UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Q.822.1

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(10/2001)

SÉRIE Q: COMMUTATION ET SIGNALISATION

Interface Q3

Service RGT de gestion de la qualité de fonctionnement en architecture Corba

Recommandation UIT-T Q.822.1

RECOMMANDATIONS UIT-T DE LA SÉRIE Q

COMMUTATION ET SIGNALISATION

SIGNALISATION DANS LE SERVICE MANUEL INTERNATIONAL	Q.1–Q.3
EXPLOITATION INTERNATIONALE AUTOMATIQUE ET SEMI-AUTOMATIQUE	Q.4–Q.59
FONCTIONS ET FLUX D'INFORMATION DES SERVICES DU RNIS	Q.60–Q.99
CLAUSES APPLICABLES AUX SYSTÈMES NORMALISÉS DE L'UIT-T	Q.100–Q.119
SPÉCIFICATIONS DES SYSTÈMES DE SIGNALISATION N° 4 ET N° 5	Q.120–Q.249
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION N° 6	Q.250–Q.309
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION R1	Q.310–Q.399
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION R2	Q.400–Q.499
COMMUTATEURS NUMÉRIQUES	Q.500–Q.599
INTERFONCTIONNEMENT DES SYSTÈMES DE SIGNALISATION	Q.600–Q.699
SPÉCIFICATIONS DU SYSTÈME DE SIGNALISATION N° 7	Q.700–Q.799
Généralités	Q.700
Sous-système transport de messages	Q.701–Q.709
Sous-système commande des connexions sémaphores	Q.711–Q.719
Sous-système utilisateur de téléphonie	Q.720–Q.729
Services complémentaires du RNIS	Q.730–Q.739
Sous-système utilisateur de données	Q.740–Q.749
Gestion du système de signalisation n° 7	Q.750–Q.759
Sous-système utilisateur du RNIS	Q.760–Q.769
Sous-système application de gestion des transactions	Q.770–Q.779
Spécification des tests	Q.780–Q.799
INTERFACE Q3	**Q.800–Q.849**
SYSTÈME DE SIGNALISATION D'ABONNÉ NUMÉRIQUE N° 1	Q.850–Q.999
Généralités	Q.850–Q.919
Couche Liaison de données	Q.920–Q.929
Couche Réseau	Q.930–Q.939
Gestion utilisateur-réseau	Q.940–Q.949
Description d'étape 3 des services complémentaires utilisant le système DSS1	Q.950–Q.999
RÉSEAUX MOBILES TERRESTRES PUBLICS	Q.1000–Q.1099
INTERFONCTIONNEMENT AVEC LES SYSTÈMES MOBILES À SATELLITES	Q.1100–Q.1199
RÉSEAU INTELLIGENT	Q.1200–Q.1699
PRESCRIPTIONS ET PROTOCOLES DE SIGNALISATION POUR LES IMT-2000	Q.1700–Q.1799
SPECIFICATIONS DE LA SIGNALISATION RELATIVE À LA COMMANDE D'APPEL INDÉPENDANTE DU SUPPORT	Q.1900–Q.1999
RNIS À LARGE BANDE	Q.2000–Q.2999

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

Recommandation UIT-T Q.822.1

Service RGT de gestion de la qualité de fonctionnement en architecture Corba

Résumé

La présente Recommandation définit un modèle d'information à utiliser en gestion de la qualité de fonctionnement (PM, *performance management*) sur la base de l'architecture CORBA. Elle définit un langage de définition d'interface (IDL, *interface definition language*), une série d'interfaces, des notifications et des constantes. L'objet de la présente Recommandation est de définir un modèle CORBA/IDL semblable à celui qui a été défini dans la Rec. UIT-T X.739 et la Rec. UIT-T Q.822 au moyen d'éléments CMISE. La présente Recommandation est conforme aux normes de modélisation en architecture CORBA (Rec. UIT-T X.780, X.780.1, Q.816, Q.816.1 et M.3120).

Source

La Recommandation Q.822.1 de l'UIT-T, élaborée par la Commission d'études 4 (2001-2004) de l'UIT-T, a été approuvée le 7 octobre 2001 selon la procédure définie dans la Résolution 1 de l'AMNT.

AVANT-PROPOS

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

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

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

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

NOTE

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

DROITS DE PROPRIÉTÉ INTELLECTUELLE

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

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

TABLE DES MATIÈRES

Recommandation UIT-T Q.822.1

Service RGT de gestion de la qualité de fonctionnement en architecture Corba

1 Domaine d'application

La présente Recommandation définit un service support à utiliser dans la gestion de qualité de fonctionnement des télécommunications (PM) sur la base de l'architecture CORBA. Elle définit en langage de définition d'interface (IDL, *interface definition language*) une série d'interfaces tant à granularité fine qu'en version façade, des types de données et des constantes. Elle vise à définir une spécification CORBA/IDL semblable à celle qui a été définie dans la Rec. UIT-T X.739 et la Rec. UIT-T Q.822 au moyen d'éléments CMISE. La présente Recommandation est conforme aux normes de modélisation CORBA proposées dans les Rec. UIT-T X.780, X.780.1, Q.816, Q.816.1 et M.3120.

2 Références normatives

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

[1] OMG Document formal/99-10-07, *The Common Object Request Broker: Architecture and Specification* (Architecture et spécification de gestionnaire commun de requêtes d'objets), Revision 2.3.1.

[2] OMG Document formal/2000-06-20, *Notification Service Specification* (Spécification du service de notification), Version 1.0.

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

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

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

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

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

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

[9] Recommandation UIT-T X.738 (1993), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: fonction de récapitulation*.

[10] Recommandation UIT-T X.739 (1993), *Technologies de l'information – Interconnexion des systèmes ouverts – Gestion-systèmes: objets et attributs métriques*.

[11] Recommandation UIT-T Q.822 (1994), *Description d'étape 1, d'étape 2 et d'étape 3 de l'interface Q3 – Gestion de la qualité de fonctionnement*.

[12] Recommandation UIT-T Q.823 (1996), *Spécifications fonctionnelles d'étape 2 et d'étape 3 de la gestion de trafic*.

3 Définitions

La présente Recommandation utilise les définitions suivantes:

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

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

DN nom distinctif (*distinguished name*)

FDN nom distinctif absolu (*fully distinguished name*)

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

NE élément de réseau (*network element*)

PM gestion de la qualité de fonctionnement (*performance management*)

QS qualité de service

RDN nom distinctif relatif (*relative distinguished name*)

RGT réseau de gestion des télécommunications

SNMP protocole simple de gestion de réseau (*simple network management protocol*)

TL1 langage de transaction 1 (*transaction language 1*)

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

4 Aperçu général du service de gestion de la qualité de fonctionnement

4.1 Exigences

4.1.1 Exigences fonctionnelles

Le mesurage de la qualité de fonctionnement est utilisé de deux façons dans le RGT. La première consiste à mesurer la qualité des entités de transport et de protocole. Dans cette méthode, les mesures doivent respecter un seuil et sont collectées à intervalles non critiques en temps réel pour la chronologie de l'ingénierie et du service. C'est l'utilisation qui est prise en charge par l'application de la Rec. UIT-T Q.822 et des normes techniques spécifiques qui sont directement fondées sur elle. La seconde façon d'utiliser les mesures de qualité de fonctionnement consiste à prendre en charge la gestion du trafic réseau. Dans cette application, les mesures ne sont pas soumises à un seuil mais sont collectées périodiquement, à intervalles suffisants pour prendre en charge l'application des commandes de gestion de réseau. L'intervalle typique pour cette acquisition est normalement de 5 minutes. C'est l'utilisation prise en charge par la Rec. UIT-T Q.823, qui réutilise les mécanismes de la Rec. UIT-T Q.822 ainsi que le simple releveur générique de la Rec. UIT-T X.738 afin de produire un rapport de type relevé unique récapitulant la précédente période de granularité. Cette utilisation est signalée à un système d'exploitation de gestion réseau. Ces deux utilisations des mesures de qualité de fonctionnement sont résumées dans les cas d'utilisation de la Figure 1.

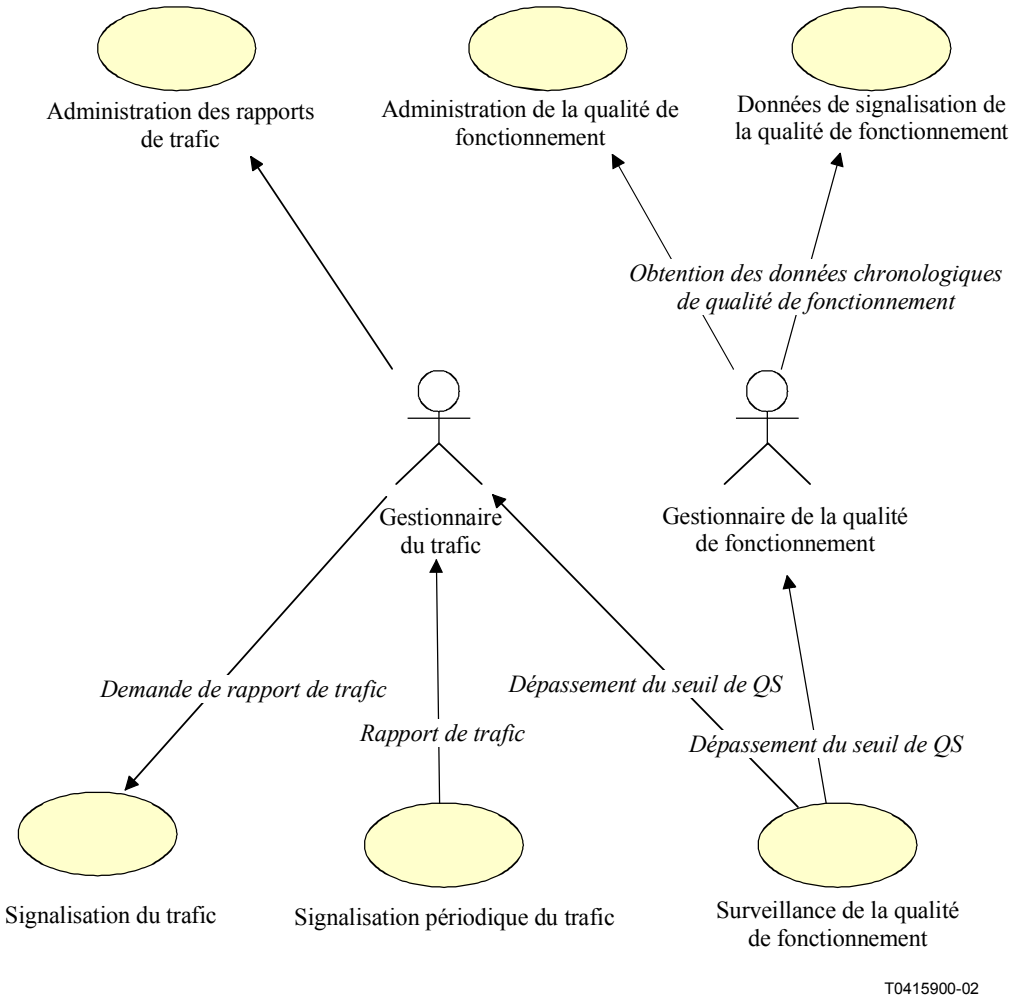

Figure 1/Q.822.1 – Cas d'utilisation des mesures de qualité de fonctionnement

En résumé, les fonctions RGT de gestion de la qualité de fonctionnement (PM) comprennent l'acquisition des données, leur stockage, la fixation des seuils et la signalisation des données.

