Union internationale des télécommunications

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

X.782

(05/2012)

SÉRIE X: RÉSEAUX DE DONNÉES, COMMUNICATION ENTRE SYSTÈMES OUVERTS ET SÉCURITÉ

Gestion OSI – Fonctions de gestion et fonctions ODMA

Lignes directrices relatives à la définition de services web pour les objets gérés et les interfaces de gestion

Recommandation UIT-T X.782

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX DE DONNÉES, COMMUNICATION ENTRE SYSTÈMES OUVERTS ET SÉCURITÉ

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.379
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.889
Applications génériques de l'ASN.1	X.890–X.899
TRAITEMENT RÉPARTI OUVERT	X.900–X.999
SÉCURITÉ DE L'INFORMATION ET DES RÉSEAUX	X.1000–X.1099
APPLICATIONS ET SERVICES SÉCURISÉS (1)	X.1100–X.1199
SÉCURITÉ DU CYBERESPACE	X.1200–X.1299
APPLICATIONS ET SERVICES SÉCURISÉS (2)	X.1300–X.1499

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

Recommandation UIT-T X.782

Lignes directrices relatives à la définition de services web pour les objets gérés et les interfaces de gestion

Résumé

La Recommandation UIT-T X.782 définit un ensemble de lignes directrices concernant la modélisation d'objets gérés ainsi qu'une interface de gestion pour la gestion de réseau fondée sur les services web. Lue conjointement avec la Recommandation UIT-T Q.818, elle sert de cadre pour les interfaces de gestion de réseau fondée sur les services web. Elle spécifie de quelle manière les interfaces de services web orientées service devraient être définies. Elle comprend les scénarios d'application appropriés des services web dans le cadre des interfaces de gestion de réseau, les méthodes génériques d'accès aux objets gérés fondés sur XML et la modélisation d'informations dans le langage WSDL et dans le schéma XML. Elle fournit des définitions WSDL et des schémas XML dans le but de définir certains types de données fondamentaux: les objets gérés génériques et les méthodes génériques d'accès aux objets gérés. La présente Recommandation et la Recommandation UIT-T Q.818 constituent à elles deux un cadre pour les interfaces de gestion de réseau fondé sur les services web et la grande diversité d'applications s'y rapportant.

Historique

Edition	Recommandation	Approbation	Commission d'études	ID unique*
1.0	ITU-T X.782	2012-05-14	2	11.1002/1000/11625
1.1	ITU-T X.782 (2012) Cor. 1	2013-03-16	2	11.1002/1000/11890

Mots clés

Traitement réparti, langage de balisage extensible (XML), objets gérés, interfaces de gestion de réseau, services web, langage de description des services web (WSDL), schéma XML.

* Pour accéder à la Recommandation, reporter cet URL http://handle.itu.int/ dans votre navigateur Web, suivi de l'identifiant unique, par exemple http://handle.itu.int/11.1002/1000/11830-en.

AVANT-PROPOS

L'Union internationale des télécommunications (UIT) est une institution spécialisée des Nations Unies dans le domaine des télécommunications et des technologies de l'information et de la communication (ICT). Le Secteur de la normalisation des télécommunications (UIT-T) 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 développeurs de consulter la base de données des brevets du TSB sous http://www.itu.int/ITU-T/ipr/.

TABLE DES MATIÈRES

Recommandation UIT-T X.782

Lignes directrices relatives à la définition de services web pour les objets gérés et les interfaces de gestion

1 Domaine d'application

L'architecture de gestion de réseau définie dans la Recommandation [UIT-T M.3010] permet d'utiliser plusieurs protocoles de gestion. Jusqu'à présent, au niveau de la couche application, on pouvait utiliser la paire GDMO/CMIP et les protocoles GIOP/IIOP de l'architecture CORBA. Grâce à la méthode de spécification des interfaces de gestion définie dans la Recommandation [UIT-T M.3020], il est possible de mettre en place de nouvelles technologies dans les interfaces de gestion de réseau et on peut désormais recourir aux services web/langage XML pour la gestion de réseau.

La présente Recommandation et la Recommandation [UIT-T Q.818] visent à déterminer un cadre permettant de définir la façon dont il convient de modéliser les interfaces prises en charge par les systèmes de gestion et par les éléments de réseau, au moyen des services web/schéma XML. La présente Recommandation a pour objet de fournir les lignes directrices et les instructions suivantes:

– modalités de conception des interfaces fondées sur le langage WSDL et orientées service;
– scénarios d'application appropriés et inappropriés concernant les services web dans le cadre des interfaces de gestion de réseau;
– méthodes génériques d'accès aux objets gérés;
– héritage des objets gérés et des interfaces;
– lignes directrices relatives à la modélisation d'informations concernant les interfaces fondées sur les services web;
– conventions de style concernant les spécifications dans le langage WSDL et dans le schéma XML.

2 Références

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.

[UIT-T E.164] Recommandation UIT-T E.164 (2010), *Plan de numérotage des télécommunications publiques internationales*.

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

[UIT-T M.3020] Recommandation UIT-T M.3020 (2011), *Méthodologie pour la spécification des interfaces de gestion*.

[UIT-T M.3701] Recommandation UIT-T M.3701 (2010), S*ervices communs de gestion – Gestion des états – Spécifications et analyse – Indépendance vis-à-vis du protocole*.

[UIT-T Q.818] Recommandation UIT-T Q.818 (2012), *Services de gestion fondés sur des services web.*

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

[UIT-T X.703] Recommandation UIT-T X.703 (1997), *Technologies de l'information – Architecture de gestion répartie ouverte*.

[ATIS-I-000002] ATIS Specification ATIS-I-000002 (2011), *ATIS XML Schema Development Guidelines*.

[OASIS WSN] OASIS Specification (2006), *Web Services Base Notification v1.3*.

[OASIS UDDI] OASIS Specification (2004), *Universal Description, Discovery and Integration (UDDI) v3.0.2*.

[W3C Primer] W3C Recommendation (2004), *XML Schema Part 0: Primer Second Edition.*

[W3C SOAP] W3C Recommendation (2007), *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).*

[W3C WSDL] W3C Recommendation (2001), *Web Services Description Language (WSDL) 1.1*.

[W3C XML] W3C Recommendation (2000), *Extensible Markup Language (XML) 1.0 Second Edition*.

[W3C XS-P1] W3C Recommendation (2004), *XML Schema Part 1: Structures Second Edition*.

[W3C XS-P2] W3C Recommendation (2004), *XML Schema Part 2: Datatypes Second Edition*.

3 Définitions

3.1 Termes définis ailleurs

La présente Recommandation utilise les termes suivants définis ailleurs:

3.1.1 agent [UIT-T M.3020]

3.1.2 classe d'objets gérés [UIT-T X.701]

3.1.3 gestionnaire [UIT-T M.3020]

3.1.4 notification [UIT-T X.703]

3.2 Termes définis dans la présente Recommandation

La présente Recommandation ne définit aucun nouveau terme.

4 Abréviations et acronymes

La présente Recommandation utilise les abréviations et acronymes suivants:

B2B interentreprises (*business to business*)

BPEL langage d'exécution des processus d'entreprise (*business process execution language*)

C2B entre l'entreprise et le client (*customer to business*)

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

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

DCOM modèle d'objets composants répartis (*distribute component object model*)

DN nom distinctif (*distinguished name*)

DTD définition de type de document (*document type definition*)

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

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

GIOP protocole général entre courtiers ORB (*general inter-ORB protocol*)

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

IIOP protocole Internet entre courtiers ORB (*internet inter-ORB protocol*)

IT technologies de l'information (*information technology*)

LAN réseau local (*local area network*)

MO objet géré (*managed object*)

MOC classe d'objets gérés (*managed object class*)

MOO opérations sur objets multiples (*multiple object operation*)

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

OOAD analyse et conception orientées objet (*object-oriented analysis and design*)

OS système d'exploitation (*operating system*)

RDN nom distinctif relatif (*relative distinguished name*)

RGT réseau de gestion des télécommunications

SOA architecture orientée service (*service-oriented architecture*)

SOAP protocole simple d'accès aux objets (*simple object access protocol*)

UDDI découverte, description et intégration universelles (*universal description discovery and integration*)

WS services web (*Web services*)

WSDL langage de description des services web (*Web services description language*)

WSN notification des services web (*Web services notification*)

XML langage de balisage extensible (*extensible markup language*)

XSD définition de schéma XML (*XML schema definition*)

5 Conventions

Un petit nombre de conventions sont appliquées dans la présente Recommandation afin d'informer le lecteur de l'intention du texte. Bien que la plus grande partie de la Recommandation soit normative, les alinéas qui décrivent succinctement les exigences obligatoires qu'un système de gestion (gérant ou géré) doit respecter sont précédés de la lettre "R" en caractère gras entre parenthèses, suivie d'un nom court indiquant l'objet de l'exigence et d'un numéro. Par exemple:

(R) EXEMPLE-1 Exemple d'exigence obligatoire.

Les exigences dont l'application par un système de gestion est facultative sont précédées de la lettre "O" au lieu de "R". Par exemple:

(O) EXEMPLE-2 Exemple d'exigence facultative.

Les déclarations d'exigences servent à créer des profils d'observance et de conformité.

On trouvera dans la présente Recommandation des exemples de spécifications en WSDL et en XML; l'Annexe A contient quant à elle les spécifications normatives en WSDL et en XML concernant les types de données, les classes de base et d'autres structures de modélisation orientées service relevant

du présent cadre. Les spécifications en WSDL et en XML sont écrites en caractères Courier New de taille 10:

```
<!-- Example XML schema -->
<xsd:complexType name="AType">
   <xsd:sequence>
      <xsd:element name="a" type="xsd:string"/>
      <xsd:element name="b" type="xsd:long"/>
   </xsd:sequence>
</xsd:complexType>
```

6 Aperçu général du cadre de gestion fondée sur les services web

Les services web sont beaucoup utilisés dans le secteur informatique. L'Appendice I donne davantage de renseignements sur les caractéristiques de la technologie des services web. Il est possible d'utiliser des services web semblables à la technologie CORBA dans les interfaces de gestion de réseau.

La présente Recommandation et la Recommandation [UIT-T Q.818] déterminent un cadre permettant de définir la façon dont il convient de modéliser les interfaces prises en charge par les systèmes de gestion et par les éléments de réseau, au moyen du langage WSDL et du schéma XML.

Le cadre de gestion fondée sur les services web comprend les aspects suivants.

