auquel est adjoint le suffixe "_F" (soulignement suivi de "F" en capitale). Ainsi l'interface de façade créée pour l'objet géré "Equipment" doit être nommé "Equipment_F".

4) Si l'interface d'objet à granularité fine hérite directement de l'interface *ManagedObject,* l'interface de façade créée pour elle est l'héritière de l'interface *ManagedObject_F.* Si l'interface d'objet à granularité fine hérite d'une sous-classe de l'interface *ManagedObject*, l'interface de façade créée pour elle doit hériter de l'interface de façade créée pour cette sous-classe. Ainsi, la hiérarchie d'héritage des interfaces de façade correspond à celle des interfaces d'objets gérés. Par exemple, supposons que l'interface d'objets à granularité fine *EquipmentHolder* hérite de l'interface *Equipment*, l'interface *EquipmentHolder_F* doit alors hériter de l'interface *Equipment_F*. Les interfaces de façade pour toutes les superclasses d'objets à granularité fine doivent être créées avant la création de la façade d'objet à granularité fine.

5) La totalité du contenu de l'interface d'objet à granularité fine doit être copiée dans l'interface de façade avec les modifications suivantes:

- un paramètre *in* du type *NameType* et nommé *name* doit être ajouté au premier paramètre de chaque opération. Ce paramètre doit être utilisé pour être transféré dans le nom de l'objet géré cible sur lequel l'opération doit être invoquée. Si l'opération a déjà un paramètre nommé "name", il doit être renommé;
- tout paramètre ou tout type renvoyé qui utilise une référence IOR d'objet géré doit être traduit en un *NameType*. Une valeur d'IOR d'objet géré sera identifiée comme une référence à une interface *ManagedObject* ou une interface de sous-classe. L'utilisation de tels types de valeurs est découragée pour les interfaces d'objet à granularité fine sachant qu'elles devraient être peu nombreuses;
- dans une description de comportement associée à une spécification d'interface à granularité grossière on peut accepter de faire référence à une description de comportement d'une structure équivalente associée à une spécification d'interface à granularité fine plutôt que de la recopier.

6) Les directives de modélisation à granularité fine interdisent actuellement l'utilisation des attributs IDL OMG pour modéliser des attributs d'objets gérés. En effet, l'IDL OMG ne permet pas de lever des exceptions définies par l'utilisateur sur les opérations d'accès aux attributs. Ainsi à la place des attributs d'objets gérés sont modélisés par des opérations distinctes utilisées pour obtenir ou fixer des valeurs d'attribut, ou ajouter ou supprimer des valeurs à ou depuis un attribut. Cela étant, dans le futur l'OMG permettra probablement des exceptions définies par l'utilisateur sur les opérations d'accès aux attributs. Si tel était le cas et si une interface d'objet à granularité fine utilisait des attributs IDL lors de la création d'une façade pour cet objet, les attributs devraient être traduits en opérations IDL distinctes, et alors le paramètre *name* devrait être ajouté au premier paramètre de ces opérations.

7) Le reste de la séquence IDL d'interface à granularité fine, tels les types de données, les types de valeur, est utilisé pour l'interface à granularité grossière sans modification.

11 Conformité de l'IDL à granularité grossière

Le présent paragraphe définit les critères qui doivent être respectés par d'autres documents normatifs revendiquant la conformité avec les présentes directives ainsi que les fonctions qui doivent être mises en œuvre par les systèmes revendiquant la conformité à la présente Recommandation.

11.1 Conformité des documents normatifs

Toute spécification revendiquant la conformité avec les présentes directives doit:

1) prendre en charge toutes les prescriptions de conformité des documents normatifs spécifiés dans la Rec. UIT-T X.780;

2) suivre les règles de mappage IDL à granularité fine → IDL à granularité grossière définies au § 10.

11.2 Conformité des systèmes

Une réalisation revendiquant la conformité à la présente Recommandation doit:

1) respecter les prescriptions d'instanciation de façade, d'atelier et d'itérateur spécifiées au § 8;

2) mettre en œuvre une interface IDL conforme aux directives de la présente Recommandation. (Voir le § 11.1.)