L'acquisition des données de qualité de fonctionnement se rapporte à la capacité d'un élément de réseau (NE) à recueillir les diverses données PM concernant une certaine entité surveillée dans cet élément NE. Cette fonction permet à un gestionnaire RGT d'attribuer un intervalle d'acquisition des données PM, de suspendre/reprendre le processus d'acquisition de données PM, et de réinitialiser les compteurs de surveillance de la qualité de fonctionnement.

Le stockage des données de fonctionnement se rapporte à la capacité d'un élément NE à mémoriser des données PM chronologiques de chaque entité surveillée pendant la durée prescrite. Cette fonction permet à un gestionnaire RGT de fixer la durée pendant laquelle un enregistrement spécifique de données PM chronologiques sera conservé, éventuellement de filtrer des données chronologiques sur la base de certains critères (comme la suppression des données "tout zéro") et de supprimer les données PM chronologiques à la fin d'un intervalle.

La fixation des seuils de qualité de fonctionnement se rapporte à la capacité d'un élément NE à informer un gestionnaire RGT de tout dépassement de seuil. Elle permet également au gestionnaire RGT de fixer des critères de fixation des seuils.

La signalisation des données de qualité de fonctionnement se rapporte à la capacité d'un élément NE à signaler des données PM sur une base programmée, ou à la suite d'une demande spontanée du gestionnaire RGT. Un rapport peut contenir des données provenant d'une certaine entité surveillée ou contenir des données provenant d'un ensemble d'entités surveillées. Cette fonction permet à un gestionnaire RGT de demander des données PM, d'autoriser/interdire l'émission de rapport programmés dans l'élément NE, et de filtrer les rapports de données PM selon certains critères (comme la suppression des données de type "tout zéro").

4.1.2 Exigences de conception

La présente Recommandation suit les directives de modélisation informationnelle qui sont définies dans la Rec. UIT-T X.780. Par ailleurs, elle s'efforce de faire en sorte que le modèle reste pratique, simple et accessible aux utilisateurs comme aux produits.

4.2 Modélisation du service

4.2.1 Domaine d'application

Afin de prendre en charge les exigences fonctionnelles, le service de gestion de la qualité de fonctionnement a besoin des données de type currentData pour collecter les mesures de qualité, des données de type historyData afin de mémoriser les mesures collectées, des données de type thresholdData afin de spécifier les seuils, et d'un releveur spécifique afin de signaler les données chronologiques. Comme les données currentData sont héritées de l'objet Scanner, le modèle a également besoin de celui-ci.

Sur la base des directives de modélisation et des besoins issus des applications PM existantes et actuelles, toutes les capacités définies pour ces objets gérés dans la Rec. UIT-T X.739 et la Rec. UIT-T Q.822 ne sont cependant pas nécessaires actuellement. Par conséquent, certains des paquetages facultatifs ne sont pas inclus dans le présent modèle. Si le besoin d'une capacité apparaît ultérieurement, le ou les paquetages correspondants seront réexaminés pour inclusion.

Pour la plupart des objets gérés mentionnés ci-dessus, leur modèle en langage IDL peut être déduit d'une conversion des directives GDMO selon la Rec. UIT-T X.739 et la Rec. UIT-T Q.822. Pour les données chronologiques, une conversion directe peut cependant se traduire par un nombre éventuellement important d'objets gérés. Ces entités sont des mémoires en lecture seulement de données chronologiques et n'ont pas besoin d'être des interfaces CORBA. Il est en revanche proposé qu'elles soient définies comme des types de valeur CORBA. La conversion Q.822 définira un type de valeur de base pour les données chronologiques, appelé HistoryDataValueType. Ce type sera accessible à l'interface CurrentData par certaines méthodes. Si une donnée actuelle est rangée dans une sous-classe, il est prévu que le type de valeur de données chronologiques correspondant soit également rangé dans cette sous-classe.

Comme les données à mémoriser ne se trouvent pas dans les interfaces CORBA mais dans les types de valeur de données chronologiques, la simple conversion des releveurs simples et homogènes des directives GDMO ne suffira pas pour signaler les données chronologiques. Il faudra plutôt un nouveau type de releveur, capable de collecter des données à partir des types de valeur de données chronologiques. Le présent modèle s'efforce, chaque fois que possible, de conserver aux interfaces leur caractère fortement dépendant du type. En effet, l'acquisition des données de trafic est à fort contenu calculatoire car il faut collecter de grandes quantités de données puis les traiter très rapidement dans le gestionnaire de réseau.

A cette fin, le releveur de données chronologiques HistoryDataScanner est appelé à extraire les données chronologiques sauvegardées par les objets CurrentData correspondants. Un mécanisme de transfert de fichiers est utilisé dans le releveur de données chronologiques afin d'extraire de grandes quantités de données car d'autres techniques sont susceptibles d'être à la fois plus lentes et plus consommatrices de ressources de calcul et de communication. La présente Recommandation traite de

la capacité de commander la production du fichier, la sélection de son format et la notification envoyée à l'utilisateur pour l'informer du moment où le fichier est produit, avec les informations pouvant être nécessaires pour obtenir le fichier. Le mécanisme spécifique de transfert de fichier et son administration sont cependant hors du domaine d'application de la présente Recommandation.

4.2.2 Modèle UML

Le présent paragraphe décrit le modèle UML pour la gestion de la qualité de fonctionnement. Ce modèle s'applique aux classes Scanner, CurrentData, HistoryData (représentée par la classe HistoryDataValueType), ThresholdData et HistoryDataScanner. Etant donné que ce modèle ne peut pas être reproduit sur une seule page, il est décrit par les Figures 2, 3 et 4.

Le paragraphe suivant décrit en langage UML le modèle résultant. Les diagrammes UML ne montrent le modèle que selon la méthode de granularité fine mais ils s'appliquent aux deux méthodes, à granularité fine et à façade. L'interface-façade IDL est convertie mécaniquement de l'interface IDL à granularité fine qui est définie dans la Rec. UIT-T X.780.1.

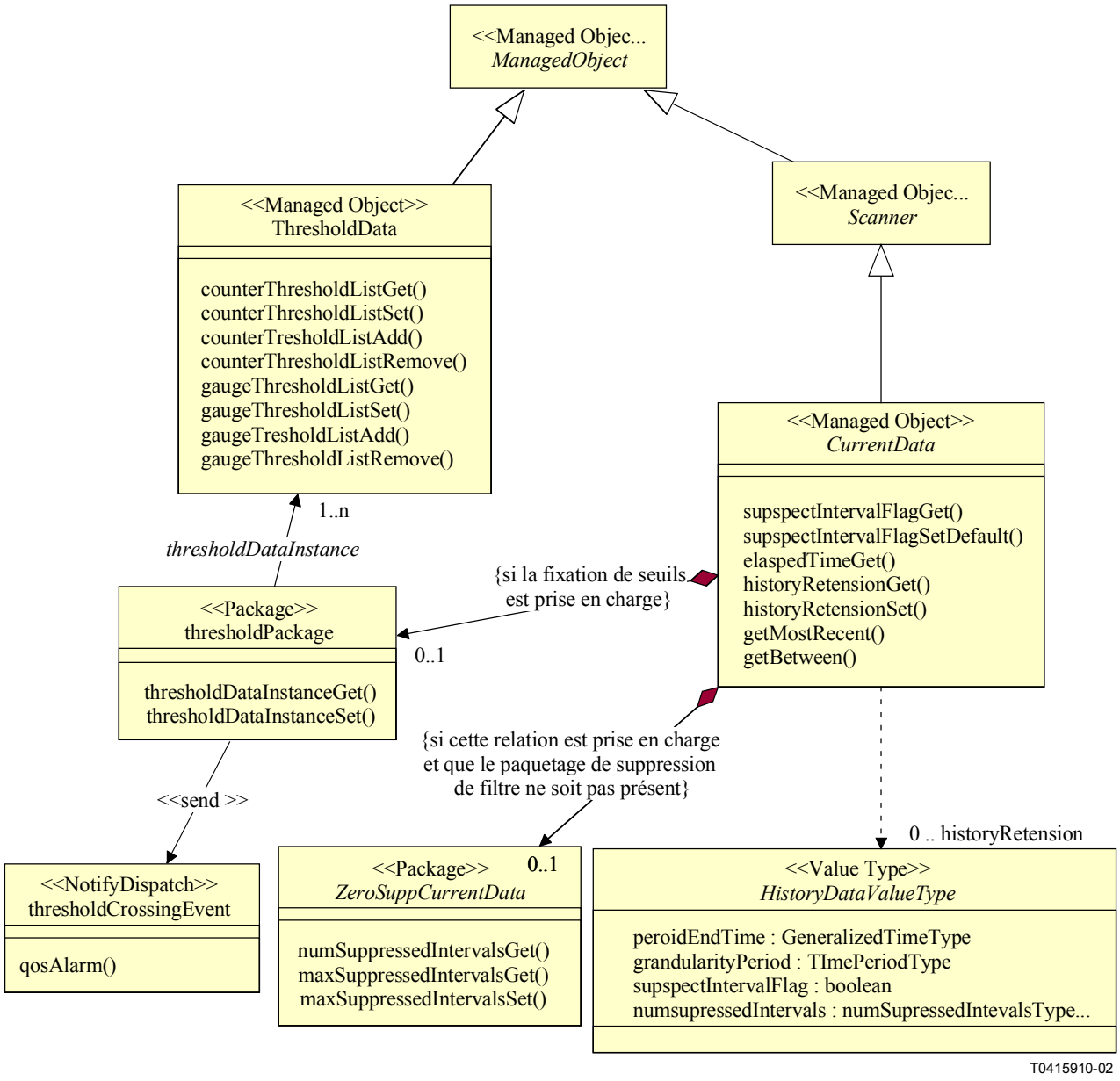

Figure 2/Q.822.1 – Diagramme des classes CurrentData, HistoryData et ThresholdData