1) Des lignes directrices relatives à la définition des objets gérés et des interfaces:
   - principes concernant la conception des interfaces fondées sur le langage WSDL et orientées service;
   - définition des objets gérés utilisant le schéma XML;
   - méthodes d'accès aux objets gérés;
   - héritage des objets gérés et des opérations d'interface;
   - lignes directrices relatives à la modélisation d'informations concernant les opérations des interfaces fondées sur les services web;
   - idiomes de style pour les spécifications en langage WSDL et dans le schéma XML.
2) Des services web prenant en charge les services de gestion de réseau:
   - définition des services de notification utilisant la norme [OASIS WSN] relative à la notification de base des services web;
   - utilisation de l'enregistrement des services UDDI défini dans la norme [OASIS UDDI];
   - définition d'un service de pulsation;
   - définition d'un service d'opérations sur objets multiples;
   - définition d'un service de contenance.

La présente Recommandation traite principalement des lignes directrices relatives à la définition des objets gérés et des interfaces et la Recommandation [UIT-T Q.818] traite principalement des services web prenant en charge les services de gestion de réseau. Prises conjointement, ces deux Recommandations forment un cadre de gestion fondée sur les services web.

7 Principes relatifs à la conception d'interfaces fondées sur le langage WSDL et orientées service

Dans cette partie, il s'agit de mettre en lumière certaines considérations relatives à la conception des interfaces dont le présent cadre de gestion doit tenir compte grâce aux interfaces orientées service. On y trouvera les principes de modélisation concernant les objets gérés orientés service et leurs méthodes d'accès.

Les considérations relatives à la conception orientée service et liées au répertoire et à la modélisation WDSL et XSD concernent les superclasses, le nommage des objets gérés ainsi que les interfaces orientées service et les opérations et les notifications associées.

Un service web est une technologie orientée service contrairement aux modèles de gestion classiques CORBA/IDL et GDMO/CMIP. L'analyse et la conception d'interfaces orientées objet (OOAD) sont axées sur le niveau de classe, c'est-à-dire qu'elles encapsulent le comportement et des données connexes dans un même objet, qui présente au moins une interface permettant d'accéder aux états et aux attributs encapsulés. La conception d'interfaces orientées service permet de séparer les données et l'état encapsulés du comportement, d'où un couplage faible à un niveau plus élevé. Avec cette méthode, certains objets (partagés) ne maitrisent plus leur propre comportement, lequel est déterminé par quelques opérations d'interface définies à l'avance.

La présente Recommandation et la Recommandation [UIT-T Q.818] visent à définir l'utilisation simple et générale qu'il est possible de faire des modèles de conception des interfaces orientées service. Les fonctions de gestion et de contrôle sont définies à l'aide d'une approche avec granularité de service, c'est-à-dire que les opérations d'interface s'organisent au sein même d'un service donné (ensembles des fonctions de gestion telles que la gestion de la configuration, la gestion de la qualité de fonctionnement, etc.) et non dans le cadre d'une classe d'objets gérés spécifique. Cette orientation vers les services requiert la souplesse offerte par la granularité d'accès propre à chaque application, où des interfaces WSDL prédéfinies permettent d'accéder à des ensembles bien définis de types d'entité du RGT.

Le présent cadre de gestion comprend les principes ci-après concernant la définition d'un modèle d'informations et d'interfaces de gestion fondés sur le langage WSDL et la définition XSD et orientés service.

– Toutes les interactions au niveau de l'interface sont définies comme étant des opérations WSDL et chaque opération comprend une demande et éventuellement la réponse correspondante, si elle est nécessaire.
– Chaque classe MOC est définie comme étant un complexType XML lorsqu'elle est échangée à travers l'interface de gestion, et chaque attribut ou état de la classe MOC est défini comme étant un élément du complexType.
– Le nommage des instances de classe MOC suit le concept DN, mais il s'agit d'une chaîne contenant l'intégralité de la liste des noms RDN.
– Le présent cadre de gestion définit ce qu'est un service générique. Il comporte cinq méthodes génériques d'accès aux objets: createMO, deleteMO, getMOAttribute, setMOAttribute et getPackages. Par ailleurs, les méthodes génériques d'accès aux objets utilisent des paires "name-value" pour exprimer les propriétés et les valeurs des différents types d'instance de classe MOC.
– Les autres fonctions de contrôle de l'interface sont définies comme étant des opérations d'interface WSDL, lesquelles sont organisées en service avec la granularité des ensembles de fonctions de gestion.
– Les types de données communs sont définis comme étant des schémas XML susceptibles d'être partagés par les définitions d'interface propres aux applications.
– Les notifications que l'agent envoie au gestionnaire devraient respecter le format et le comportement définis dans la norme [OASIS WSN]. Le contrôle des services de gestion des notifications est défini plus en détail dans la Recommandation [UIT-T Q.818].

8 Définition d'un objet géré générique à l'aide du schéma XML

8.1 Rôle des services web dans les interfaces de gestion

Afin de prendre en charge les objets logiciels représentant des ressources gérables, une classe de base est définie aux fins de la modélisation des ressources de réseau. Les autres classes MOC des modèles d'information doivent être dérivées de la classe de base, afin de pouvoir être utilisées dans le présent cadre de gestion. Des méthodes d'accès génériques et d'autres services étendus sont définis pour fournir des interfaces servant à gérer des objets gérés.

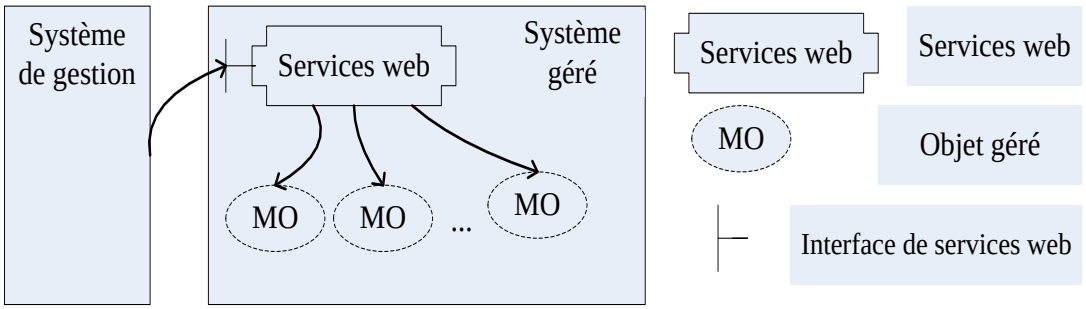

Figure 1 – Rôle des services web

La Figure 1 montre de quelle manière un système de gestion accède à un système géré prenant en charge une interface de services web. Une interface de services web se comporte comme une entité intermédiaire permettant au système de gestion de gérer les bons objets gérés du système géré représentant des ressources gérables.

8.2 Définition des objets gérés à l'aide du schéma XML

Dans une perspective de gestion OSI, un objet géré correspond à une ressource qu'il est prévu de gérer, par exemple une connexion ou un matériel physique. Il s'agit donc d'une représentation abstraite d'une ressource de ce type, qui permet de représenter ses propriétés à des fins de gestion. Un objet géré peut avoir des attributs qui donnent des renseignements utilisés pour caractériser l'objet lui-même ou les opérations associées à ses comportements. Le présent cadre de gestion vise à fournir un ensemble de capacités permettant de gérer ces objets, pour lesquels il est nécessaire de disposer de méthodes permettant de décrire leurs propriétés et leurs comportements. Un objet géré orienté service est une entité gérée représentant une ressource de service gérable pour ce qui est de l'état et du comportement partagés, l'état et le comportement étant dissociés grâce au recours à ce qu'il est convenu d'appeler une "entité de gestion" externe en ce qui concerne le comportement (par exemple un service et son interface), laquelle joue un rôle directeur concernant les comportements des entités gérées qui lui sont confiées. Puisqu'il est possible de dissocier le comportement et l'état d'un objet géré, l'état peut être décrit au moyen du schéma XML et le comportement au moyen des interfaces de services web en WSDL. Le stockage de l'état d'un objet géré dans un document XML présente un avantage important, à savoir que les services web utilisent le schéma XML pour décrire le type de données figurant dans les messages échangés et que ces informations relatives aux objets gérés fondés sur XML sont échangeables sans modification aucune.

8.2.1 Définition d'une classe d'objets gérés générique

Une classe d'objets gérés est elle-même une représentation abstraite d'objets gérés. Toutes les ressources de réseau ont en commun des attributs et toutes les classes MOC héritent, directement ou indirectement, d'une superclasse, à savoir la classe ManagedObject. Définir de nouvelles classes MOC en utilisant ManagedObject sera plus rapide et plus simple, et permettra d'améliorer la maintenance. Comme indiqué précédemment, toutes les classes MOC sont décrites avec le schéma XML, et le type de données de ManagedObject est indiqué dans le Tableau 1 et les attributs dans le Tableau 2.

Tableau 1 – Type de données de la superclasse ManagedObject

```
<xsd:complexType name="ManagedObject_C">
  <xsd:sequence>
    <xsd:element name="objectClass" type="xsd:string"/>
    <xsd:element name="objectInstance" type="x782:NameType"/>
    <xsd:element name="packages" type="x782:PackageListType"/>
    <xsd:element name="creationSource" type="x782:SourceIndicatorType"/>
  </xsd:sequence>
</xsd:complexType>
```

Tableau 2 – Attributs de la superclasse ManagedObject_C

Nom de l'attribut	Qualificatif de prise en charge	Qualificatif de lecture	Qualificatif d'écriture
objectClass	Obligatoire	Obligatoire	–
objectInstance	Obligatoire	Obligatoire	–
packages	Facultatif	Obligatoire	–
creationSource	Facultatif	Obligatoire	–

Comme le montre le Tableau 2, ManagedObject se compose de quatre attributs: objectClass, objectInstance, packages et creationSource. A chaque attribut correspond une valeur, dont la structure peut être simple ou complexe. Dans le cas présent, l'attribut de l'objet géré est différent de l'attribut de la spécification dans le schéma XML, et on peut tout simplement le faire correspondre à l'élément du schéma XML. L'attribut objectClass est utilisé pour identifier le type de classe de cette instance d'objet géré. L'attribut objectInstance est utilisé pour identifier de manière univoque une instance d'objet géré, et le type de données utilise NameType (qui a la même sémantique que le nom distinctif), comme spécifié en (1). L'attribut packages est un ensemble de chaînes permettant d'indiquer les capacités que l'objet géré prend en charge. L'attribut creationSource indique si un objet géré est créé automatiquement dans un système géré, ou par un système de gestion via une opération de gestion, ou si la source de sa création reste inconnue.

```
DN(list of xsd:string) ::= "<attriubte_name_1>=<attibute_value_1>",
            "<attribute_name_2>=<attribute_value_2>"
             …,
            " <attribute_name_n>=<attribute_value_n>"
```
(1)

Les attributs de la formule qui précède devraient donner leur nom aux attributs des classes MOC.