11.3 Directives relatives aux déclarations de conformité

Les utilisateurs des présentes directives doivent prendre un certain nombre de précautions lors de la rédaction des déclarations de conformité. Etant donné que les modules IDL sont utilisés sous forme d'espaces noms, ils peuvent, comme cela est autorisé par les règles OMG de l'IDL, être répartis sur des fichiers. Ainsi, lorsqu'un module est étendu, son nom ne changerait pas mais un nouveau fichier IDL sera simplement ajouté. Le fait d'indiquer simplement le nom d'un module dans une déclaration de conformité par conséquent ne suffit pas à identifier un ensemble d'interfaces IDL. La déclaration de conformité doit identifier un document et l'année de publication afin de s'assurer que la bonne version de l'IDL est identifiée.

ANNEXE A

Spécifications IDL de la modélisation des interfaces à granularité grossière

```
/* Ce code IDL est destiné à être mémorisé dans un fichier nommé
"itut_x780_1.idl" situé dans le chemin de recherche utilisé par les compilateurs
IDL de votre système. */

#ifndef ITUT_X780_1_IDL
#define ITUT_X780_1_IDL

#include <itut_x780.idl>

#pragma prefix "itu.int"

module itut_x780 {

// TYPES IMPORTÉS
// TYPES DE DONNÉES

     /** Cette structure conserve les résultats d'une extraction d'un ensemble de
     valeurs d'attribut depuis un seul objet géré. Les valeurs d'attribut sont
     placées dans le membre d'attribut fortement typé. Etant donné qu'il est
     possible que la totalité des valeurs des attributs n'ait pas été demandée ou
     prise en charge, les membres de noms d'attribut détiennent la liste des noms
     d'attribut pour lesquels les membres d'attribut ont des valeurs valides. Le
     reste n'est pas valide.  */

     struct AttributesGetResultType {
          ManagedObjectValueType  attributes;
          StringSetType           attributeNames; };

     typedef sequence <AttributesGetResultType> AttributesGetResultSetType;

     interface AttributesGetResultIterator;

// INTERFACE D'ITÉRATEUR ATTRIBUTES GET RESULT

     /** L'interface d'itérateur Attributs Get Result Iterator est utilisée pour
     extraire les résultats d'une opération attributesBulkGet au moyen de la
     structure de conception d'itérateur. */

     interface AttributesGetResultIterator {

          /** Cette méthode sert à extraire les résultats suivants "howMany"
          contenus dans l'ensemble de résultats.
          @param howMany     Nombre maximal d'articles à renvoyer dans les
                             résultats. Le nombre de résultats peut être
                             inférieur lorsque c'est tout ce qui reste ou s'il
```

```
                    faut faire un compromis entre les délais et
                    l'efficacité.
        @param results     Nouveau groupe de résultats.
        @return            Vrai prend la valeur "true" s'il y a encore des
                           résultats après ceux qui sont renvoyés. Si la valeur
                           renvoyée est "true", l'ensemble de résultats ne doit
                           pas être vide, étant donné que cela oblige le client
                           à procéder à des interrogations pour obtenir des
                           résultats. Dans les autres cas, l'appel doit être
                           bloqué.
        */

        boolean getNext(in unsigned short howMany,
             out AttributesGetResultSetType results)
             raises (ApplicationError);

        /** Cette méthode est utilisée pour détruire l'itérateur et libérer
        ses ressources.  En tout état de cause, l'itérateur est automatiquement
        détruit après que les derniers résultats ont été renvoyés et peut être
        détruit s'il n'est pas utilisé pendant une période de temps trop
        longue.
        */

        void destroy();

    }; // fin de l'interface AttributesGetResultIterator

// FAÇADE D'OBJET GÉRÉ

    /** La façade d'objet géré est destinée à être l'interface de base
    de laquelle héritent toutes les autres façades d'objets gérés.
    C'est l'entité centrale pour spécifier les fonctions de base que toutes les
    façades d'objets gérés sont supposées prendre en charge. */