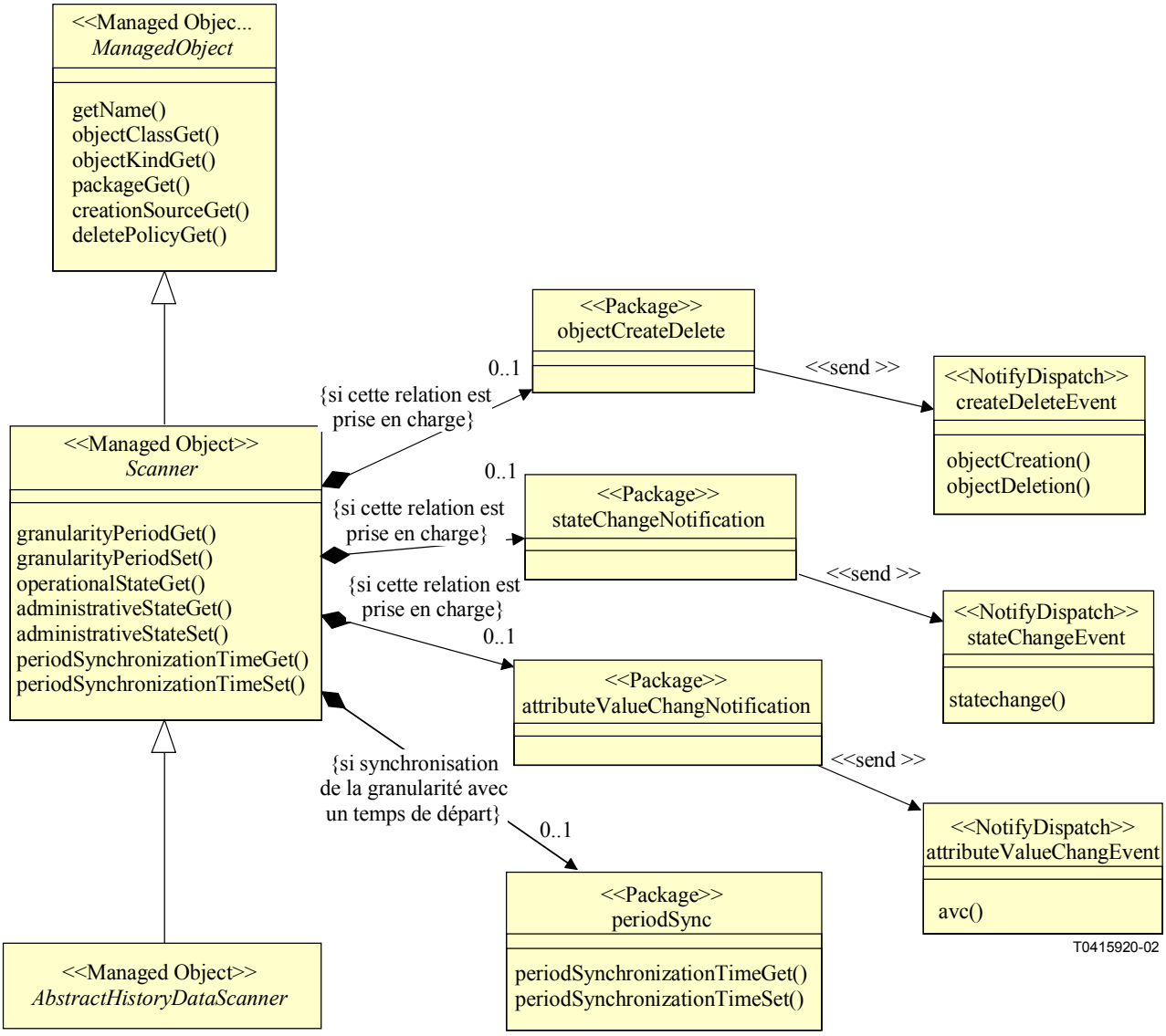

Figure 3/Q.822.1 – Diagramme de la classe Scanner

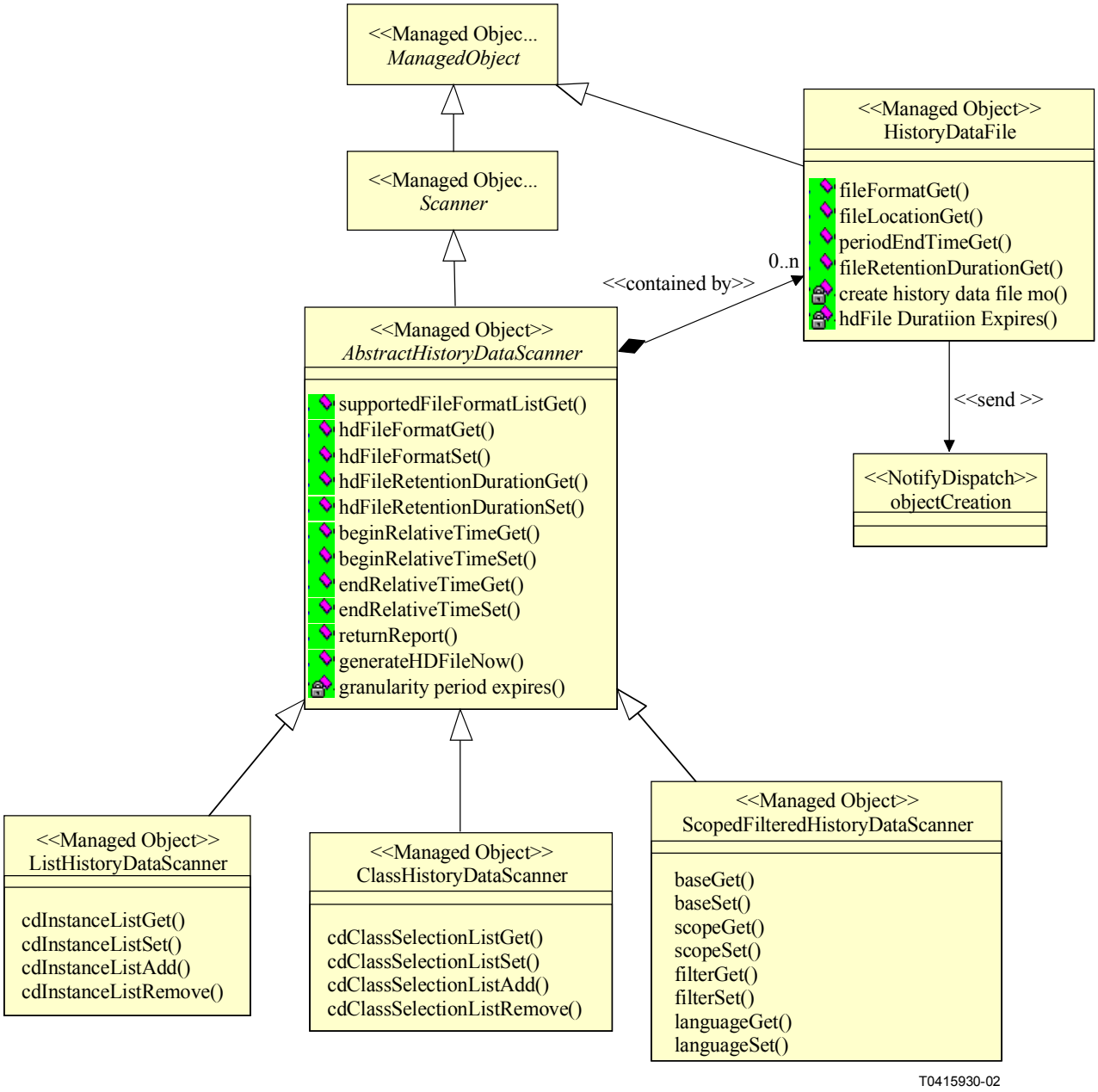

Figure 4/Q.822.1 – Diagramme des classes HistoryDataScanner et HistoryDataFile

4.2.3 Relation de confinement

Le présent paragraphe décrit la relation de confinement des interfaces CORBA définies dans la présente Recommandation.

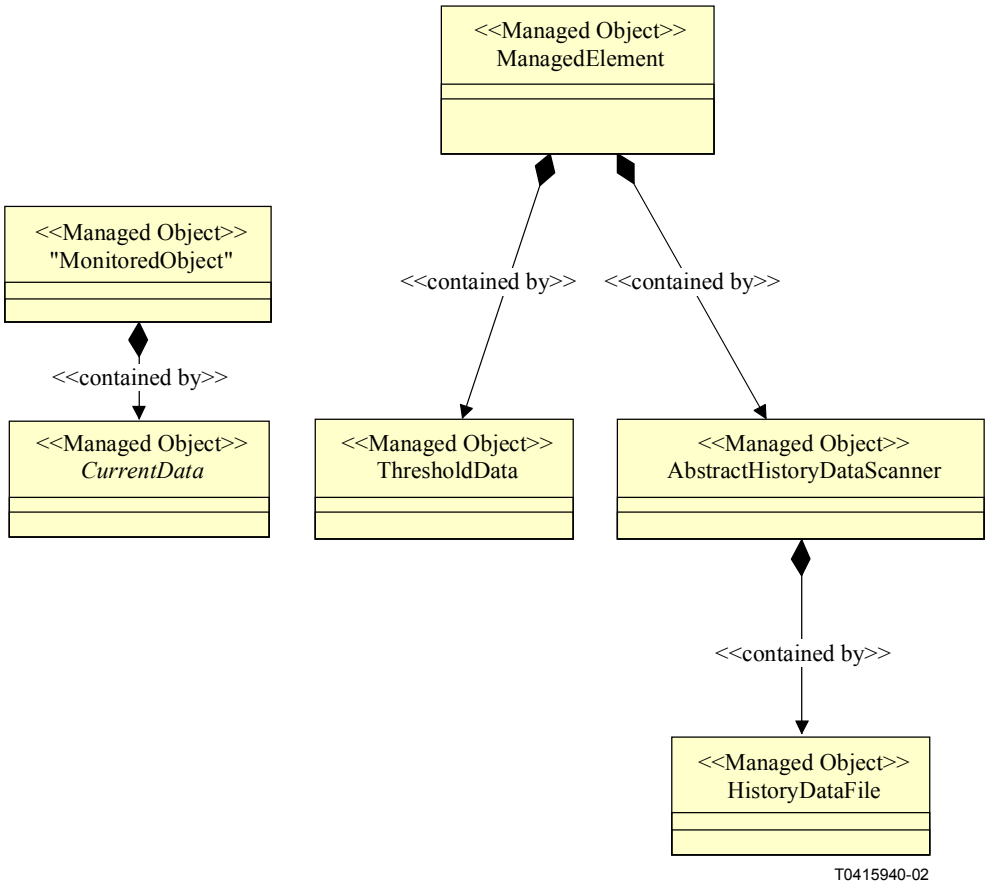

Figure 5/Q.822.1 – Relation de confinement

4.3 Description de l'interface de service

4.3.1 Interface *Scanner*

Un objet géré de cette interface représente la capacité d'extraire des valeurs d'attributs d'objets gérés et de produire, à partir de ces valeurs, des informations récapitulatives qui peuvent être fournies dans des attributs, dans des notifications, dans des réponses d'action ou dans une combinaison de ces messages. Les informations récapitulatives peuvent consister en valeurs d'attribut observées ou en statistiques calculées sur la base de ces valeurs (soit selon la durée, soit selon les objets gérés).

Les valeurs d'attribut observées sont extraites lors d'un "relevé" déclenché périodiquement à la fin de chaque période de granularité, à condition que cette période ne soit pas nulle.

L'attribut de période de granularité indique la longueur de cette période. Celle-ci ne doit pas être modifiée dans l'objet releveur à moins que la valeur de l'état administratif ne soit "verrouillé". Si le paquetage de synchronisation de période n'est pas présent, le moment auquel la période de granularité commence une fois le releveur déverrouillé est une question locale.

L'attribut d'état administratif sert à suspendre ou à reprendre la fonction de relevé. Si l'état administratif possède la valeur "déverrouillé", le releveur a l'autorisation administrative d'effectuer des relevés. Si l'état administratif possède la valeur "verrouillé", le releveur a l'interdiction administrative d'effectuer des relevés.

L'attribut d'état opérationnel représente la capacité opérationnelle du releveur de remplir ses fonctions.

Si le paquetage de notification de changement de valeur d'attribut est présent, les modifications de la période de granularité ou de l'instant de synchronisation de période doivent provoquer l'émission de notifications de changement de valeur d'attribut. Si le paquetage de notification de changement d'état est présent, les modifications de l'état opérationnel ou de l'état administratif doivent provoquer l'émission de notifications de changement d'état. Si le paquetage de notification de création/suppression d'objet est présent, la création ou la suppression d'un objet contenu dans la sous-classe du releveur doit émettre la notification de création ou de suppression correspondante.

Les attributs suivants sont définis dans l'entité:

- *administrativeState* – Cet attribut sert à activer/désactiver (suspendre et reprendre administrativement) la fonction remplie par le releveur;
- *granularityPeriod* – Cet attribut spécifie la durée pendant laquelle le "relevé" est effectué;
- *operationalState* – Cet attribut indique si la fonction de relevage représentée par cet objet possède ou ne possède pas la capacité de remplir ses fonctions normales.

Le paquetage de synchronisation de période est présent si une synchronisation interne des périodes à répétition d'intervalle est requise par un agent configurable.

L'attribut suivant est défini dans le paquetage de synchronisation de période:

- *periodSynchronizationTime* – Cet attribut indique le moment auquel une période de répétition est synchronisée. Le début de chaque période est un nombre entier de périodes avant ou après le moment spécifié par cet attribut.

L'interface *Scanner* n'est pas instanciable.