On trouvera dans la partie A.1 de l'Annexe A une définition complète utilisant le schéma XML concernant la classe générique ManagedObject_C.

(R) OBJET-1. Toutes les classes utilisées pour modéliser les ressources dans un système géré héritent, directement ou indirectement, de la classe *ManagedObject_C* décrite ci-dessus et définie à l'aide du schéma XML dans la partie A.1 de l'Annexe 1. Les capacités décrites ci-dessus doivent être prises en charge.

8.2.2 Héritage des objets gérés

Une classe MOC est définie comme étant une spécialisation d'une autre classe MOC sur la base du principe de l'héritage, comme pour tous les autres objets gérés qui héritent, directement ou indirectement, de ManagedObject. La spécialisation d'une classe MOC suppose que tous les attributs définis pour la superclasse seront pris en charge par la sous-classe. Les attributs des objets gérés étant décrits à l'aide du schéma XML, l'héritage peut être assuré par extension des types de données. Par exemple, supposons que la classe MOC equipment hérite directement de la classe de base ManagedObject, alors la classe equipment peut hériter de tous les attributs de ManagedObject, par extension, et déclarer d'autres attributs qui lui sont propres, tels que userLabel, comme le montre le Tableau 3.

Tableau 3 – Type de données de la classe Equipment_C obtenu par extension

```
<xsd:complexType name="Equipment_C">
 <xsd:complexContent>
   <xsd:extension base="x782:ManagedObject_C">
     <xsd:sequence>
       <xsd:element name="userLabel" type="xsd:string"/>
       …
     </xsd:sequence>
   </ xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

Certains objets gérés peuvent nécessiter un héritage multiple, mais le schéma XML ne permet qu'un héritage simple pour les types de données, et il n'est pas recommandé d'utiliser un héritage multiple. Si la sémantique d'un héritage multiple est requise, la classe MOC dérivée doit hériter uniquement d'une des classes MOC supérieures et les attributs des autres classes supérieures doivent être ajoutés manuellement.

8.2.3 Fonctionnalité de paquetage

On peut avoir recours aux paquetages pour regrouper des capacités (par exemple des attributs connexes) ou pour fournir des capacités de prise en charge conditionnelle. On peut définir un paquetage comme étant un complexType XSD, avec un suffixe "_P". Lorsqu'il est utilisé, cet élément peut avoir un type de complexType représentant ce paquetage, avec minOccurs="0" maxOccurs="1" comme qualificatifs, ce qui indique que le paquetage en question peut être conditionnel ou facultatif. Le Tableau 4 donne un exemple de définition et d'utilisation d'un paquetage.

Tableau 4 – Type de données de la classe equipment obtenu par extension

```
<!-- definition of a Package -->
<xsd:complexType name="StatePackage_P">
    <xsd:sequence>
       <xsd:element name="adminstrativeState"
type="x782:AdminstrativeStateType"/>
       <xsd:element name="operationalState" type="x782:OperationalStateype"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Equipment_C">
 <xsd:complexContent>
  <xsd:extension base="x782:ManagedObject_C">
    <xsd:sequence>
      <xsd:element name="equipmentId" type="xsd:string"/>
      <xsd:element name="userLabel" type="xsd:string"/>
      …
      <!-- usage of the Package -->
    <xsd:element  name="statePackage"  type="x782:StatePackage_P"  minOccurs="0"
maxOccurs="1" /></xsd:sequence>
   </ xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

8.2.4 Attributs et types de données courants

Le tableau qui suit expose certains des attributs et des types de données courants qui peuvent être partagés grâce au présent cadre de gestion.

Tableau 5 – Attributs et types de données habituels

Nom de l'attribut	Type de données	Description
administrativeState	AdministrativeStateType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
availabilityStatus	AvailabilityStatusSetType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
backedUpStatus	BackedUpStatusType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
controlStatus	ControlStatusSetType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
creationSource (Note)	SourceIndicatorType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
externalTime	ExternalTimeType	
objectClass (Note)	ObjectClassType	Désigne une classe MOC
objectInstance (Note)	NameType	Désigne une instance d'objet géré
operationalState	OperationalStateType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
packages (Note)	StringSetType	Désigne les paquetages qu'une instance d'objet géré prend en charge.
proceduralStatus	ProceduralStatusSetType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
standbyStatus	StandbyStatusType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
systemLabel	SystemLabelType	Désigne une étiquette pour un système.
unknownStatus	UnknownStatusType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
usageState	UsageStateType	Voir la Recommandation [UIT-T M.3701] pour de plus amples détails
NOTE – Tous les objets gérés héritent de ces attributs.		

On trouvera dans la partie A.1 de l'Annexe A les définitions XSD détaillées pour les types de données qui précèdent.

9 Méthodes d'accès aux objets gérés

Ce paragraphe décrit le cadre de gestion de réseau fondée sur les services web, lequel doit fournir un ensemble de méthodes permettant de contrôler les ressources de réseau. Ces méthodes fournissent des capacités fondamentales permettant de gérer des objets gérés et, par conséquent, elles sont appelées méthodes d'accès génériques (voir le Tableau 6). La Figure 1 donne la procédure d'accès et le cadre de gestion utilise la technologie des services web pour échanger les informations relatives aux objets gérés. Les services web permettent de séparer les états et les comportements des objets gérés et exposent leurs comportements par l'intermédiaire d'une interface de services web. Un service web étant une technologie orientée service, dans le présent cadre de gestion, tous les objets gérés sont conçus de manière à ce qu'il soit possible d'y accéder au moyen d'une interface unique; celle-ci doit savoir quel objet constitue la véritable cible d'une opération et l'identificateur unique de l'objet géré cible devrait être fourni dans chaque demande d'accès. Dans la droite ligne de ces exigences, on trouvera dans le Tableau 6 quelques méthodes d'accès génériques indispensables:

Tableau 6 – Méthodes d'accès génériques

Nom de l'opération	Paramètre d'entrée	Paramètre de sortie
getMOAtrributes	– objectInstance : Name – attributeNameList : SEQUENCE OF String	– attributeNameAndValueList : SEQUENCE OF {attributeName, attributeType, attributeValue} – status : ENUMERATION
setMOAttributes	– objectInstance : DN – attributeNVMList : SEQUENCE OF {attributeName, attributeType, attributeValue, modifyOption}	– status
createMO	– objectclass – objectClassInstance – attributeNameAndValueList	– status
deleteMO	– objectInstance	– status
getPackages	– objectInstance	– packages : List of string – status

Où,

1) getMOAttributes – sert à récupérer tout ou partie des valeurs d'attribut d'un objet géré en une opération. Cette opération utilise le nom distinctif comme le premier paramètre permettant d'identifier de façon univoque l'objet géré et la liste des noms d'attribut à interroger. Le résultat qui en découle comprend les valeurs d'attribut et le statut de l'opération. La liste attributeNameAndValueList est une liste de triplets comprenant les éléments attributeName, attributeType et attributeValue. L'élément attributeType désigne le type original correspondant à attributeValue et les valeurs d'attribut sont renvoyées via l'élément "any" du schéma XML en ce qui concerne les valeurs de type arbitraire. Le paramètre status indique si l'opération réussit ou échoue. Etant donné que "any" est défini pour le type de données de la valeur d'attribut renvoyée, lorsqu'il reçoit du client une demande semblable, le serveur renverra les attributs demandés dans le paramètre de sortie attributeNameAndValueList, où le champ attributeValue sera codé sous la forme d'un texte XML, à partir d'un élément variable, et l'application client pourra le décoder à l'aide du paramètre attributeType.

2) setMOAttributes – sert à modifier les valeurs d'attribut d'un objet géré existant. En plus d'utiliser objectInstance pour indiquer l'objet géré cible dont les valeurs doivent être modifiées, l'opération utilise également une liste de quadruplets comprenant les éléments attributeName, attributeType, attributeValue et modifyOption, afin de définir les attributs de l'objet géré. Les trois premiers éléments sont les mêmes que ceux dont il est question dans le paragraphe précédent. L'élément modifyOption indique de quelle façon il faut définir les valeurs d'attribut de l'objet géré correspondantes. Il s'agit d'un type énumération composé de "REPLACE", "ADDValues", "REMOVEValues" et "SETToDefault". "REPLACE" signifie que la ou les valeurs d'attribut spécifiées doivent être utilisées pour remplacer la ou les valeurs d'attribut actuelles. "ADDValues" signifie que la ou les valeurs d'attribut spécifiées doivent être ajoutées à la valeur ou aux valeurs d'attribut actuelles. "REMOVEValues" signifie que la ou les valeurs d'attribut spécifiées doivent être retirées de la ou des valeurs d'attribut actuelles. "SETToDefault" signifie que l'attribut doit être réinitialisé à sa valeur par défaut. L'élément modifyOption est facultatif et, dans le cas où il n'est pas spécifié, on procèdera sur la base de "REPLACE".

3) createMO – sert à créer un objet géré dans le système géré. Il faut préciser la classe et le nom de l'objet géré ainsi créé. Le paramètre attributeNameAndValueList est utilisé pour fournir

des valeurs d'attribut, mais on peut s'en passer et, dans ce cas, les attributs prennent les valeurs par défaut.

4) deleteMO – sert à libérer toute ressource associée à un objet géré et à supprimer ce dernier. Le nom distinctif est utilisé pour identifier l'objet géré cible et pour ensuite renvoyer le statut de l'opération. Si l'objet géré cible ou tout objet géré qu'il contient ne peut être supprimé, le statut qui sera renvoyé à l'issue de l'opération sera OperationFailed.

5) getPackages – sert à renvoyer les capacités de l'objet géré cible (un groupe d'attributs ou d'opérations). Une instance d'objet géré peut ou non prendre en charge tous les groupes de capacités définis dans une classe MOC, et le client utilise cette opération pour obtenir les capacités que l'instance d'objet géré prend réellement en charge.