    interface ManagedObject_F {

        /** Cette méthode renvoie le nom détecté de la classe la plus
        spécifique de l'objet géré (par exemple "itut_x780::EquipmentR1").
        *NOTE*  Cette opération renvoie le nom de classe de l'objet, et
        non la façade. Il s'agit du nom de l'interface de façade sans
        le suffixe "_F". Il s'agit du même nom qui est contenu dans le
        paramètre objectClass des notifications, même sur les interfaces
        prenant en charge les façades.
        @param name    Nom de l'instance d'objet géré sur laquelle l'opération
                       doit être invoquée.
        @return        Nom d'interface de l'objet géré.
        */

        ObjectClassType objectClassGet(in NameType name)
             raises (ApplicationError);


        /** Cette méthode renvoie une liste de tous les paquetages
        conditionnels pris en charge par cette instance.
        @param name    Nom de l'instance d'objet géré sur laquelle l'opération
                       doit être invoquée.
        @return        Liste des noms de paquetage pris en charge par l'objet
                       géré

        StringSetType packagesGet (in NameType name)
             raises (ApplicationError);
```

```
/** Cette méthode renvoie une indication sur la façon dont l'objet
a été créé.
@param name    Nom de l'instance d'objet géré sur laquelle
               l'opération doit être invoquée.
@return        Indique si l'objet géré nommé a été créé de
               manière autonome ou par un système gérant.
*/

SourceIndicatorType creationSourceGet(in NameType name)
     raises (ApplicationError);


/** Cette méthode renvoie à une valeur indiquant si l'objet peut
être supprimé et si tel est le cas, si tous les objets contenus
sont automatiquement supprimés.
@param name    Nom de l'instance d'objet géré sur laquelle
               l'opération doit être invoquée.
@return        Politique de suppression de l'objet nommé géré
*/

DeletePolicyType deletePolicyGet (in NameType name)
     raises (ApplicationError);


/** Cette méthode peut être utilisée pour obtenir de manière
générique tous les attributs pris en charge par une instance.
Chaque interface est supposée sous-classer le type de valeur
d'objet géré et ajouter d'autres attributs pris en charge par
cette interface.  L'objet géré doit renvoyer un objet de valeur
de ce type.  Le client doit alors "réduire" la référence pour
accéder à tous les attributs. <p>

Le client peut également soumettre une liste de noms indiquant
les attributs qu'il souhaite recevoir.  Ces noms doivent
correspondre aux noms de membres contenus dans l'objet valeur.
Pour les membres qui ne figurent pas sur la liste, et pour les
membres qui font partie de paquetage mais qui ne sont pas pris en
charge, le serveur peut renvoyer une valeur quelconque mais celle-ci
doit être aussi courte que possible.  Le serveur peut également
renvoyer la liste des attributs, laquelle peut être plus courte en
raison de l'exclusion d'attributs dans les paquetages non pris en
charge. Le client doit considérer la valeur de tout membre qui ne se
trouve pas dans la liste renvoyée comme étant un "déchet". <p>

Une liste de noms d'attribut nulle indique que tous les attributs
pris en charge ne doivent pas être renvoyés. Le serveur doit
renvoyer la liste réelle.
@param name             Nom de l'instance d'objet géré sur laquelle
                        l'opération doit être invoquée.
@param attributeNames   Liste des noms d'attributs à extraire.
@return                 Valeur type contenant des attributs.
*/

ManagedObjectValueType attributesGet (
     in        NameType name,
     inout     StringSetType attributeNames)
     raises    (ApplicationError);


/** Cette méthode est utilisée pour envoyer plusieurs attributs
à partir de plusieurs objets gérés du même type. Le client fournit
une liste de noms d'attributs et une liste de noms d'objets gérés à
partir de laquelle il faut extraire les attributs.  <p>
```

```
Les données sont renvoyées dans des types de valeurs d'objets gérés
fortement typés, à raison d'un pour chaque objet géré nommé source.
Si la façade ne permet pas l'accès à un nom d'objet géré fourni par