4.3.2 Interface *CurrentData*

4.3.2.1 Classe CurrentData

La classe d'objets CurrentData, héritée du releveur, contient des objets gérés de support qui enregistrent les données actuelles de qualité de fonctionnement aux fins de la surveillance. Les données de qualité de fonctionnement (mesures) des ressources surveillées sont modélisées par des attributs dans la définition des sous-classes de la classe CurrentData, car celle-ci ne peut pas être instanciée. Les objets qui représentent des ressources surveillées contiennent les objets CurrentData correspondants dans la relation de corrélation de noms.

Les mesures sont collectées pendant un intervalle (par exemple 5 minutes) spécifié par l'attribut granularityTime. A la fin de chaque intervalle, l'attribut elapsedTime sera mis à jour en fonction de la différence entre le temps actuel et le début de l'intervalle de surveillance présent. Les mesures, granularityTime et suspectIntervalFlag sont copiés dans une structure de données chronologiques correspondantes (représentée par la classe HistoryDataValueType). Par ailleurs, l'instant periodEndTime de la structure de données chronologiques est attribué au temps actuel.

Si la valeur de l'attribut historyRetention est supérieur à zéro, les structures des données chronologiques seront conservées dans le système géré pendant au moins la durée qui équivant au nombre d'intervalles spécifiés dans l'attribut historyRetention. Pendant ce temps, les données chronologiques pourront être consultées au moyen d'une série d'opérations comme suit:

- *getMostRecent* – Cette opération extrait les données chronologiques les plus récentes, étant entendu par là qu'il s'agit des plus récentes données chronologiques sauvegardées, qui peuvent être situées à plusieurs intervalles avant l'instant actuel parce que la rétention chronologique peut être réglée à zéro ou parce que toutes les mesures relevées sont égales à zéro. Il est possible que l'opération getMostRecent ne soit pas en mesure de récupérer des données. Par exemple, toutes les mesures sont représentées par des zéros (aucune donnée chronologique n'a été sauvegardée) ou l'attribut historyRetention a été mis à zéro. La valeur booléenne renvoyée par la méthode getMostRecent indique la disponibilité des données. La valeur "Vrai" signifie que le paramètre de sortie contient les données et la valeur "Faux" signifie que le paramètre n'est pas défini;
- *getBetween* – Cette opération extrait un ensemble de données chronologiques à l'intérieur d'une fenêtre temporelle spécifiée par les paramètres de temps de début et de fin. Le paramètre resultIterator est utilisé pour transférer de grandes tranches de données. Si le paramètre howMany n'est pas fourni, la quantité initiale de données renvoyées est une question locale.

Les attributs suivants sont définis dans l'entité:

- *suspectIntervalFlag* – Cet attribut sert à indiquer que les données de qualité de fonctionnement pour la période actuelle ne sont peut-être pas fiables, pour des raisons comme les suivantes:
  - détection de données suspectes par la ressource même qui effectue l'acquisition des données;
  - transition de l'attribut administrativeState à destination ou en provenance de l'état "verrouillé";
  - transition de l'attribut operationalState à destination ou en provenance de l'état "désactivé";
  - réinitialisation des compteurs de qualité de fonctionnement au cours de l'intervalle;
  - création de l'instance d'objet CurrentData (ou de sa sous-classe) au cours de la période de surveillance;
- *elapsedTime* – Cet attribut représente la différence entre le temps actuel et le début de l'intervalle de surveillance en cours;
- *historyRetention* – Cet attribut spécifie le nombre minimal d'intervalles pendant lesquels la structure des données chronologiques doit être conservée.

Si le paramètre thresholdPackage est présent, l'objet CurrentData contient au moins un pointeur sur un objet de la classe ThresholdData. Si un des seuils (définis dans l'objet ThresholdData référencé) est franchi, une notification d'alarme de qualité de service (QS) est émise par l'objet CurrentData. Le champ thresholdInfo de cette alarme doit contenir l'attribut de la mesure qui viole le seuil. Si d'autre(s) franchissement(s) de seuil se produisent au cours de l'intervalle en cours, des alarmes additionnelles doivent être également émises.

L'attribut thresholdDataInstanceList contient un ensemble de pointeurs sur des objets de la classe ThresholdData, spécifiés par système(s) gérant(s). Plusieurs notifications d'alarme QS peuvent donc être émises pour un même attribut surveillé. Il est également possible que les notifications soient en conflit. Par exemple, une notification indique qu'une alarme est activée tandis qu'une autre notification indique que cette alarme est désactivée: c'est au système gérant qu'il appartient de conserver la cohérence des notifications.

Les nouveaux seuils résultant de la modification de l'attribut thresholdDataInstanceList ou de la modification d'une valeur de seuil dans l'objet ThresholdData référencé, devraient prendre effet immédiatement. Si une situation d'alarme existe avant l'apparition d'une modification de valeur de seuil (c'est-à-dire qu'un ancien seuil a été franchi) et que la nouvelle valeur de seuil soit extérieure à l'étendue de l'ancienne valeur de seuil (par exemple, dans le cas d'un compteur croissant, la nouvelle valeur de seuil est supérieure à la précédente) et que la valeur actuelle de la mesure soit dans l'étendue admissible de la nouvelle valeur de seuil, une notification d'alarme de QS est émise avec une sévérité de valeur "désactivée". Si la nouvelle valeur de seuil est réglée dans l'étendue de l'ancienne valeur de seuil, de façon que le nouveau seuil soit franchi, une notification d'alarme de QS est émise si une situation d'alarme n'est pas déjà en instance.

L'attribut suivant est défini dans le paquetage de seuil thresholdPackage:

- *thresholdDataInstanceList* – Attribut de type "list" dans lequel chaque élément est un pointeur sur un objet de la classe ThresholdData qui contient les seuils ou limites des paramètres de qualité de fonctionnement.

Le zeroSuppressionPackage est un paquetage conditionnel de la classe CurrentData. Lorsqu'il est présent et qu'un intervalle se termine par des mesures de qualité "tout en zéros", aucune structure de données chronologiques n'est créée.

Les attributs suivants sont définis dans le paquetage zeroSuppressionPackage:

- *numSuppressedIntervals* – Cet attribut est utilisé pour compter le nombre d'intervalles consécutifs pendant lesquels une suppression (c'est-à-dire la non-création d'une structure de données chronologiques) s'est produite. Cet attribut reflète les mesures de qualité jusqu'à l'intervalle actuel, non compris. Cet attribut est incrémenté à la fin d'un intervalle si une suppression s'est produite. Si ce n'est pas le cas, l'attribut est réinitialisé. Si c'est le cas et s'il atteint la valeur du paramètre maxSuppressedIntervals, un enregistrement de données chronologiques est créé et l'attribut sera réinitialisé.
- *maxSuppressedIntervals* – Cet attribut limite le nombre maximal d'intervalles supprimés qui seront collectés sans création d'une structure des données chronologiques. La valeur par défaut –1 signifie qu'il n'y a aucune limite quant au nombre d'intervalles consécutifs supprimés. Par ailleurs, l'attribut maxSuppressedIntervals est effectivement égal à l'infini.

Soit par exemple une instance (d'une sous-classe de la classe) CurrentData concernant la suppression de zéros avec l'attribut maxSuppressedIntervals réglé à 32, et l'intervalle réglé à 15 minutes. Pour la compression des enregistrements, cela signifie qu'après 32 intervalles consécutifs (soit 8 heures) supprimés (par exemple tout en zéros), au moins un enregistrement de données chronologiques (avec les paramètres PM tout en zéros) sera produit avec un décompte de 32. Cela garantit qu'au moins un enregistrement de données chronologiques sera créé pour chaque attribut maxSuppressedIntervals.

L'interface CurrentData n'est pas instanciable.

4.3.2.2 Classe HistoryData

La classe HistoryData, représentée par la classe HistoryDataValueType, contiendra une copie des mesures de qualité de fonctionnement et d'autres attributs sélectionnés qui sont présents dans un objet de données actuelles à la fin de l'intervalle actuel (par exemple 5 minutes). Un nouvel enregistrement de cette structure de type de valeur est créé à la fin de chaque intervalle si l'attribut historyRetention, contenu dans l'objet de données actuelles, est supérieur à zéro. Cet enregistrement sera ensuite conservé dans l'élément de réseau pendant au moins une durée équivalente au nombre d'intervalles spécifié dans l'attribut historyRetention.

Ce type de valeur générique modélise les données chronologiques. Des objets spécifiques de données actuelles (par exemple des objets de données actuelles SDH/SONET) seront définis avec des types de valeur chronologique spécifiques, qui seront hérités de ce type de valeur générique HistoryValuetype.

Les attributs suivants sont définis dans l'entité:

- *periodEndTime* – Cet attribut indique l'instant de fin de l'intervalle.
- *granularityPeriod* – Cet attribut sert à copier le même attribut issu de l'objet CurrentData.
- *suspectIntervalFlag* – Cet attribut sert à copier le même attribut issu de l'objet CurrentData.
- *numSuppressedIntervals* – Cet attribut indique le nombre d'intervalles qui ont été supprimés avant la création de l'enregistrement des types de valeurs chronologiques considéré. Sa valeur par défaut est 0. Lors de la copie à partir de l'objet CurrentData, la valeur par défaut est utilisée. Lors de la copie à partir de l'objet CurrentData avec le paquetage ZeroSuppressionPackage, la valeur de l'attribut correspondant est utilisée.

4.3.3 Interface *ThresholdData*

Les objets de la classe ThresholdData sont des objets gérés de support qui contiennent les valeurs de réglage des seuils pour les paramètres PM. Au moins un des paquetages counterThresholdListPackage ou gaugeThresholdListPackage doit être instancié.

Les seuils sont établis au moyen des objets gérés de la classe ThresholdData, qui spécifient les seuils à appliquer. Les valeurs de seuil contenues dans un objet ThresholdData peuvent s'appliquer à de multiples objets CurrentData, lesquels pointent sur les objets ThresholdData. Les objets de la classe CurrentData (ou de ses sous-classes) indiquent l'intervalle d'acquisition utilisé avec le seuil. Une notification de type qualityOfServiceAlarm est produite chaque fois qu'un seuil est franchi.

Un seuil peut être propre à un même objet géré. Dans ce cas, l'objet ThresholdData est contenu dans l'objet géré qui est observé. Un objet CurrentData contenu dans l'objet géré pointe sur un autre objet ThresholdData. Ce lien est nécessaire afin d'assurer la cohérence avec le cas d'objets gérés multiples.

Il est parfois souhaitable qu'un seuil s'applique à un groupe d'objets gérés. Dans ce cas, l'objet ThresholdData est extérieur à l'objet géré qui est observé. Il peut être contenu dans l'objet de la classe ManagedElement. L'objet ThresholdData fait l'objet d'un pointage par tous les objets CurrentData auxquels il s'applique.

Si le paquetage createDeleteNotificationsPackage est présent, la création ou la suppression d'un objet de seuil provoquera l'émission d'une notification de type objectCreation ou objectDeletion.

Si le paquetage attributeValueChangeNotificationPackage est présent, toute modification de seuil provoquera l'émission d'une notification de type attributeValueChange.

Le paquetage counterThresholdListPackage est présent si une instance le prend en charge et si le paquetage gaugeThresholdListPackage n'est pas présent.

L'attribut suivant est défini dans le paquetage counterThresholdListPackage:

- *counterThresholdList* – Cet attribut contient un ensemble de réglages de seuil pour les attributs de qualité de fonctionnement de type compteur (par exemple de secondes erronées). Chaque réglage de seuil se compose de l'identificateur d'attribut, de la valeur de seuil et (facultativement) de la sévérité de l'événement de dépassement de seuil.

Le paquetage gaugeThresholdListPackage est présent si une instance le prend en charge et que le paquetage counterThresholdListPackage ne soit pas présent.