On a souvent recours à des listes et à des structures tabulaires pour fournir une représentation compacte et efficace des données potentiellement complexes qui doivent être transmises en un seul bloc, ce qui pose des problèmes, notamment en ce qui concerne les difficultés qui existent en matière de validation et de compréhension des données. Dans la mesure où les services web sont orientés web et où l'héritage des services irait à l'encontre de leur couplage faible, une telle solution n'est pas recommandée. On peut envisager un modèle de conception basé sur des singletons, afin de préserver la cohérence des données des objets gérés.

Toutes les méthodes dont il est question ci-dessus fonctionnent avec seulement un objet géré. Comme il peut y avoir des millions d'entités à gérer, il est nécessaire que le cadre de gestion prenne en charge les opérations sur objets multiples (MOO) au moyen d'une seule invocation de méthode (ou peut-être un petit nombre d'invocations). Le service MOO offre cette capacité et fait l'objet de la Recommandation [UIT-T Q.818].

On trouvera dans la partie A.2 de l'Annexe A une définition complète d'interface utilisant le schéma XML et le langage WSDL concernant les méthodes génériques d'accès aux objets gérés.

(R) OBJET-2. La mise en oeuvre des méthodes d'accès aux objets gérés doit permettre d'assurer toutes les opérations décrites précédemment et dont les spécifications en WSDL sont définies dans la partie A.2 de l'Annexe A.

10 Héritage des objets gérés et des opérations d'interface

10.1 Héritage d'attributs des objets gérés

Une classe d'objets gérés peut être définie comme une spécialisation d'une autre classe d'objets gérés sur la base du principe de l'héritage. La spécialisation d'une classe MOC implique que toutes les méthodes et tous les attributs définis pour la superclasse seront aussi pris en charge par la sous-classe. Dans le cas des interfaces orientées service et fondées sur les services web, seuls les attributs des classes MOC sont pris en charge. L'ajout de l'héritage des opérations affaiblirait certaines caractéristiques telles que la bonne interopérabilité et le couplage faible.

Tandis que les attributs des objets gérés sont décrits à l'aide du schéma XML, on peut se référer au paragraphe 4.2 "Deriving Types by Extension" de la norme [W3C Primer] pour mettre en place un héritage des attributs. Etant donné que les types de données d'attributs des objets gérés de base sont définis au niveau d'un type complexe dans le schéma XML, la sous-classe peut étendre le type de données des objets gérés de base en utilisant la valeur de l'attribut *base* au niveau de l'élément *extension* dans le schéma XML.

Lorsqu'un type complexe est dérivé par extension, son véritable modèle de contenu correspond au modèle de contenu du type de base, auquel s'ajoute le modèle de contenu spécifié dans la dérivation du type. De plus, les deux modèles de contenu sont considérés comme étant des composants enfant d'un groupe séquentiel. Dans le cas de UKAddress, le modèle de contenu de UKAddress correspond au modèle de contenu de Address, auquel s'ajoutent les déclarations concernant l'élément postcode et l'attribut exportCode.

```
<complexType name="Address">
  <sequence>
   <element name="name"   type="string"/>
   <element name="street" type="string"/>
   <element name="city"   type="string"/>
  </sequence>
 </complexType>
 <complexType name="USAddress">
  <complexContent>
   <extension base="ipo:Address">
    <sequence>
     <element name="state" type="ipo:USState"/>
     <element name="zip"   type="positiveInteger"/>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
```

Pour USAddress, on a ce qui suit:

```
  <sequence>
   <element name="name"   type="string"/>
   <element name="street" type="string"/>
   <element name="city"   type="string"/>
   <element name="state" type="ipo:USState"/>
   <element name="zip"   type="positiveInteger"/>
  </sequence>
</complexType>
```

L'approche ci-dessus peut être utilisée pour l'héritage des éléments qui prend en charge la sémantique relative à l'héritage des attributs d'un objet géré.

10.2 Considérations relatives à l'héritage des opérations d'interface

La dernière version du langage WSDL (version 2.0) prend en charge la syntaxe d'héritage pour les opérations, mais son utilisation n'est pas généralisée dans le secteur privé. Ce sont des versions antérieures du langage WSDL (antérieures à la version 1.1) ne prenant pas en charge une telle syntaxe qui sont largement utilisées dans le secteur privé. C'est pourquoi l'expression de l'héritage des opérations n'entre pas dans le cadre de la présente Recommandation. Cette caractéristique peut être prise en charge, d'un point de vue sémantique, en répétant les définitions des opérations d'interface héritées des superinterfaces.

11 Lignes directrices relatives à la modélisation d'informations concernant les interfaces fondées sur les services web

11.1 Espace de noms

Le présent cadre utilise l'identificateur URI suivant pour l'espace de noms cible:

Tableau 5 – Espace de noms cible du présent cadre de gestion

Recommandation	Espace de noms cible
UIT-T X.782	http://www.itu.int/xml-namespace/itu-t/x.782
UIT-T Q.818	http://www.itu.int/xml-namespace/itu-t/q.818
Autre, du style X.nnnn	http://www.itu.int/xml-namespace/itu-t/x.nnnn

Lorsque d'autres Recommandations seront élaborées, il faudra remplacer "X.nnnn" par le numéro de la Recommandation dans la dernière ligne du tableau ci-dessus.

11.2 Type complexe

Lorsque l'on définit le contenu d'une classe MOC à l'aide du schéma XML, on utilise le complexType XSD pour modéliser la classe MOC. Un complexType XSD contient une séquence pouvant comporter un ou plusieurs élément(s) XSD. Le nom de chaque complexType correspondant à une classe MOC devrait contenir le suffixe "_C".

Il est également possible de définir d'autres types de données d'attribut courants comme étant un complexType, il faut alors que le nom du type prenne le suffixe "Type".

11.3 Attribut

Les attributs et les états d'une classe MOC sont définis comme les éléments du complexType correspondant à la classe MOC. Il est à noter qu'ici, on entend par attribut la propriété conceptuelle d'une entité modélisée comme étant une classe MOC, à ne pas confondre avec le mot clé attribute dans une définition XSD. Dans le présent cadre de gestion, on n'utilisera pas le mot clé attribute dans la définition XSD d'un élément.

11.4 Demande

On utilise une demande pour définir un message concernant les interactions entre un client de service web et une application de serveur pour des services web. Le client de service web envoie au serveur un message contenant la demande. Une demande contient tous les paramètres d'entrée relatifs à une opération dans un service web. Ces paramètres d'entrée peuvent être en plusieurs parties, mais une demande peut aussi ne définir qu'une seule partie, qui associe tous les paramètres d'entrée comme ses propres éléments internes, lesquels sont contenus dans le complexType correspondant.

11.5 Réponse

On utilise une réponse pour définir le message que le serveur de services web renvoie au client. Une réponse contient tous les paramètres de sortie ainsi que la valeur de retour d'une opération dans un service web. Les paramètres de sortie peuvent être en plusieurs parties, mais une réponse peut aussi ne définir qu'une seule partie, qui associe tous les paramètres de sortie comme ses propres éléments internes, lesquels sont contenus dans le complexType correspondant.

11.6 Notification

Toute notification devant être envoyée à travers l'interface de gestion devrait respecter le format spécifié dans la norme [OASIS WSN]. La Recommandation [UIT-T Q.818] définit un en-tête commun à toutes les notifications ainsi que le contenu de certains types de notification courants, notamment:

objectCreation, objectDeletion, attributeValueChange, stateChange, communicationAlarm, environmentalAlarm, equipmentAlarm, processingErrorAlarm, qualityOfServiceAlarm, Violation, integrityViolation, operationalViolation, physicalViolation, securityViolation, timeDomainViolation, relationshipChange et heartbeat.

Lorsque le présent cadre de gestion sera appliqué, il conviendra de se conformer aux définitions des notifications que l'on trouve dans la Recommandation [UIT-T Q.818] dans le cas des types de notification connus. Toute nouvelle définition de notification devra suivre la même méthode et utiliser l'en-tête de notification.

11.7 Conventions de nommage des classes MOC, des paquetages, des attributs et des types de données

Les conventions de nommage suivantes s'appliquent pour la modélisation fondée sur le schéma XML:

– Tous les attributs d'une classe MOC sont définis comme étant un complexType XSD, le nom de cette classe MOC devrait contenir le suffixe "_C" et la première lettre du nom devrait prendre une majuscule, afin que ce complexType ne soit pas confondu avec d'autres définitions de type de données normales. Par exemple: ManagedObject_C, Equipment_C.

– Un attribut d'une classe MOC est défini comme un élément du complexType représentant la classe MOC, et la première lettre de son nom devrait prendre une minuscule.

– On peut définir un paquetage comme un complexType XSD, son nom doit comporter le suffixe "_P" et la première lettre de ce nom doit prendre une majuscule.

– Le nom d'une définition de type de données normale devrait contenir le suffixe "Type" et la première lettre qui le compose devrait prendre une majuscule, afin qu'il soit bien lisible. Par exemple: AdminstrativeStateType.

– Utiliser lowerCamelCase, par exemple "personName", pour les éléments.

– Utiliser UpperCamelCase pour définir les noms de simpleType et de complexType.

– Le nom d'un type contenant un ensemble de valeurs (ensemble non ordonné) devrait comporter le suffixe SetType et le nom d'un type contenant une liste de valeurs (séquence ordonnée) devrait comporter le suffixe ListType.

12 Idiomes de style pour les spécifications de schéma XML[1]

12.1 Modèle de données utilisant le schéma XML

12.1.1 Aperçu de l'utilisation du schéma XML

Le langage XML permet de définir arbitrairement des données en utilisant des étiquettes ad hoc. Il est facile d'utiliser un document XML lorsque celui-ci est contraint par une structure bien définie. Le schéma XML permet de définir les règles, la sémantique et la structure applicables à des documents XML. Un schéma permet de valider la programmation d'un document XML structuré. On définit un schéma XML en utilisant le format XML dans un fichier dont l'extension est .xsd.

12.1.2 Version des spécifications de schéma

L'espace de noms pour le schéma XML 1.1 http://www.w3.org/2001/XMLSchema est identique à celui applicable à un document utilisant le schéma XML 1.0 et l'espace de nom pour le document d'instance XML demeure: http://www.w3.org/2001/XMLSchema-instance. De cette manière, les développeurs de schéma peuvent développer des documents utilisant le schéma XML 1.0 sans se préoccuper de la mise à jour des espaces de noms lorsqu'ils ajoutent des caractéristiques propres au schéma XML 1.1. Ce dernier s'accompagne d'un nouvel espace de noms utile pour le contrôle des versions (http://www.w3.org/2007/XMLSchemaversioning).