le client, aucun type de valeur pour cet objet n'est renvoyé.  Etant
donné qu'un système géré peut fournir plusieurs interfaces de façade
du même type, le client peut devoir invoquer cette opération sur
plusieurs interfaces pour extraire les valeurs à partir de tous les
objets gérés d'un type donné dans un système.  <p>

Même si le client ne demande pas à ce que soient renvoyées les valeurs
de l'attribut "name", la façade doit renvoyer le nom dans chaque type
de valeur d'objet géré. S'il ne le fait pas, le client ne saura
pas à quelle instance d'objet géré ces valeurs s'appliquent. <p>

Avec chaque type de valeur d'objet géré renvoyé est adjointe une
liste des noms des attributs associée à ce type de valeur qui ont
des valeurs valides. Cette liste peut ne pas correspondre à
la liste des attributs demandée étant donné qu'il se peut que
l'instance ne prenne pas en charge tous les attributs demandés. La
valeur "name" doit toujours se trouver sur la liste renvoyée.
Si l'instance ne prend en charge aucun des attributs demandés, la
façade doit renvoyer le type de valeur d'objet géré pour
l'intance considérée avec seulement l'attribut de nom
contenant une valeur valide.  <p>

Etant donné qu'un grand volume de données peut être renvoyé, la
structure de conception de l'itérateur est utilisée. Le client spécifie
le nombre maximal de types de valeurs à renvoyer. Le reste doit être
renvoyé dans un itérateur. Si on utilise un itérateur, la valeur
renvoyée doit être "true". Dans les autres cas, elle doit être "false"
et la référence de l'itérateur doit être nulle
@param names            Noms des objets gérés à partir desquels doivent
                        être extraites les valeurs d'attribut.
@param attributeNames   Noms des attributs à extraire,
@param howMany          Nombre maximal de types de valeur à renvoyer
                        dans le paramètre d'attribut.
@param attributes       Premier lot de résultats.
@param iterator         Référence à un itérateur si nécessaire. Dans
                        les autres cas, cette référence est nulle.
@return                 Valeur "true" si un itérateur est renvoyé, dans
                        les autres cas elle prend la valeur "false".
*/

boolean attributesBulkGet (
     in       NameSetType                    names,
     in       StringSetType                  attributeNames,
     in       unsigned short                 howMany,
     out      AttributesGetResultSetType     attributes,
     out      AttributesGetResultIterator    iterator)
     raises   (ApplicationError);

/** Cette méthode détruit l'objet. Elle est utilisée pour libérer
simplement toutes les ressources associées à l'objet géré. Elle ne
procède à aucune vérification pour les objets contenus et ne supprime
pas les liens de noms de l'arbre de nommage. <p>

L'objet de cette opération est de permettre la prise en charge de
services de destruction de l'objet géré.  <p><b>

NOTE – L'invocation directe de cette opération depuis un système
gérant peut fausser l'arbre de nommage et il est recommandé
seulement de ne l'invoquer que dans des circonstances extraordinaires.
Les clients qui souhaitent supprimer un objet doivent à la place
```

```
          de cette opération utiliser le service terminateur. </b>
          @param name    Nom de l'instance d'objet géré sur laquelle
                         l'opération doit être invoquée.
          */

          void destroy(in NameType name)
               raises    (ApplicationError,
                         DeleteError);

     }; // fin de l'interface ManagedObject_F
```

// Module ApplicationErrorConst

```
/** Ce module contient les constantes définies pour le code d'erreur contenu dans
les structures Application Error Info renvoyées avec les exceptions Error
Application.
*/

module ApplicationErrorConst {

     /** Ce code d'exception d'erreur d'application indique qu'un objet
     cible d'une opération n'a pu être trouvé. */

     const short objectNotFound = 4;

}; // fin du module ApplicationErrorConst

}; // fin du module itut_x780

#endif  // fin de #ifndef ITUT_X780_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, 2002