L'attribut suivant est défini dans le paquetage gaugeThresholdListPackage:

- *gaugeThresholdList* – Cet attribut contient un ensemble de réglages de seuil pour les attributs de qualité de fonctionnement de type jauge. Chaque réglage de seuil se compose de l'identificateur d'attribut, de la valeur de seuil et (facultativement) de la sévérité de l'événement de dépassement de seuil.

4.3.4 Interface *HistoryDataFile*

Les objets HistoryDataFile forment une sous-classe de la classe ManagedObject. Ils s'appliquent à chaque fichier de données chronologiques, qui est donc associé à un seul objet de la classe HistoryDataFile. Chaque objet HistoryDataFile contient un fichier de données chronologiques sous-jacent. L'objet HistoryDataFile n'est créé qu'après la production du fichier de données chronologiques. Il est supprimé dès que le fichier de données chronologiques est purgé.

Le système géré doit commander le cycle de vie des objets HistoryDataFile. Un comportement de destruction, selon lequel le fichier de données chronologiques est purgé si l'objet HistoryDataFile est supprimé, est prévu si le système gérant a l'intention de purger le fichier de données chronologiques avant l'expiration de la période de conservation de fichier.

L'objet HistoryDataFile possède les attributs suivants:

- *fileLocation* – Cet attribut indique l'emplacement du fichier en format de localisation URL. Le temps qui s'écoule entre la réutilisation du même emplacement de fichier pour deux fichiers différents doit être suffisamment au-delà de toute possibilité de confusion;
- *fileFormat* – Cet attribut indique le format utilisé pour le fichier;
- *periodEndTime* – Cet attribut indique l'instant de fin de la production de fichier;
- *fileRetentionDuration* – Cet attribut indique la période de conservation de fichier qui s'écoule avant que le fichier soit purgé en raison d'une acquisition d'informations superflues par le système géré.

Lorsqu'un fichier de données chronologiques est produit et qu'un objet HistoryDataFile est créé, une notification objectCreation est envoyée par le système géré afin d'informer le système gérant de la disponibilité du fichier de données chronologiques. Lorsqu'un fichier de données chronologiques est supprimé et que l'objet HistoryDataFile associé est supprimé, une notification objectDeletion peut être envoyée par le système géré au système gérant si le paquetage objectDeleteNotificationPackage est pris en charge.

Même s'il existe un objet HistoryDataFileFactory, un objet HistoryDataFile ne doit jamais être créé par le système gérant. L'objet HistoryDataFileFactory est censé être utilisé par le système géré.

4.3.5 Interface *AbstractHistoryDataScanner*

Les objets AbstractHistoryDataScanner forment une sous-classe de la classe Scanner et une superclasse de tous les autres releveurs de données chronologiques. Ils recueillent les attributs et opérations communs des releveurs de données chronologiques. L'interface AbstractHistoryDataScanner n'est pas instanciable.

Les releveurs de données chronologiques assurent le relevé et la signalisation des données chronologiques tout en offrant la capacité de commander la production de fichiers de données chronologiques, la sélection du format du fichier, et la notification des utilisateurs lors de la production d'un fichier.

L'objet AbstractHistoryDataScanner possède les attributs suivants:

- *supportedFileFormatList* – Cet attribut indique les formats de fichier pris en charge par le système géré étant donné qu'il peut prendre en charge plusieurs formats de fichier;
- *hdFileFormat* – Cet attribut indique le format de fichier utilisé;
- *fileRetentionDuration* – Cet attribut indique la période de conservation de tous les fichiers de données chronologiques produits par le releveur considéré;
- *beginRelativeTime* – Cet attribut indique l'instant de début par rapport au temps actuel ainsi que l'instant relatif de fin définissant une fenêtre de sélection de données chronologiques. La valeur négative de cet attribut indique un instant du passé;

- *endRelativeTime* – Cet attribut indique l'instant de fin par rapport au temps actuel. La valeur négative de cet attribut indique un instant du passé. La valeur zéro indique l'absence d'instant de fin, ce qui se traduira par une production périodique de fichier jusqu'à ce qu'elle soit arrêtée par le système gérant;
- *granularityPeriod (attribut hérité de la classe Scanner)* – Cet attribut indique une fenêtre temporelle telle que les données chronologiques et leurs marqueurs temporels s'inscrivant dans cette fenêtre soient signalés dans un fichier.

Avant de fixer d'éventuelles valeurs dans ces attributs, le paramètre administrativeState doit être mis à la valeur "verrouillé"; lorsqu'il est mis à la valeur "déverrouillé", la production périodique de fichier commence. Si la période de granularité est égale à zéro, il n'y a pas de production périodique de fichier.

A la fin de chaque période de granularité du releveur de données chronologiques, celui-ci relève ces données à partir d'un ensemble d'objets de données actuelles spécifiés dans le releveur détaillé, collecte d'éventuelles données si l'instant de fin de sa période s'inscrit dans l'intervalle temporel de la granularité actuelle, puis produit un fichier de données dans un format indiqué par l'attribut hdFileFormat. Lorsque ce fichier est produit, un objet HistoryDataFile est créé par le système géré et une notification objectCreation est envoyée au système gérant par l'objet HistoryDataFile qui vient d'être créé.

Le releveur de données chronologiques comporte deux opérations pour la signalisation non périodique des données chronologiques:

- *returnHDReport* – Cette opération renvoie une séquence de types de valeur de données chronologiques dont l'instant de fin de période s'inscrit dans l'instant actuel et la fin de la dernière période de granularité du releveur;
- *generateHDFileNow* – Cette opération produit immédiatement un fichier de données chronologiques au lieu d'attendre la fin de la prochaine période de granularité du releveur. L'instant de fin de période du type valeur de données chronologiques indiqué dans ce fichier doit être compris entre l'instant actuel et la fin de la dernière période de granularité.

4.3.6 Interface *ListHistoryDataScanner*

L'interface ListHistoryDataScanner est un releveur spécialisé de données chronologiques et une sous-classe de la classe AbstractHistoryDataScanner. Elle renvoie des rapports et des fichiers de données chronologiques sur la base d'une simple liste d'objets.

L'objet ListHistoryDataScanner possède l'attribut suivant:

- *cdInstancesList* – Cet attribut contient un ensemble d'objets de données actuelles dont les données chronologiques doivent être relevées/signalées dans un fichier.

La Figure 6 décrit le simple cas d'utilisation de l'objet ListHistoryDataScanner.

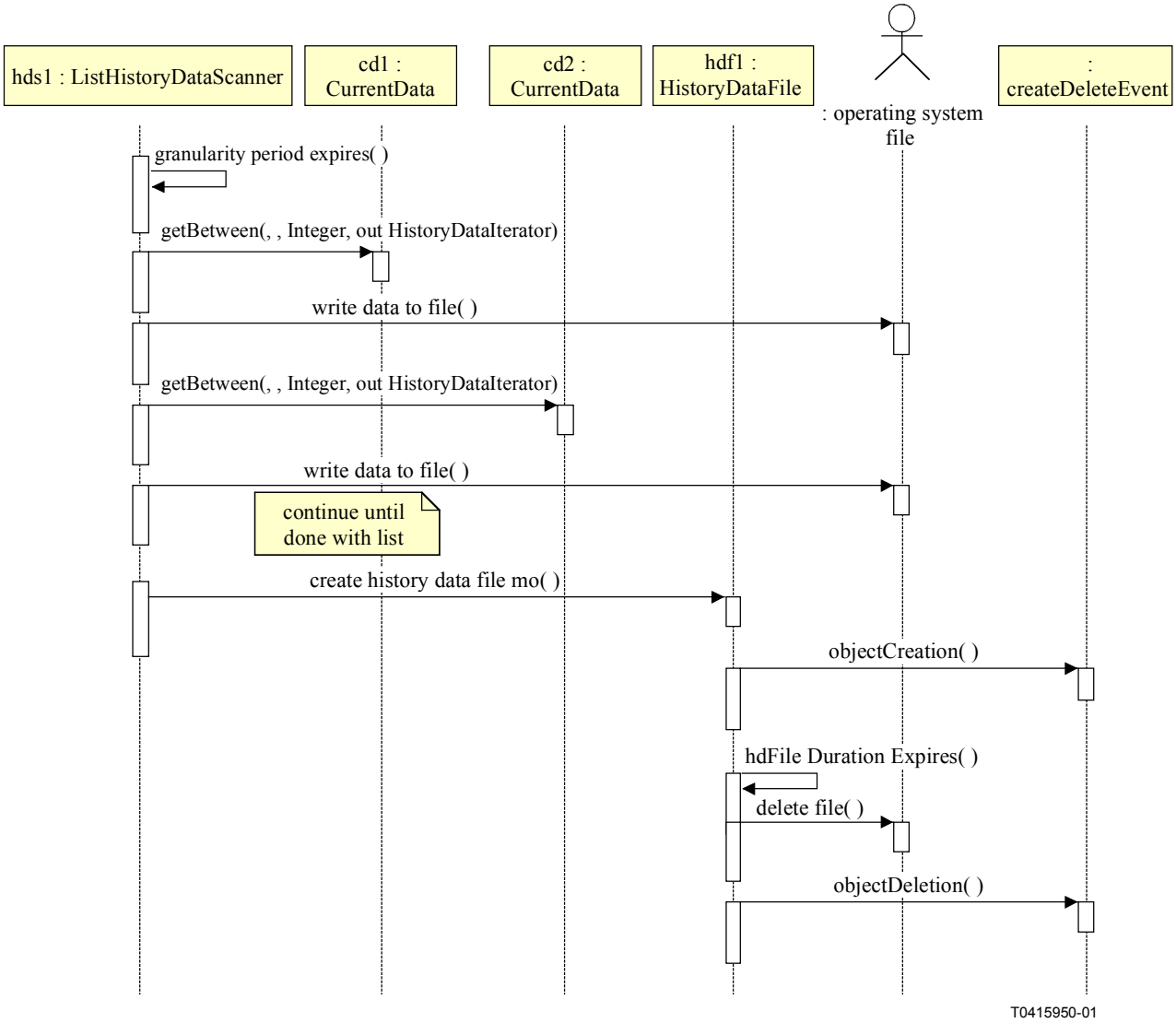

Figure 6/Q.822.1 – Cas d'utilisation de l'interface ListHistoryDataScanner

4.3.7 Interface *ClassHistoryDataScanner*

Les objets ClassHistoryDataScanner représentent un releveur spécialisé de données chronologiques et forment une sous-classe de la classe AbstractHistoryDataScanner. Ils renvoient des rapports et des fichiers de données chronologiques sur la base d'une liste de classes d'objets.

L'objet ClassHistoryDataScanner possède l'attribut suivant:

- *cdClassSelectionList* – Cet attribut contient un ensemble de classes de données actuelles groupées par l'objet géré qui les contient (par exemple l'équipement de réseau). Les objets de données actuelles dont il y a lieu de relever/signaler les données chronologiques dans un fichier sont les instances spécifiées des classes de données actuelles contenues dans l'objet géré correspondant.

Avant de produire un rapport ou fichier nouveau, l'objet ClassHistoryDataScanner doit réévaluer les instances de données actuelles qui concordent avec d'éventuelles instances de données actuelles ajoutées ou supprimées depuis le dernier relevé.

4.3.8 Interface *ScopedFilteredHistoryDataScanner*

Les objets ScopedFilteredHistoryDataScanner représentent un releveur spécialisé de données chronologiques et forment une sous-classe de la classe AbstractHistoryDataScanner. Ils renvoient des rapports et des fichiers de données chronologiques sur la base d'une détection de domaine et d'un filtrage.