La présente Recommandation utilise la version 1.1 du schéma XML, tel que le précise le paragraphe 7.2 de la Recommandation [UIT-T Q.818].

12.2 Considérations relatives à la conception utilisant le schéma XML

12.2.1 Développement d'un schéma unique ou d'un ensemble de schémas associés

Le langage de définition de schémas permet aux structures d'importer un schéma à partir d'autres documents. Lorsque l'on développe un ensemble de schémas associés, les considérations relatives à l'espace de noms revêtent toute leur importance, notamment en ce qui concerne la façon dont le document d'instance XML qui en résulte fait référence aux autres schémas.

– Référence à l'aide de <xsd:import>: L'élément import permet de faire référence à des composants de schéma provenant de documents de schéma dont l'espace de noms cible est différent. Il s'agit là de la méthode de conception de schémas la plus courante, également appelée conception d'espace de noms hétérogène.

– Référence à l'aide de <xsd:include>: L'élément include ajoute au schéma récepteur les composants de schéma provenant d'autres documents de schéma ayant le même espace de noms cible (ou dont l'espace de noms cible n'est pas précisé). Il permet donc d'ajouter au schéma récepteur tous les composants d'un schéma inclus. A partir de là, deux méthodes de conception différentes sont possibles.

– Méthode de conception de l'espace de noms homogène: tous les schémas associés qui sont développés prennent le même espace de noms cible.

– Méthode de conception de l'espace de noms dite caméléon: au moins un schéma pris en charge est défini sans espace de noms cible et le schéma principal inclut le ou les schémas pris en charge. Le ou les schémas pris en charge prennent l'espace de noms du schéma principal.

---

[1] Le texte de cette partie est tiré de la norme [ATIS-I-000002] et a fait l'objet de quelques modifications afin d'être adapté au présent cadre de gestion fondé sur les services web.

Les méthodes de conception homogène et caméléon permettent de modifier les définitions d'un type, d'un groupe ou d'un groupe d'attributs qui sont définies dans le schéma pris en charge. La redéfinition des types concerne les éléments du schéma récepteur et ceux du schéma inclus. Les types ayant été redéfinis peuvent donc interagir avec les types dérivés et créer des conflits.

Dans la présente Recommandation et dans la Recommandation [UIT-T Q.818], seule la méthode [xsd:import] est utilisée.

12.2.2 Modèles de conception des schémas

On mentionne fréquemment les quatre modèles de conception structurelle de schéma suivants en ce qui concerne le schéma XML:

– Poupées russes: ce modèle fournit un élément global unique dans le fichier du schéma. Tous les éléments enfant sont définis dans la hiérarchie de définition de l'élément unique. Cette méthode ne prévoit pas la réutilisation des éléments et nécessite que les éléments soient définis dans un fichier de schéma unique.

– Tranches de salami: dans ce modèle de conception, plusieurs éléments racine sont définis, mais aucun complexType. Ce modèle permet de réutiliser les éléments individuels.

– Jardin d'Eden: dans ce modèle de conception, tous les éléments et les types font l'objet d'une exposition globale et les types complexes sont définis en faisant référence aux éléments globaux réutilisables. La particularité de ce modèle tient à ce qu'il peut y avoir de nombreux éléments racine.

– Stores vénitiens: dans ce modèle de conception, les types sont créés en premier, ce qui permet ensuite de créer les éléments. Les types simples et les types complexes sont créés de façon à être réutilisés le plus possible. La particularité de ce modèle tient à ce qu'il y a un seul élément racine.

Dans le présent cadre de gestion, on utilise le modèle du jardin d'Eden.

12.2.3 Conception visant la résilience

Lorsque l'on développe un schéma, l'un des objectifs est de faire en sorte qu'il résiste aux modifications et de donner la souplesse nécessaire pour anticiper les besoins futurs des utilisateurs. Dans cette partie, on examine quelques-uns des mécanismes qui permettent de conférer souplesse et extensibilité à un modèle de schéma XML.

12.2.3.1 Utilisation de structures génériques

Grâce au schéma W3C, des structures du type <xsd:any> et <xsd:anyAttribute> peuvent permettre à un document d'instance de contenir des données XML qui ne sont pas directement contraintes par le modèle de contenu du schéma XML de définition. De cette manière, le mécanisme d'extensibilité dispose d'une certaine maîtrise, qui peut être spécifiée à l'aide des attributs namespace et processContents (voir les normes [W3C XS-P1] et [W3C XS-P2]). Par exemple, on peut utiliser l'attribut namespace pour contraindre les données XML à un ensemble d'espaces de noms prédéfinis, ce qui permet de valider les données XML étendues dans un document d'instance. L'attribut "processContents" contrôle la façon dont ces données XML étendues sont validées par le valideur XML.

La structure générique <xsd:any> donne aux vendeurs la possibilité de développer des fonctionnalités qui leurs sont propres et qui ne sont pas définies dans le schéma de définition. En utilisant correctement les structures <xsd:any> et <xsd:anyAttribute>, on peut élaborer un modèle de contenu de schéma qui résiste bien aux modifications et est peu sujet à la perturbation de schéma. Il convient toutefois de prendre garde, la structure <xsd:any> peut, par inadvertance, autoriser la création de modèles de contenu non déterministes qui peuvent poser problème dans le cas de certains analyseurs syntaxiques XML. Les développeurs de schéma devront tenir compte de cette contrainte lorsqu'ils utiliseront la structure <xsd:any>.

L'utilisation de la structure <xsd:any> est compatible avec le présent cadre de gestion, mais des explications devraient être fournies chaque fois qu'elle est utilisée.

12.2.3.2 Utilisation de substitutionGroup

Le modèle de contenu d'un membre d'un groupe de substitution est lié à celui des autres membres, par dérivation des types. L'élément remplaçable est appelé élément de tête, et il doit être défini à l'échelle du schéma. Concrètement, la structure substitutionGroup permet de constituer un ensemble d'éléments qui peuvent être spécifiés à l'aide d'un élément générique.

La structure substitutionGroup peut s'avérer utile dans les cas suivants:

– Elaboration de hiérarchies de classes associées: en créant des hiérarchies de classes, on peut faire en sorte que les langages de programmation orientés objet tirent parti de l'héritage.

– Adaptation d'un schéma externe à un besoin particulier: si un élément d'un schéma importé est défini globalement, il est possible de créer une structure substitutionGroup pour cet élément en construisant un type dérivé et de permettre que ce type dérivé soit substitué dans une structure de données complexe.

Les hiérarchies de classes et la structure substitutionGroup entraînent un couplage fort entre les structures de données et peuvent déboucher sur une conception fragile et non modifiable. On peut donc leur préférer des structures telles que <xsd:choice> afin de créer un modèle de contenu composite. La conception composite peut amener de la simplicité et assurer un découplage.

La présente Recommandation utilise une extension du complexType pour représenter la hiérarchie de classes, et la structure substitutionGroup n'est pas utilisée dans le présent cadre de gestion.

12.3 Recommandations à l'intention des développeurs de schémas

12.3.1 Recommandations relatives à la conception utilisant le schéma XML

L'auteur d'une interface fondée sur le langage XML devrait:

– créer des schémas partagés autant que possible.

Il doit par ailleurs:

– utiliser le format d'espace de noms suivant pour les schémas générés par l'UIT-T:

  http://www.itu.int/xml-namespace/itu-t/<ITU-T document number> ou

  http://www.itu.int/xml-namespace/itu-t/<ITU-T document number>/<data model identifier>

– nommer les espaces de noms en utilisant uniquement des caractères en minuscules.

– Il est possible qu'une modification du document n'entraîne que peu de modifications du schéma, voire aucune (c'est par exemple le cas lorsque les références sont mises à jour).

C'est pour cette raison que la structure <ITU-T document number> ne comprend pas le numéro de la version du document. Si le numéro de la version du document apparaissait dans la structure, chaque mise à jour du document entraînerait automatiquement une modification de l'espace de noms.

L'auteur devrait:

– déclarer tous les types simples et complexes au niveau global;

– déclarer les éléments et les attributs au niveau local; un élément principal encapsule tous les autres;

– veiller à ce que l'attribut de la version du schéma (le numéro de la version mineure du schéma) soit présent et augmente chaque fois qu'une modification est apportée au schéma. La valeur initiale devrait être "0";

– veiller à ce que tous les schémas aient au moins 2 espaces de noms: l'espace de noms du schéma XML W3C et l'espace de noms lié à la norme associée;

– donner un exemple d'attributs de schéma pour la version 1.0 du schéma lié au service d'accès à l'objet géré (moas).