L'objet ScopedFilteredHistoryDataScanner possède les attributs suivants:

- *base* – objet géré de base qui est utilisé pour la détection de domaine;
- *scope* – valeur de détection définie dans la Rec. UIT-T Q.816;
- *filter* – expression de filtre à contrainte qui sert à évaluer les objets concordants;
- *language* – chaîne indiquant le langage dans lequel l'expression de filtre est écrite.

Les objets de données actuelles dont les données chronologiques doivent être relevées/signalées dans un fichier sont ceux qui ont été sélectionnés par une opération de détection et filtrage. Avant de produire un rapport ou fichier nouveau, l'objet ScopedFilteredHistoryDataScanner doit réappliquer l'opération de détection et filtrage afin de sélectionner le plus récent ensemble d'objets de données actuelles.

5 Langage IDL du service de gestion de qualité de fonctionnement

```
#ifndef _itut_q822_1_idl_
#define _itut_q822_1_idl_


#include <itut_x780.idl>
#include <itut_x780_1.idl>
#include <itut_x780ct.idl>
#include <itut_q816.idl>
#include <itut_q816_1.idl>
#include <itut_m3120.idl>


#pragma prefix "itu.int"


/**
This IDL code is intended to be stored in a file named "itut_q822_1.idl"
located in the search path used by IDL compilers installed on your system.
*/


/**
This module, itut_q822d1, contains IDL definition based on objects defined
in ITU-T Rec. Q.822. The IDL definitions in this file are the object interfaces.
*/
module itut_q822d1
{


/**
```

5.1 Importations

```
*/

/**
Types imported from itut_x780
*/
     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::DeletePolicyType DeletePolicyType;
     typedef itut_x780::GeneralizedTimeType GeneralizedTimeType;
     typedef itut_x780::Istring Istring;
     typedef itut_x780::IstringSetType IstringSetType;
     typedef itut_x780::MONameType MONameType;
     typedef itut_x780::MONameSetType MONameSetType;
     typedef itut_x780::NameBindingType NameBindingType;
     typedef itut_x780::OperationalStateType OperationalStateType;
     typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
     typedef itut_x780::ScopedNameSetType ScopedNameSetType;
     typedef itut_x780::UIDType UIDType;


/**
Types imported from itut_x780ct
*/
     typedef itut_x780ct::SeverityIndicatingThresholdType
          SeverityIndicatingThresholdType;
     typedef itut_x780ct::SeverityIndicatingGaugeThresholdSetType
          SeverityIndicatingGaugeThresholdSetType;
     typedef itut_x780ct::TimeIntervalType TimeIntervalType;
     typedef itut_x780ct::TimePeriodType TimePeriodType;


/**
Types imported from itut_q816
*/
     typedef itut_q816::ScopeType ScopeType;
     typedef itut_q816::FilterType FilterType;
     typedef itut_q816::LanguageType LanguageType;


/**
Types imported from itut_m3120
*/
     typedef itut_m3120::ManagedElementNameType ManagedElementNameType;
/**
Interfaces imported from itut_x780

itut_x780::ManagedObject
itut_x780::ManagedObjectFactory
*/


/**
```

5.2 Déclarations anticipées

```
*/

/**
Interface forward declarations
*/
     interface Scanner;
     interface HistoryDataIterator;
     interface CurrentData;
     interface ThresholdData;
     interface HDFile;
     interface AbstractHDScanner;
     interface ListHDScanner;
     interface ClassHDScanner;
     interface ScopedFiltedHDScanner;


/**
Valuetype forward declarations
*/
     valuetype ScannerValueType;
     valuetype CurrentDataValueType;
     valuetype HistoryDataValueType;
     valuetype ThresholdDataValueType;
     valuetype HDFileValueType;
     valuetype AbstractHDScannerValueType;
     valuetype ListHDScannerValueType;
     valuetype ClassHDScannerValueType;
     valuetype ScopedFiltedHDScannerValueType;


/**
Interface forward declarations
*/
     typedef MONameType ScannerNameType;
     typedef MONameType CurrentDataNameType;
     typedef MONameType ThresholdDataNameType;
     typedef MONameType HDFileNameType;
     typedef MONameType AbstractHDScannerNameType;
     typedef MONameType ListHDScannerNameType;
     typedef MONameType ClassHDScannerNameType;
     typedef MONameType ScopedFiltedHDScannerNameType;


/**
```

5.3 Structures et définitions de type

```
*/

/**
This data type defines a sequence of HistoryDataValueType. The order is
significant.
*/
     typedef sequence <HistoryDataValueType> HistoryDataSeqValueType;

     typedef sequence <CurrentDataNameType> CurrentDataNameSetType;

     typedef UIDType HDFileFormatType;
```

```
     typedef sequence <HDFileFormatType> HDFileFormatSetType;

     typedef Istring URLType;

     struct CounterThresholdSettingType
     {
          string                              attributeName;
          SeverityIndicatingThresholdType  threshold;
     };

/**
The order is insignificant.
*/
     typedef sequence <CounterThresholdSettingType> CounterThresholdSetType;

     struct GaugeThresholdSettingType
     {
          string                         attributeName;
          SeverityIndicatingGaugeThresholdSetType threshold;
     };

/**
The order is insignificant.
*/
     typedef sequence<GaugeThresholdSettingType>
          GaugeThresholdSetType;

/**
This data structure represents all the CurrentData classes contained by a
containing object through either direct or indirect containment.
*/
     struct CDClassSelectionType
     {
          MONameType         containingObject;
          ScopedNameSetType       cdClassList;
     };

     typedef sequence<CDClassSelectionType> CDClassSelectionSetType;


/**
```

5.4 Exceptions

```
*/

/**
Exceptions and Constants for Conditional Package
*/
     exception NOcounterThresholdListPackage {};

     exception NOdeleteNotificationPackage {};

     exception NOgaugeThresholdListPackage {};

     exception NOperiodSynchronizationPackage {};

     exception NOthresholdPackage {};


     const string counterThresholdListPackage =
          "itut_q822d1::counterThresholdListPackage";
     const string deleteNotificationPackage =
          "itut_q822d1::deleteNotificationPackage";
```

```
     const string gaugeThresholdListPackage =
          "itut_q822d1::gaugeThresholdListPackage";
     const string periodSynchronizationPackage =
          "itut_q822d1::periodSynchronizationPackage";
     const string thresholdPackage =
          "itut_q822d1::thresholdPackage";

/**
Exceptions for Performance Management
*/

/**
```

5.5 Interfaces à granularité fine

```
*/

/**
```

5.5.1 Interface *Scanner*

```
Pour le comportement, voir le paragraphe 4.3.1.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants.

- periodSynchronizationPackage: présent si une synchronisation interne par agent
configurable est requise pour les périodes répétitives.

L'interface Scanner n'est pas instanciable.
*/
     valuetype ScannerValueType: itut_x780::ManagedObjectValueType
     {
          public AdministrativeStateType   administrativeState;
               // GET-REPLACE
          public OperationalStateType      operationalState;
               // GET
          public TimePeriodType            granularityPeriod;
               // GET-REPLACE
          public GeneralizedTimeType       periodSynchronizationTime;
               // conditional
               // periodSynchronizationPackage
               // GET-REPLACE
     }; // valuetype ScannerValueType


     interface Scanner: itut_x780::ManagedObject
     {
          AdministrativeStateType administrativeStateGet ()
               raises (itut_x780::ApplicationError);

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

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

          TimePeriodType granularityPeriodGet ()
               raises (itut_x780::ApplicationError);

          void granularityPeriodSet
               (in TimePeriodType granularityPeriod)
               raises (itut_x780::ApplicationError);
```

```
        GeneralizedTimeType periodSynchronizationTimeGet ()
             raises (itut_x780::ApplicationError,
                  NOperiodSynchronizationPackage);

        void periodSynchronizationTimeSet
             (in GeneralizedTimeType periodSyncTime)
             raises (itut_x780::ApplicationError,
                  NOperiodSynchronizationPackage);

        CONDITIONAL_NOTIFICATION(
             itut_x780::Notifications, objectCreation,
             createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
             itut_x780::Notifications, objectDeletion,
             createDeleteNotificationsPackage)
        CONDITIONAL_NOTIFICATION(
             itut_x780::Notifications, attributeValueChange,
             attributeValueChangeNotificationPackage)
        CONDITIONAL_NOTIFICATION(
             itut_x780::Notifications, stateChange,
             stateChangeNotificationPackage)

    };  // interface Scanner


/**
```

5.5.2 Interface *CurrentData*

Voir le paragraphe 4.3.2.1 pour le comportement.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants.

- thresholdPackage: présent si une notification d'alarme de qualité de service doit être émise en raison d'un franchissement de seuil.

- zeroSupressionPackage: présent si une instance le prend en charge.

L'interface CurrentData n'est pas instanciable.

```
*/

/**
```

5.5.2.1 Classe CurrentDataValueType

```
*/
    valuetype CurrentDataValueType: ScannerValueType
    {
        public boolean              suspectIntervalFlag;
             // GET REPLACE-WITH-DEFAULT
        public TimeIntervalType     elapsedTime;
             // GET
        public short                historyRetention;
             // Mandatory now (used to be conditional)
             // GET-REPLACE
        public MONameSetType        thresholdDataInstanceList;
             // conditional
             // thresholdPackage
             // GET-REPLACE ADD-REMOVE
        public short                numSuppressedIntervals;
             // conditional
             // zeroSuppressionPackage
             // GET
```

```
          public short                maxSuppressedIntervals;
               // conditional
               // zeroSuppressionPackage
               // GET-REPLACE
     }; // valuetype CurrentDataValueType


/**
```

5.5.2.2 Classe HistoryDataValueType

```
Voir le paragraphe 4.3.2.2 pour le comportement.
*/
     valuetype HistoryDataValueType
     {
          public GeneralizedTimeType  periodEndTime;
               // GET
          public TimePeriodType       granularityPeriod;
               // GET
          public boolean              suspectIntervalFlag;
               // GET
          public short                numSupressedIntevals;
               // GET
     }; // valuetype HistoryDataValueType


/**
```

5.5.2.3 Classe HistoryDataIterator

```
La classe HistoryDataIterator est utilisée pour transférer de grandes tranches de
données chronologiques.
*/
     interface HistoryDataIterator
     {
/**
This method is used to return the next howMany history data. howMany is the
maximum number of history data for which results should be returned in the first
batch; it must be non-zero. historyDataList is a sequence of history data
records.
The method will return true if there is data in out parameter, false otherwise.
*/
          boolean getNext
               (in unsigned short howMany,
               out HistoryDataSeqValueType historyDataList)
               raises (itut_x780::ApplicationError);

/**
This method is used to destroy the iterator and release its resources. This must
be done by the application.
*/
          void destroy ();

     };  // interface HistoryDataIterator


/**
```

5.5.2.4 Classe CurrentData

```
*/
     interface CurrentData: Scanner
     {
/**
Define default value for suspectIntervalFlag
*/
          const boolean suspectIntervalFlagDefault = FALSE;

/**
Define default value for maxSuppressedIntervals. The default value -1 means that
there is no limit on the number of consecutive suppressed intervals. On the other
hand, the maxSuppressedIntervals is effectively equal to infinity.
*/
          const short maxSuppressedIntervalsDefault = -1;

          boolean suspectIntervalFlagGet ()
               raises (itut_x780::ApplicationError);

          void suspectIntervalFlagDefaultSet
               (in boolean suspectIntervalFlag)
               raises (itut_x780::ApplicationError);

          TimeIntervalType elapsedTimeGet ()
               raises (itut_x780::ApplicationError);

          short historyRetentionGet ()
               raises (itut_x780::ApplicationError);

          void historyRetentionSet
               (in short intervalNum)
               raises (itut_x780::ApplicationError);

          MONameSetType thresholdDataInstanceListGet ()
               raises (itut_x780::ApplicationError,
                    NOthresholdPackage);

          void thresholdDataInstanceListSet
               (in MONameSetType dataInstanceList)
               raises (itut_x780::ApplicationError,
                    NOthresholdPackage);

          void thresholdDataInstanceListAdd
               (in MONameSetType dataInstanceList)
               raises (itut_x780::ApplicationError,
                    NOthresholdPackage);

          void thresholdDataInstanceListRemove
               (in MONameSetType dataInstanceList)
               raises (itut_x780::ApplicationError,
                    NOthresholdPackage);

          boolean getMostRecent
               (out HistoryDataValueType historyData)
               raises (itut_x780::ApplicationError);
```

```
          HistoryDataSeqValueType getBetween
               (in GeneralizedTimeType startTime,
               in GeneralizedTimeType endTime,
               in unsigned short howMany,
               out HistoryDataIterator resultIterator)
               raises (itut_x780::ApplicationError);

/**
*/
          short numSuppressedIntervalsGet ()
               raises (itut_x780::ApplicationError);

          short maxSuppressedIntervalsGet ()
               raises (itut_x780::ApplicationError);

          void maxSuppressedIntervalsSet
               (in short maxSuppInterval)
               raises (itut_x780::ApplicationError);

          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, qualityOfServiceAlarm,
               NOthresholdPackage)

     };  // interface CurrentData


/**
```

5.5.3 ThresholdData

Voir le paragraphe 4.3.3 pour le comportement.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants.