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.itu.int/xml-namespace/itu-t/x.782"
   xmlns:moas="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
   version="1.0">
.....
</schema>
```

12.3.2 Recommandations relatives au codage utilisant le schéma XML

L'auteur devrait se conformer aux lignes directrices suivantes concernant les noms des étiquettes des types, des éléments et des attributs:

– utiliser autant que possible des noms courants dans le secteur privé, afin que les documents provenant de ce secteur et les schémas soient cohérents;
– utiliser des noms descriptifs; éviter toute abréviation superflue;
– ne pas utiliser le même nom deux fois dans un même schéma;
– éviter les acronymes autant que possible et les écrire en majuscules lorsque leur emploi est nécessaire;
– éviter les points et les barres obliques;
– n'utiliser que des caractères alphanumériques pour définir les noms d'élément et de type;
– mettre les noms au singulier, à moins que le concept même n'existe qu'au pluriel.

L'auteur devrait envisager de faire de l'élément le type de modèle par défaut et se conformer aux lignes directrices suivantes:

– si l'objet peut être envisagé comme un objet indépendant, il faut en faire un élément;
– si l'objet doit être réutilisable, il faut en faire un type global.

L'auteur devrait en outre se conformer aux directives générales suivantes:

– déclarer les éléments comme étant facultatifs, à moins qu'ils ne soient absolument nécessaires;
– utiliser minOccurs="0" au lieu de nillable="True";
– utiliser maxOccurs. Si une dimension est multiple, il faut la définir. Dans le cas contraire, utiliser maxOccurs="unbounded";
– créer des types simples autant que faire se peut;
– créer un type global lorsque l'élement doit être réutilisable (les types globaux sont définis directement dans l'élément de schéma);
– tous les types doivent être définis globalement;
– utiliser elementFormDefault = "qualified";
– utiliser attributeFormDefault = "unqualified";
– coder à l'aide des caractères UTF-8: <?xml version="1.0" encoding="UTF-8"?>;
– les éléments <documentation> doivent être utilisés chaque fois qu'une réutilisation est probable et que l'on souhaite faire preuve de clarté (l'attribut xsd:lang doit être utilisé pour spécifier le langage);

– recourir aux annotations pour décrire toutes les définitions de types, lesquelles devront au minimum comporter le nom du type en question;
– utiliser exclusivement l'élément xsd:dateTime pour la date et l'heure;
– utiliser uniquement <sequence> ou <choice> lorsqu'un composeur est nécessaire.

Lorsqu'il ajoute des contraintes, l'auteur devrait se conformer aux lignes directrices suivantes:

– lors de la conception d'un nouveau schéma, les nouveaux types simples doivent être contraints et les restrictions appropriées doivent être appliquées, chaque fois que cela est possible;
– identifier les facettes qui contraignent la plage de valeurs;
– choisir le type simple approprié. Par exemple, lorsqu'un nombre est nécessaire, il est préférable d'utiliser, lorsque cela est possible, le type simple nonNegativeInteger ou positiveInteger plutôt que integer;
– lors de la définition d'un type de chaîne, utiliser, si possible, maxLength et des modèles ("pattern"), afin de renseigner des restrictions supplémentaires. Par exemple, lors de la définition d'un numéro de téléphone international à 15 chiffres, on peut utiliser la Recommandation [UIT-T E.164] comme modèle:

```
<xsd:restriction base="xsd:string">
   <xsd:pattern          value="([1-9][0-9]{0,2})(-[1-9][0-9]{0,4})?(-[1-
9][0-9]{0,13})(-[0-9]+)?"/>
</xsd:restriction>
```

– lorsqu'un type simple admet uniquement un ensemble de valeurs prédéfinies, utiliser une facette énumération pour restreindre les valeurs qu'il est autorisé à prendre;
– utiliser les types union si les types de données admettent au moins deux ensembles différents de valeurs qui peuvent être contraints séparément. C'est par exemple le cas pour le code d'état et le code postal.

12.3.3 Structures de schéma XML à éviter

L'auteur devrait se conformer aux lignes directrices suivantes en ce qui concerne les structures de schéma XML:

– ne pas utiliser <xsd:all> (utiliser plutôt <sequence> ou <choice>). Les structures <xsd:sequence> et <xsd:choice> donnent automatiquement un ordre fixe aux éléments enfant dans le document d'instance;
– ne pas utiliser d'instructions de traitement. Ces instructions ne font pas partie du document mais sont transmises à l'application afin de remplir les fonctions propres à cette dernière. Il n'est pas nécessaire que ces instructions suivent la structure interne et, en tant que telles, sont de peu d'utilité dans les définitions de schéma;
– ne pas utiliser d'observations de style DTD ou XML. Les éléments liés à l'annotation et à la documentation fournissent une structure pour les observations et les informations. Les observations de style XML ne sont d'aucune utilité pour le traitement d'un document d'instance;
– ne pas utiliser de redéfinition de groupes ou de groupes XML. La redéfinition d'un groupe peut engendrer un conflit entre le traitement du type ayant été redéfini et les données d'instance du type dérivé;
– ne pas utiliser de groupe de substitution. La structure d'un groupe de substitution crée un couplage fort et augmente la complexité, ce qui peut conduire à une conception fragile et non modifiable;
– ne pas utiliser de valeurs par défaut/fixes. Utiliser de tels attributs induira en erreur les lecteurs du document de schéma, étant donné que ces attributs sont inutiles.

12.4 Lignes directrices relatives aux extensions de schéma

Il est admis que l'auteur d'un schéma ne peut pas anticiper toutes les exigences futures de celui-ci. L'auteur devrait donc utiliser une définition de schéma qui donne une certaine flexibilité et permet à ceux qui mettent le schéma en oeuvre de prendre en charge des données privées ou personnalisées. De manière générale, il devrait anticiper la nécessité de prendre en charge des données privées en prévoyant des structures permettant d'ajouter certaines données de façon générique (par exemple à l'aide de la paire name-value).

Toutefois, un tel ajout peut s'avérer insuffisant dans le cas d'une exigence complexe, où une validation, des assertions de règles et des politiques peuvent par exemple être nécessaires.

L'auteur d'un schéma devrait utiliser l'une des deux possibilités d'extension qui existent pour prendre en charge des données privées dans un schéma:

1) anticiper les exigences liées à l'extension à des parties précises du schéma et permettre aux utilisateurs d'ajouter des données privées openContent; ou
2) permettre l'héritage des types au moyen de <xsd:extension>.

L'utilisation d'une telle extension n'est pas encouragée, car elle pourrait conduire à des problèmes d'interprétation. Par exemple, si la personne chargée de la mise en oeuvre étend un schéma standard en utilisant xsd:extension pour ajouter des éléments et des attributs qui lui sont propres, les données XML qui en résultent pourraient ne pas être correctement analysées par d'autres personnes chargées de mettre le schéma en oeuvre et dont les applications se fondent sur le schéma standard original.

Il est possible d'éviter de telles situations si les deux parties utilisent le même schéma XML. Les extensions sont compatibles avec le présent cadre de gestion, mais les types étendus devraient aussi figurer dans une norme publique.

13 Conformité

Le présent paragraphe définit les critères que doivent respecter les autres documents normatifs déclarés conformes aux directives définies ici ainsi que les fonctions qui doivent être mises en oeuvre par les systèmes déclarés conformes à la présente Recommandation.

13.1 Conformité des documents normatifs

Toute spécification déclarée conforme aux directives définies ici doit:

1) définir toutes les classes qui modélisent les ressources comme étant directement ou indirectement dérivées de *ManagedObject_C*, dont il est question au paragraphe 8.2.1 et qui est défini dans le schéma XML dans le partie A.1 de l'Annexe A;
2) prendre en charge l'héritage des attributs à l'aide du mécanisme spécifié au paragraphe 10.1;
3) utiliser les définitions pour les types d'attribut génériques dont il est question au paragraphe 8.2.4, chaque fois que cela est possible;
4) utiliser les types de données courants définis dans le schéma XML de la partie A.1 de l'Annexe A, lorsqu'il y a lieu;
5) respecter les lignes directrices relatives à la modélisation des interfaces fondées sur les services web spécifiées au paragraphe 11;
6) respecter les conventions relatives à la conception utilisant le schéma XML spécifiées au paragraphe 12.

13.2 Conformité des systèmes

Une mise en oeuvre déclarée conforme à la présente Recommandation doit:

1) prendre en charge toutes les capacités des méthodes d'accès aux objets gérés décrites au paragraphe 9, ainsi que l'interface en WSDL correspondante, comme le définit la partie A.2 de l'Annexe A.

13.3 Directives pour les déclarations de conformité

La déclaration de conformité doit mentionner un document et une année de publication afin de garantir que c'est la bonne version du schéma XML et du langage WSDL qui est identifiée.

Annexe A

Définitions courantes en WSDL et dans le schéma XML

(Cette annexe fait partie intégrante de la présente Recommandation.)

On trouvera dans la présente annexe des définitions courantes d'interfaces en WSDL ainsi que de certains types de données fondés sur le schéma XML.

A.1 Définitions courantes de schémas XML concernant les types de données et un objet géré générique

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema Definition for common data types to be used in this framework.
       Filename : x782.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
   targetNamespace="http://www.itu.int/xml-namespace/itu-t/x.782"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified"
   version="1.0">

<xsd:simpleType name="RDNType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="NameType">
   <xsd:sequence>
      <xsd:element name="rdn" type="x782:RDNType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="NameSetType">
   <xsd:sequence>
      <xsd:element name="dn" type="x782:NameType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="UIDType">
   <xsd:sequence>
      <!-- uri indicates the namespace where the constant is defined. -->
      <xsd:element name="uri" type="xsd:string"/>
      <!-- value indicates the constant value for this item in the above
namespace. -->
      <xsd:element name="value" type="xsd:unsignedLong"/>
   </xsd:sequence>
</xsd:complexType>
```

```
<xsd:complexType name="UIDSetType">
   <xsd:sequence>
      <xsd:element name="uid" type="x782:UIDType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MOClassListType">
   <xsd:sequence>
      <xsd:element name="moClass" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="AdministrativeStateType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="locked"/>
      <xsd:enumeration value="unlocked"/>
      <xsd:enumeration value="suttingDown"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="OperationalStateType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="disabled"/>
      <xsd:enumeration value="enabled"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="AvailabilityStatusType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="inTest"/>
      <xsd:enumeration value="failed"/>
      <xsd:enumeration value="powerOff"/>
      <xsd:enumeration value="offLine"/>
      <xsd:enumeration value="offDuty"/>
      <xsd:enumeration value="dependency"/>
      <xsd:enumeration value="degraded"/>
      <xsd:enumeration value="notInstalled"/>
      <xsd:enumeration value="logFull"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="AvailabilityStatusSetType">
```

```
   <xsd:sequence>
      <xsd:element name="availableState" type="x782:AvailabilityStatusType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="BackedUpStatusType">
   <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>
<xsd:simpleType name="ControlStatusType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="subjectToTest"/>
      <xsd:enumeration value="partOfServicesLocked"/>
      <xsd:enumeration value="reservedForTest"/>
      <xsd:enumeration value="suspended"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="ControlStatusSetType">
   <xsd:sequence>
      <xsd:element name="controlState" type="x782:ControlStatusType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="ExternalTimeType">
   <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>

<xsd:simpleType name="ObjectClassType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="ProceduralStatusType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="initializationRequired"/>
      <xsd:enumeration value="notInitialized"/>
      <xsd:enumeration value="initializing"/>
      <xsd:enumeration value="reporting"/>
      <xsd:enumeration value="terminating"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="ProceduralStatusSetType">
   <xsd:sequence>
```

```
      <xsd:element name="proceduralState" type="x782:ProceduralStatusType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="SourceIndicatorType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="resourceOperation"/>
      <xsd:enumeration value="managementOperation"/>
      <xsd:enumeration value="unknown"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="StandbyStatusType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="hotStandby"/>
      <xsd:enumeration value="coldStandby"/>
      <xsd:enumeration value="providingService"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="StringSetType">
   <xsd:sequence>
      <xsd:element name="value" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="SystemLabelType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="UsageStateType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="idle"/>
      <xsd:enumeration value="active"/>
      <xsd:enumeration value="busy"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="UnknownStatusType">
   <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>

<xsd:complexType name="AttributeValueType">
```

```
   <xsd:sequence>
      <xsd:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="AttributeNameAndValueType">
   <xsd:sequence>
      <xsd:element name="attributeName" type="xsd:string"/>
      <xsd:element name="attributeType" type="xsd:string"/>
      <xsd:element name="attributeValue" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="AttributeNameAndValueSetType">
   <xsd:sequence>
      <xsd:element name="attributeNameAndValue"
type="x782:AttributeNameAndValueType" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="AdditionalTextType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="AnyValueType">
   <xsd:sequence>
      <xsd:element name="typeURI" type="xsd:string"/>
      <xsd:element name="value" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="AdditionalInformationSetType">
   <xsd:sequence>
      <xsd:element name="additionalInfo" type="x782:AnyValueType" minOccurs="0"
maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="NotificationIDType">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="NotificationIDSetType">
   <xsd:sequence>
      <xsd:element name="source" type="x782:NameType" minOccurs="0"
maxOccurs="unbounded"/>
```

```
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CorrelatedNotificationType">
   <xsd:sequence>
      <xsd:element name="source" type="x782:NameType"/>
      <xsd:element name="notifIDs" type="x782:NotificationIDSetType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CorrelatedNotificationSetType">
   <xsd:sequence>
      <xsd:element name="notifications" type="x782:CorrelatedNotificationType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType><xsd:complexType name="AttributeChangeType">
   <xsd:sequence>
      <xsd:element name="attribugteName" type="xsd:string"/>
      <xsd:element name="attributeTypeURI" type="xsd:string"/>
      <xsd:element name="oldValue" type="x782:AttributeValueType"/>
      <xsd:element name="newValue" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="AttributeChangeSetType">
   <xsd:sequence>
      <xsd:element name="attributeChange" type="x782:AttributeChangeType"
minOccurs="1" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ProbableCauseType">
   <xsd:complexContent>
          <xsd:extension base="x782:UIDType"/>
       </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="PerceivedSeverityType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="indeterminate"/>
      <xsd:enumeration value="critical"/>
      <xsd:enumeration value="major"/>
      <xsd:enumeration value="minor"/>
      <xsd:enumeration value="warning"/>
      <xsd:enumeration value="cleared"/>
   </xsd:restriction>
```

```
</xsd:simpleType>

<xsd:simpleType name="TrendIndicationType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="lessSevere"/>
      <xsd:enumeration value="noChange"/>
      <xsd:enumeration value="moreSevere"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ThresholdIndicationType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="up"/>
      <xsd:enumeration value="down"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="ThresholdLevelIndType">
   <xsd:sequence>
      <xsd:element name="indication" type="x782:ThresholdIndicationType"/>
      <!-- observed value -->
      <xsd:element name="low" type="xsd:float" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="high" type="xsd:float"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ThresholdInfoType">
   <xsd:sequence>
      <xsd:element name="attributeID" type="xsd:string"/>
      <xsd:element name="observedValue" type="xsd:float"/>
      <xsd:element name="thresholdLevel" type="x782:ThresholdLevelIndType"/>
      <xsd:element name="armTime" type="xsd:dateTime"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ProposedRepairActionSetType">
   <xsd:complexContent>
      <xsd:extension base="x782:UIDType"/>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SuspectObjectType">
   <xsd:sequence>
      <xsd:element name="moClass" type="xsd:string"/>
      <xsd:element name="suspectedMOInstance" type="x782:NameType"/>
```

```
      <xsd:element name="failureProbability" type="xsd:unsignedShort"
minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SuspectObjectSetType">
   <xsd:sequence>
      <xsd:element name="suspectedMO" type="x782:SuspectObjectSetType"
minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SecurityAlarmCauseType">
   <xsd:complexContent>
      <xsd:extension base="x782:UIDType"/>
   </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SecurityAlarmDetectorType">
   <xsd:sequence>
      <xsd:element name="mechanism" type="x782:UIDType"/>
      <xsd:element name="obj" type="x782:NameType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ServiceUserType">
   <xsd:sequence>
      <xsd:element name="typeURI" type="xsd:string"/>
      <xsd:element name="value" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ServiceProviderType">
   <xsd:sequence>
      <xsd:element name="typeURI" type="xsd:string"/>
      <xsd:element name="value" type="x782:AttributeValueType"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="SpecificProblemSetType">
   <xsd:complexContent>
      <xsd:extension base="x782:UIDSetType"/>
   </xsd:complexContent>
</xsd:complexType>

<!-- XML Schema Definition for generic Managed Object -->
```

```
<xsd:complexType name="ManagedObject_C">
   <xsd:sequence>
      <xsd:element name="objectClass" type="xsd:string"/>
      <xsd:element name="objectInstance" type="x782:NameType"/>
      <xsd:element name="packages" type="x782:StringSetType"/>
      <xsd:element name="creationSource" type="x782:SourceIndicatorType"/>
   </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

A.2 Définitions courantes en WSDL et dans le schéma XML concernant les méthodes d'accès aux objets

(1) Définition dans le schéma XML concernant le service d'accès aux objets gérés de l'UIT

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML Schema Definition for data types to be used in MO access Service
specified in this Recommendation.
       Filename : x782_MOAccessService.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
   xmlns:moas="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
   targetNamespace="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService"
   elementFormDefault="qualified"
   attributeFormDefault="unqualified"
   version="1.0">
<xsd:import namespace="http://www.itu.int/xml-namespace/itu-t/x.782"
schemaLocation="x782.xsd"/>
   <xsd:complexType name="AttributeNameListType">
      <xsd:sequence>
         <xsd:element name="attributeName" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="GetMOAttributesRequestType">
      <xsd:sequence>
         <xsd:element name="objectInstance" type="x782:NameType"/>
         <xsd:element name="attributeNameList"