- counterThresholdListPackage: présent si une instance le prend en charge et si le paquetage gaugeThresholdListPackage n'est pas présent.

- gaugeThresholdListPackage: présent si une instance le prend en charge et si le paquetage counterThresholdListPackage n'est pas présent.

```
*/
     valuetype ThresholdDataValueType: itut_x780::ManagedObjectValueType
     {
          public CounterThresholdSetType   counterThresholdList;
               // conditional
               // counterThresholdListPackage
               // GET-REPLACE ADD-REMOVE
          public GaugeThresholdSetType gaugeThresholdList;
               // conditional
               // gaugeThresholdListPackage
               // GET-REPLACE ADD-REMOVE
     }; // valuetype ThresholdDataValueType


     interface ThresholdData: itut_x780::ManagedObject
     {
          CounterThresholdSetType counterThresholdListGet ()
               raises (itut_x780::ApplicationError,
                    NOcounterThresholdListPackage);

          void counterThresholdListSet
               (in CounterThresholdSetType counterThresholdList)
               raises (itut_x780::ApplicationError,
                    NOcounterThresholdListPackage);
```

```
     void counterThresholdListAdd
          (in CounterThresholdSetType counterThresholdList)
          raises (itut_x780::ApplicationError,
               NOcounterThresholdListPackage);

     void counterThresholdListRemove
          (in CounterThresholdSetType counterThresholdList)
          raises (itut_x780::ApplicationError,
               NOcounterThresholdListPackage);

     GaugeThresholdSetType gaugeThresholdListGet ()
          raises (itut_x780::ApplicationError,
               NOgaugeThresholdListPackage);

     void gaugeThresholdListSet
          (in GaugeThresholdSetType gaugeThresholdList)
          raises (itut_x780::ApplicationError,
               NOgaugeThresholdListPackage);

     void gaugeThresholdListAdd
          (in GaugeThresholdSetType gaugeThresholdList)
          raises (itut_x780::ApplicationError,
               NOgaugeThresholdListPackage);

     void gaugeThresholdListRemove
          (in GaugeThresholdSetType gaugeThresholdList)
          raises (itut_x780::ApplicationError,
               NOgaugeThresholdListPackage);

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

};  // interface ThresholdData


interface ThresholdDataFactory: itut_x780::ManagedObjectFactory
{
     itut_x780::ManagedObject create
          (in NameBindingType nameBinding,
          in MONameType superior,
          in string reqID,   // auto naming if null
          out MONameType name,
          in IstringSetType packageNameList,
          in CounterThresholdSetType counterThresholdList,
               // conditional
               // counterThresholdListPackage
               // GET-REPLACE ADD-REMOVE
          in GaugeThresholdSetType gaugeThresholdList)
               // conditional
               // gaugeThresholdListPackage
               // GET-REPLACE ADD-REMOVE
          raises (itut_x780::ApplicationError,
               itut_x780::CreateError);
```

```
     }; // interface ThresholdDataFactory


/**
```

5.5.4 Interface HDFile (HistoryDataFile)

```
Voir le paragraphe 4.3.4 pour le comportement.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants.

- objectDeleteNotificationPackage: présent si une instance le prend en charge.
*/
     valuetype HDFileValueType: itut_x780::ManagedObjectValueType
     {
          public HDFileFormatType     fileFormat;
               // GET, SET-BY-CREATE
          public URLType          fileLocation;
               // GET, SET-BY-CTEATE
          public GeneralizedTimeType  periodEndTime;
               // GET, SET-BY-CREATE
          public TimePeriodType       fileRetentionDuration;
               // GET, SET-BY-CREATE
     }; // valuetype HDFileValueType


     interface HDFile: itut_x780::ManagedObject
     {
          HDFileFormatType fileFormatGet ()
               raises (itut_x780::ApplicationError);

          URLType fileLocationGet ()
               raises (itut_x780::ApplicationError);

          GeneralizedTimeType periodEndTimeGet ()
               raises (itut_x780::ApplicationError);

          TimePeriodType fileRetentionDurationGet ()
               raises (itut_x780::ApplicationError);

          MANDATORY_NOTIFICATION(
               itut_x780::Notifications, objectCreation)
          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, objectDeletion,
               deleteNotificationPackage)

     }; // interface HDFile


     interface HDFileFactory: itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID, // auto naming if empty string
               out MONameType name,
               in HDFileFormatType fileFormat,
               in URLType fileLocation,
               in GeneralizedTimeType periodEndTime,
               in TimePeriodType fileRetentionDuration)
               raises (itut_x780::ApplicationError,
                    itut_x780::CreateError);
```

```
     }; // interface HDFileFactory


/**
```

5.5.5 Interface AbstractHDScanner (AbstractHistoryDataScanner)

```
Voir le paragraphe 4.3.5 pour le comportement.
*/
     valuetype AbstractHDScannerValueType: ScannerValueType
     {
          public HDFileFormatSetType  supportedFileFormatList;
               // GET
          public HDFileFormatType     hdFileFormat;
               // GET, SET
          public TimePeriodType       hdFileRetentionDuration;
               // GET, SET
          public TimePeriodType       beginRelativeTime;
               // GET, SET
          public TimePeriodType       endRelativeTime;
               // GET, SET
     }; // valuetype HDScannerValueType


     interface AbstractHDScanner: Scanner
     {

          HDFileFormatSetType supportedFileFormatListGet ()
               raises (itut_x780::ApplicationError);

/**
The hdFileFormat must be one of the supported file formats.
*/
          HDFileFormatType hdFileFormatGet ()
               raises (itut_x780::ApplicationError);

          void hdFileFormatSet
               (in HDFileFormatType hdFileFormat)
               raises (itut_x780::ApplicationError);

          TimePeriodType hdFileRetentionDurationGet ()
               raises (itut_x780::ApplicationError);

          void hdFileRetentionDurationSet
               (in TimePeriodType hdFileRetentionDuration)
               raises (itut_x780::ApplicationError);

          TimePeriodType beginRelativeTimeGet ()
               raises (itut_x780::ApplicationError);

          void beginRelativeTimeSet
               (in TimePeriodType beginRelativeTime)
               raises (itut_x780::ApplicationError);

          TimePeriodType endRelativeTimeGet ()
               raises (itut_x780::ApplicationError);

          void endRelativeTimeSet
               (in TimePeriodType endRelativeTime)
               raises (itut_x780::ApplicationError);
```

```
        HistoryDataSeqValueType returnHDReport
             (in unsigned short howMany,
             out HistoryDataIterator resultIterator)
             raises (itut_x780::ApplicationError);

        void generateHDFileNow ()
             raises (itut_x780::ApplicationError);

    };  // interface AbstractHDScanner


/**
```

5.5.6 Interface ListHDScanner

```
Voir le paragraphe 4.3.6 pour le comportement.
*/
    valuetype ListHDScannerValueType: AbstractHDScannerValueType
    {
        public CurrentDataNameSetType    cdInstanceList;
             // GET, SET
    }; // valuetype ListHDScannerValueType


    interface ListHDScanner: AbstractHDScanner
    {
        CurrentDataNameSetType cdInstanceListGet ()
             raises (itut_x780::ApplicationError);

        void cdInstanceListSet
             (in CurrentDataNameSetType instanceList)
             raises (itut_x780::ApplicationError);

        void cdInstanceListAdd
             (in CurrentDataNameSetType instanceList)
             raises (itut_x780::ApplicationError);

        void cdInstanceListRemove
             (in CurrentDataNameSetType instanceList)
             raises (itut_x780::ApplicationError);

    }; // interface ListHDScanner


    interface ListHDScannerFactory: itut_x780::ManagedObjectFactory
    {
        itut_x780::ManagedObject create
             (in NameBindingType nameBinding,
             in MONameType superior,
             in string reqID, // auto naming if empty string
             out MONameType name,
             in CurrentDataNameSetType cdInstanceList,
             in HDFileFormatType fileFormat,
             in TimePeriodType fileRetentionDuration,
             in TimePeriodType beginRelativeTime,
             in TimePeriodType endRelativeTime,
             in AdministrativeStateType administrativeState,
             in TimePeriodType granularityPeriod,
             in GeneralizedTimeType periodSynchronizationTime)
                  // conditional
                  // periodSynchronizationPackage
```

```
            raises (itut_x780::ApplicationError,
                itut_x780::CreateError);

    };  // interface ListHDScannerFactory


/**
```

5.5.7 Interface ClassHDScanner

```
Voir le paragraphe 4.3.7 pour le comportement.
*/
    valuetype ClassHDScannerValueType: AbstractHDScannerValueType
    {
        public CDClassSelectionSetType   cdClassSelectionList;
            // GET, SET
    }; // valuetype ClassHDScannerValueType


    interface ClassHDScanner: AbstractHDScanner
    {
        CDClassSelectionSetType cdClassSelectionListGet ()
            raises (itut_x780::ApplicationError);

        void cdClassSelectionListSet
            (in CDClassSelectionSetType classList)
            raises (itut_x780::ApplicationError);

        void cdClassSelectionListAdd
            (in CDClassSelectionSetType classList)
            raises (itut_x780::ApplicationError);

        void cdClassSelectionListRemove
            (in CDClassSelectionSetType classList)
            raises (itut_x780::ApplicationError);

    }; // interface ClassHDScanner


    interface ClassHDScannerFactory: itut_x780::ManagedObjectFactory
    {
        itut_x780::ManagedObject create
            (in NameBindingType nameBinding,
            in MONameType superior,
            in string reqID, // auto naming if empty string
            out MONameType name,
            in CDClassSelectionSetType cdClassSelectionList,
            in HDFileFormatType fileFormat,
            in TimePeriodType fileRetentionDuration,
            in TimePeriodType beginRelativeTime,
            in TimePeriodType endRelativeTime,
            in AdministrativeStateType administrativeState,
            in TimePeriodType granularityPeriod,
            in GeneralizedTimeType periodSynchronizationTime)
                // conditional
                // periodSynchronizationPackage
            raises (itut_x780::ApplicationError,
                itut_x780::CreateError);

    };  // interface ClassHDScannerFactory


/**
```