type="moas:AttributeNameListType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="StatusType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="OperationSucceed"/>
         <xsd:enumeration value="OperationFailed"/>
      </xsd:restriction>
   </xsd:simpleType>
```

```
   <xsd:complexType name="GetMOAttributesResponseType">
      <xsd:sequence>
         <xsd:element name="attributeNameAndValueList"
type="x782:AttributeNameAndValueSetType"/>
         <xsd:element name="status" type=" moas:StatusType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="ModifyOptionType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="REPLACE"/>
         <xsd:enumeration value="ADDValues"/>
         <xsd:enumeration value="REMOVEValues"/>
         <xsd:enumeration value="SETToDefault"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name="AttributeNVMType">
      <xsd:sequence>
         <xsd:element name="attributeName" type="xsd:string"/>
         <xsd:element name="attributeType" type="xsd:string"/>
         <xsd:element name="attributeValue" type="x782:AttributeValueType"/>
         <xsd:element name="modifyOption" type="moas:ModifyOptionType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="AttributeNVMListType">
      <xsd:sequence>
         <xsd:element name="attributeNVM" type=" moas:AttributeNVMType"
minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="SetMOAttributesRequestType">
      <xsd:sequence>
         <xsd:element name="objectInstance" type="x782:NameType"/>
         <xsd:element name="attributeNVMList" type="
moas:AttributeNVMListType"/>
      </xsd:sequence>
   </xsd:complexType>


   <xsd:complexType name="CreateMORequestType">
      <xsd:sequence>
         <xsd:element name="objectClass" type="xsd:string"/>
         <xsd:element name="objectInstance" type="x782:NameType"/>
```

```
         <xsd:element name="attributeNameAndValueList"
type="x782:AttributeNameAndValueSetType"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="GetPackagesResponseType">
      <xsd:sequence>
         <xsd:element name="status" type="moas:StatusType"/>
         <xsd:element name="packages" type="x782:StringSetType"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

(2) Définition en WSDL concernant le service d'accès aux objets gérés de l'UIT

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- WSDL Operation Definition for MO Access Service specified in this
Recommendation.
       Filename : x782_MOAccessService.wsdl -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:x782="http://www.itu.int/xml-namespace/itu-t/x.782"
xmlns:moas="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
name="MOAccessService"
targetNamespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService">
<import namespace="http://www.itu.int/xml-namespace/itu-t/x.782"
location="x782.xsd"/>
<import namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"
location="x782_MOAccessService.xsd"/>

   <wsdl:message name="getMOAttributesRequest">
      <wsdl:part name="getMOAttributesInput"
type="moas:GetMOAttributesRequestType"/>
   </wsdl:message>
   <wsdl:message name="getMOAttributesResponse">
      <wsdl:part name="getMOAttributesOutput"
type="moas:GetMOAttributesResponseType"/>
   </wsdl:message>
   <wsdl:message name="setMOAttributesRequest">
      <wsdl:part name="setMOAttributesInput"
type="moas:SetMOAttributesRequestType"/>
   </wsdl:message>
   <wsdl:message name="setMOAttributesResponse">
      <wsdl:part name="status" type="moas:StatusType"/>
   </wsdl:message>
   <wsdl:message name="createMORequest">
      <wsdl:part name="createMOInput" type="moas:CreateMORequestType"/>
   </wsdl:message>
   <wsdl:message name="createMOResponse">
```

```
      <wsdl:part name="status" type="moas:StatusType"/>
   </wsdl:message>
   <wsdl:message name="deleteMORequest">
      <wsdl:part name="objectInstance" type="x782:NameType"/>
   </wsdl:message>
   <wsdl:message name="deleteMOResponse">
      <wsdl:part name="status" type="moas:StatusType"/>
   </wsdl:message>
   <wsdl:message name="getPackagesRequest">
      <wsdl:part name="objectInstance" type="x782:NameType"/>
   </wsdl:message>
   <wsdl:message name="getPackagesResponse">
      <wsdl:part name="getPackageOutput" type="moas:GetPackagesResponseType"/>
   </wsdl:message>
   <wsdl:portType name="MOAccessServicePortType">
      <wsdl:operation name="getMOAttributes">
         <wsdl:input message="moas:getMOAttributesRequest"/>
         <wsdl:output message="moas:getMOAttributesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="setMOAttributes">
         <wsdl:input message="moas:setMOAttributesRequest"/>
         <wsdl:output message="moas:setMOAttributesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="createMO">
         <wsdl:input message="moas:createMORequest"/>
         <wsdl:output message="moas:createMOResponse"/>
      </wsdl:operation>
      <wsdl:operation name="deleteMO">
         <wsdl:input message="moas:deleteMORequest"/>
         <wsdl:output message="moas:deleteMOResponse"/>
      </wsdl:operation>
      <wsdl:operation name="getPackages">
         <wsdl:input message="moas:getPackagesRequest"/>
         <wsdl:output message="moas:getPackagesResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="MOAccessServiceBinding"
type="moas:MOAccessServicePortType">
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="getMOAttributes">
         <soap:operation soapAction="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/getMOAttributes"/>
         <wsdl:input>
```

```
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="
http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="
http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="setMOAttributes">
         <soap:operation soapAction=" http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/setMOAttributes"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="
http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="createMO">
         <soap:operation soapAction="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/createMO"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="deleteMO">
         <soap:operation soapAction="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/deleteMO"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
```

```
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getPackages">
         <soap:operation soapAction="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService/getPackages"/>
         <wsdl:input>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://www.itu.int/xml-namespace/itu-t/x.782/MOAccessService"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="MOAccessService">
      <wsdl:port name="MOAccessService" binding="moas:MOAccessServiceBinding">
         <soap:address location="http://www.itu.int/xml-namespace/itu-
t/x.782/MOAccessService"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Appendice I

Aperçu de la technologie des services web et scénarios d'application pour les interfaces de gestion de réseau

(Cet appendice ne fait pas partie intégrante de la présente Recommandation.)

Dans cet appendice, on trouvera un aperçu des technologies des services web.

I.1 Caractéristiques de la technologie des services web

Les services web sont un mécanisme simplifié qui permet de connecter des applications indépendamment de la technologie ou des dispositifs utilisés, ou de leur emplacement. Ils sont basés sur des protocoles normalisés utilisés dans le secteur privé et pris en charge par tous les vendeurs, qui permettent de tirer parti de l'Internet en ce qui concerne les communications à bas coût ainsi que d'autres mécanismes de transport. La technique de couplage faible pour les messages est compatible avec une connectivité multiple et des scénarios d'échange d'informations grâce à des services autodescriptifs qui peuvent être découverts automatiquement.

Contrairement aux environnements répartis classiques, les services web mettent l'accent sur l'interopérabilité. Ils ne dépendent d'aucun langage de programmation, tandis que les environnements classiques sont généralement liés à un langage en particulier. De même, puisqu'ils peuvent facilement être liés à différents mécanismes de transport, les services web sont plus souples quant au choix de ces mécanismes. En outre, ils ne sont généralement pas liés à un cadre client ou à un cadre serveur particulier, contrairement aux environnements classiques. Dans l'ensemble, ils conviennent bien à un ensemble de relations dont le couplage est faible et la granularité grossière. Le fait qu'ils utilisent le langage XML est un avantage supplémentaire des services web, car ce langage permet d'utiliser des documents dans des environnements très variés.

Un service web est une technologie fondée sur le schéma XML, qui présente les avantages suivants en ce qui concerne les applications logicielles.

1) Bonne interopérabilité

Les services web sont interopérables en toutes circonstances car ils utilisent des protocoles indépendants des plates-formes et des langages, comme le protocole SOAP. Ils apportent un niveau d'interopérabilité supplémentaire entre les applications logicielles. De nombreux fournisseurs de plates-formes, développeurs de logiciels et fournisseurs de services communs rendent leurs logiciels compatibles avec les capacités SOAP, WSDL et UDDI.

2) Couplage faible

Les services web sont des modules logiciels autodescriptifs qui encapsulent des fonctionnalités discrètes. Ils sont accessibles via des protocoles de communication Internet normalisés tels que XML et SOAP. Ils peuvent être développés dans de nombreux langages de mise en oeuvre différents, et tout autre service web ou application peut y avoir accès. Les services web appartiennent donc à la catégorie des applications à couplage faible.

3) Utilisation généralisée

Les services web sont désormais largement utilisés dans le secteur des services informatiques, notamment dans le domaine du commerce électronique et des applications interentreprises. On trouve déjà des plates-formes stables qui permettent de développer des applications des services web.

4) Réutilisation des logiciels et des données

Les services web sont compatibles avec le modèle de développement des logiciels basé sur les composants, qui permet aux développeurs de réutiliser les composants fondamentaux créés par d'autres afin d'assembler des applications complexes et de leur ajouter des extensions.

Les services web permettent non seulement de réutiliser le code source, mais également les données qui se trouvent derrière le code source. Une autre façon de réutiliser les logiciels dans les services web consiste à intégrer les fonctions en question dans plusieurs applications et de les exposer grâce à des interfaces de service web.

5) Composition de services facile

Les services web permettent de définir des applications de plus en plus complexes en ajoutant progressivement des composants à des niveaux d'abstraction plus avancés. Un client qui invoque un service composite peut à son tour être exposé comme un service web. En combinant les fonctionnalités de plusieurs services web, on peut ainsi obtenir un nouveau service web: on parle de composition de services. La composition peut être opérée sur des services simples ou sur des services composites.

Les services web sont un moyen harmonisé d'exposer les fonctionnalités des applications et d'y accéder; des langages spécialisés, par exemple le BPEL, permettent de définir et d'exécuter les processus d'entreprise associés.

6) Faible coût

Actuellement, de nombreux outils, produits et technologies prennent en charge les normes liées aux services web. On dispose donc d'un large choix lorsqu'il s'agit de réduire les coûts liés au développement et à l'exploitation de nouvelles applications, les coûts liés à l'environnement et les coûts d'intégration.

I.2 Scénarios appropriés et inappropriés d'application des services web à la gestion de réseau

(1) Scénarios d'application généralement appropriés

En général, selon les caractéristiques des services web, les scénarios d'application suivants sont appropriés.

– Communication à travers un pare-feu

Etant donné que les services web utilisent le protocole normalisé SOAP comme protocole de transfert, ils peuvent facilement passer à travers les pare-feu ou les serveurs proxy qui peuvent exister entre différentes applications associées. La situation est généralement plus complexe lorsque d'autres intergiciels de communication sont utilisés.

– Intégration d'une application

Le fait est que les applications nécessitent souvent d'exécuter des programmes sur un type de plate-forme tandis que les applications auprès desquelles elles doivent obtenir des données ou auxquelles elles doivent envoyer des données utilisent d'autres plates-formes. Même lorsqu'il n'y a qu'une plate-forme, il est souvent nécessaire d'intégrer divers logiciels de différents fabricants. Grâce aux services web, les applications peuvent exposer auprès d'autres applications les fonctions et les données en vue d'une utilisation standard.

– Intégration d'interfaces B2B

Pour désigner l'intégration de transactions commerciales interentreprises, on parle généralement d'intégration B2B. Grâce aux services web, une entreprise peut intégrer des applications commerciales essentielles et les exposer aux fournisseurs et aux clients voulus; l'utilisation de services web dans le cadre de l'intégration B2B permet d'assurer l'interopérabilité en toute simplicité, c'est là son principal avantage.

– Interface ouverte garantissant une bonne interchangeabilité

Les interfaces de services web sont définies dans le langage de description des services web (WSDL). Le langage WSDL définit les services comme étant des ensembles de points d'extrémité et de ports d'un réseau. Dans cette optique, la spécification WSDL fournit un format XML que les documents doivent respecter. La définition abstraite des ports et des messages est dissociée de leur utilisation concrète ou de leur instance, ce qui permet de la réutiliser. De cette façon, le langage WSDL décrit l'interface ouverte et publique pour les services web. Etant donné que la définition des données et la définition des interfaces sont dissociées, le client et le serveur des applications des services web peuvent être développés séparément, ils peuvent communiquer à travers cette interface ouverte, et le changement d'interfaces n'aura que peu de conséquences sur le développement des applications des services web.

(2) Scénarios d'application généralement inappropriés

Un service web étant une technologie à faible couplage conçue principalement pour l'interopérabilité et l'intégration des applications dans un environnement hétérogène, elle comporte aussi des faiblesses, notamment une faible vitesse d'exécution et une efficacité de codage inférieure par rapport à d'autres technologies (par exemple l'architecture CORBA ou le modèle DCOM). Dans certains cas, mieux vaut ne pas utiliser les services web. C'est par exemple le cas:

– des systèmes à machine unique;

– des applications de réseau local isomorphes (applications en interfonctionnement dans un même réseau local et faisant appel à la même plate-forme et au même intergiciel);

– des interactions en ligne avec une grande quantité de données.

(3) Considérations relatives aux scénarios d'application des services web dans le cadre de la gestion de réseau

Au vu de l'analyse qui précède, on considère que dans un domaine de gestion de réseau, les interfaces suivantes conviennent mieux aux applications des services web:

– les interfaces B2B/C2B;

– les interfaces F;

– les interfaces OS-OS de haut niveau (couche gestion de service ou couche gestion d'entreprise), etc.

Dans les cas qui précèdent, on peut tirer parti des avantages des services web, comme le couplage faible, l'intégration d'applications interentreprises, l'accès fondé sur le web et la bonne extensibilité dans le cas des nouvelles applications de service.

L'utilisation des services web peut s'avérer inopportune dans le cas des interfaces de gestion EMS-NE, dans la mesure où elles sont en général fournies par le même vendeur et ne sont donc pas nécessairement conçues comme étant des interfaces ouvertes.

Pour les interfaces de gestion NMS-EMS, il est possible d'utiliser les services web, bien que dans certains cas, cela ne soit pas le plus judicieux. Par exemple, dans le cas d'un réseau local, l'architecture CORBA présente une meilleure efficacité de codage et une plus grande vitesse d'exécution.

Bibliographie

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

[b-UIT-T X.780.2] Recommandation UIT-T X.780.2 (2007), *Lignes directrices relatives au RGT pour la définition d'objets gérés et d'objets de façade CORBA orientés service.*

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série D	Principes de tarification et de comptabilité et questions de politique générale et d'économie relatives aux télécommunications internationales/TIC
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	Environnement et TIC, changement climatique, déchets d'équipements électriques et électroniques, efficacité énergétique; construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	Gestion des télécommunications y compris le RGT et maintenance des réseaux
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 et mesures et tests associés
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, communication entre systèmes ouverts et sécurité**
Série Y	Infrastructure mondiale de l'information, protocole Internet, réseaux de prochaine génération, Internet des objets et villes intelligentes
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse
Genève, 2017