5.5.8 Interface ScopedFilteredHDScanner

```
Voir le paragraphe 4.3.8 pour le comportement.
*/
     valuetype ScopedFiltedHDScannerValueType: AbstractHDScannerValueType
     {
          public MONameType       base;
               // GET, SET
          public ScopeType        scope;
               // GET, SET
          public FilterType       filter;
               // GET, SET
          public LanguageType     language;
               // GET, SET
     }; // valuetype ScopedFiltedHDScannerValueType


     interface ScopedFiltedHDScanner: AbstractHDScanner
     {
          MONameType baseGet ()
               raises (itut_x780::ApplicationError);

          void baseSet
               (in MONameType base)
               raises (itut_x780::ApplicationError);

          ScopeType scopeGet ()
               raises (itut_x780::ApplicationError);

          void scopeSet
               (in ScopeType scope)
               raises (itut_x780::ApplicationError);

          FilterType filterGet ()
               raises (itut_x780::ApplicationError);

          void filterSet
               (in FilterType filter)
               raises (itut_x780::ApplicationError);

          LanguageType languageGet ()
               raises (itut_x780::ApplicationError);

          void languageSet
               (in LanguageType language)
               raises (itut_x780::ApplicationError);

     }; // interface ScopedFiltedHDScanner


     interface ScopedFiltedHDScannerFactory:
          itut_x780::ManagedObjectFactory
     {
          itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID, // auto naming if empty string
               out MONameType name,
               in MONameType base,
               in ScopeType scope,
               in FilterType filter,
               in LanguageType language,
               in HDFileFormatType fileFormat,
               in TimePeriodType fileRetentionDuration,
```

```
                in TimePeriodType beginRelativeTime,
                in TimePeriodType endRelativeTime,
                in AdministrativeStateType administrativeState,
                in TimePeriodType granularityPeriod,
                in GeneralizedTimeType periodSynchronizationTime)
                     // conditional
                     // periodSynchronizationPackage
                raises (itut_x780::ApplicationError,
                     itut_x780::CreateError);

     };  // interface ScopedFiltedHDScannerFactory


/**
```

5.6 Interfaces-façades

```
Les releveurs-façades de données chronologiques sont définis ici conformément à
la Rec. UIT-T X.780.1. Les releveurs de données chronologiques à granularité fine
seront cependant suffisants pour l'implémentation des façades.
*/

/**
```

5.6.1 Interface-façade Scanner_F

```
Voir le paragraphe 4.3.1 pour le comportement.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants.

- periodSynchronizationPackage: présent si une synchronisation interne par agent
configurable est requise pour les périodes répétitives.

L'interface Scanner n'est pas instanciable.
*/
     interface Scanner_F: itut_x780::ManagedObject_F
     {
          AdministrativeStateType administrativeStateGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

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

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

          TimePeriodType granularityPeriodGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void granularityPeriodSet
               (in MONameType name,
               in TimePeriodType granularityPeriod)
               raises (itut_x780::ApplicationError);

          GeneralizedTimeType periodSynchronizationTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError,
                    NOperiodSynchronizationPackage);
```

```
          void periodSynchronizationTimeSet
               (in MONameType name,
               in GeneralizedTimeType periodSyncTime)
               raises (itut_x780::ApplicationError,
                    NOperiodSynchronizationPackage);

          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, objectCreation,
               createDeleteNotificationsPackage)
          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, objectDeletion,
               createDeleteNotificationsPackage)
          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, attributeValueChange,
               attributeValueChangeNotificationPackage)
          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, stateChange,
               stateChangeNotificationPackage)

     };  // interface Scanner_F


/**
```

5.6.2 Interface-façade CurrentData_F

```
Voir le paragraphe 4.3.2.1 pour le comportement.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants.

- thresholdPackage: présent si une Notification d'alarme de QS doit être émise
pour un franchissement de seuil.

- zeroSupressionPackage: présent si une instance le prend en charge.

L'interface CurrentData n'est pas instanciable.
*/
     interface CurrentData_F: Scanner_F
     {
/**
Define default value for suspectIntervalFlag
*/
          const boolean suspectIntervalFlagDefault = FALSE;

/**
Define default value for maxSuppressedIntervals. The default value -1 means that
there is no limit on the number of consecutive suppressed intervals. On the other
hand, the maxSuppressedIntervals is effectively equal to infinity.
*/
          const short maxSuppressedIntervalsDefault = -1;

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

          void suspectIntervalFlagDefaultSet
               (in MONameType name,
               in boolean suspectIntervalFlag)
               raises (itut_x780::ApplicationError);

          TimeIntervalType elapsedTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);
```

```
short historyRetentionGet
     (in MONameType name)
     raises (itut_x780::ApplicationError);

void historyRetentionSet
     (in MONameType name,
     in short intervalNum)
     raises (itut_x780::ApplicationError);

MONameSetType thresholdDataInstanceListGet
     (in MONameType name)
     raises (itut_x780::ApplicationError,
          NOthresholdPackage);

void thresholdDataInstanceListSet
     (in MONameType name,
     in MONameSetType dataInstanceList)
     raises (itut_x780::ApplicationError,
          NOthresholdPackage);

void thresholdDataInstanceListAdd
     (in MONameType name,
     in MONameSetType dataInstanceList)
     raises (itut_x780::ApplicationError,
          NOthresholdPackage);

void thresholdDataInstanceListRemove
     (in MONameType name,
     in MONameSetType dataInstanceList)
     raises (itut_x780::ApplicationError,
          NOthresholdPackage);

boolean getMostRecent
     (in MONameType name,
     out HistoryDataValueType historyData)
     raises (itut_x780::ApplicationError);

HistoryDataSeqValueType getBetween
     (in MONameType name,
     in GeneralizedTimeType startTime,
     in GeneralizedTimeType endTime,
     in unsigned short howMany,
     out HistoryDataIterator resultIterator)
     raises (itut_x780::ApplicationError);

short numSuppressedIntervalsGet
     (in MONameType name)
     raises (itut_x780::ApplicationError);

short maxSuppressedIntervalsGet
     (in MONameType name)
     raises (itut_x780::ApplicationError);

void maxSuppressedIntervalsSet
     (in MONameType name,
     in short maxSuppInterval)
     raises (itut_x780::ApplicationError);

CONDITIONAL_NOTIFICATION(
     itut_x780::Notifications, qualityOfServiceAlarm,
     NOthresholdPackage)
```

```
     };  // interface CurrentData_F


/**
```

5.6.3 Interface-façade ThresholdData_F

```
Voir le paragraphe 4.3.3 pour le comportement.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants.

- counterThresholdListPackage: présent si une instance le prend en charge et si
le paquetage gaugeThresholdListPackage n'est pas présent.

- gaugeThresholdListPackage: présent si une instance le prend en charge et si le
paquetage counterThresholdListPackage n'est pas présent.
*/
     interface ThresholdData_F: itut_x780::ManagedObject_F
     {
          CounterThresholdSetType counterThresholdListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError,
                    NOcounterThresholdListPackage);

          void counterThresholdListSet
               (in MONameType name,
               in CounterThresholdSetType counterThresholdList)
               raises (itut_x780::ApplicationError,
                    NOcounterThresholdListPackage);

          void counterThresholdListAdd
               (in MONameType name,
               in CounterThresholdSetType counterThresholdList)
               raises (itut_x780::ApplicationError,
                    NOcounterThresholdListPackage);

          void counterThresholdListRemove
               (in MONameType name,
               in CounterThresholdSetType counterThresholdList)
               raises (itut_x780::ApplicationError,
                    NOcounterThresholdListPackage);

          GaugeThresholdSetType gaugeThresholdListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError,
                    NOgaugeThresholdListPackage);

          void gaugeThresholdListSet
               (in MONameType name,
               in GaugeThresholdSetType gaugeThresholdList)
               raises (itut_x780::ApplicationError,
                    NOgaugeThresholdListPackage);

          void gaugeThresholdListAdd
               (in MONameType name,
               in GaugeThresholdSetType gaugeThresholdList)
               raises (itut_x780::ApplicationError,
                    NOgaugeThresholdListPackage);

          void gaugeThresholdListRemove
               (in MONameType name,
               in GaugeThresholdSetType gaugeThresholdList)
               raises (itut_x780::ApplicationError,
                    NOgaugeThresholdListPackage);
```

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

     };  // interface ThresholdData_F


/**
```

5.6.4 Interface-façade HDFile_F (HistoryDataFile)

```
Voir le paragraphe 4.3.4 pour le comportement.

Cette interface contient les PAQUETAGES CONDITIONNELS suivants.

- objectDeleteNotificationPackage: présent si une instance le prend en charge.
*/
     interface HDFile_F: itut_x780::ManagedObject_F
     {
          HDFileFormatType fileFormatGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          URLType fileLocationGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          GeneralizedTimeType periodEndTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          TimePeriodType fileRetentionDurationGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          MANDATORY_NOTIFICATION(
               itut_x780::Notifications, objectCreation)
          CONDITIONAL_NOTIFICATION(
               itut_x780::Notifications, objectDeletion,
               deleteNotificationPackage)

     }; // interface HDFile_F



/**
```

5.6.5 Interface-façade AbstractHDScanner_F (AbstractHistoryDataScanner)

```
Voir le paragraphe 4.3.5 pour le comportement.
*/
     interface AbstractHDScanner_F: Scanner_F
     {
```

```
          HDFileFormatSetType supportedFileFormatListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

/**
The hdFileFormat must be one of supported file format.
*/
          HDFileFormatType hdFileFormatGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void hdFileFormatSet
               (in MONameType name,
               in HDFileFormatType hdFileFormat)
               raises (itut_x780::ApplicationError);

          TimePeriodType hdFileRetentionDurationGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void hdFileRetentionDurationSet
               (in MONameType name,
               in TimePeriodType hdFileRetentionDuration)
               raises (itut_x780::ApplicationError);

          TimePeriodType beginRelativeTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void beginRelativeTimeSet
               (in MONameType name,
               in TimePeriodType beginRelativeTime)
               raises (itut_x780::ApplicationError);

          TimePeriodType endRelativeTimeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void endRelativeTimeSet
               (in MONameType name,
               in TimePeriodType endRelativeTime)
               raises (itut_x780::ApplicationError);

          HistoryDataSeqValueType returnHDReport
               (in MONameType name,
               in unsigned short howMany,
               out HistoryDataIterator resultIterator)
               raises (itut_x780::ApplicationError);

          void generateHDFileNow
               (in MONameType name)
               raises (itut_x780::ApplicationError);

     };  // interface AbstractHDScanner_F


/**
```

5.6.6 Interface-façade ListHDScanner_F

```
Voir le paragraphe 4.3.6 pour le comportement.
*/
     interface ListHDScanner_F: AbstractHDScanner_F
     {
```

```
          CurrentDataNameSetType cdInstanceListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void cdInstanceListSet
               (in MONameType name,
               in CurrentDataNameSetType instanceList)
               raises (itut_x780::ApplicationError);

          void cdInstanceListAdd
               (in MONameType name,
               in CurrentDataNameSetType instanceList)
               raises (itut_x780::ApplicationError);

          void cdInstanceListRemove
               (in MONameType name,
               in CurrentDataNameSetType instanceList)
               raises (itut_x780::ApplicationError);

     }; // interface ListHDScanner_F


/**
```

5.6.7 Interface-façade ClassHDScanner_F

```
Voir le paragraphe 4.3.7 pour le comportement.
*/
     interface ClassHDScanner_F: AbstractHDScanner_F
     {
          CDClassSelectionSetType cdClassSelectionListGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void cdClassSelectionListSet
               (in MONameType name,
               in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);

          void cdClassSelectionListAdd
               (in MONameType name,
               in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);

          void cdClassSelectionListRemove
               (in MONameType name,
               in CDClassSelectionSetType classList)
               raises (itut_x780::ApplicationError);

     }; // interface ClassHDScanner_F


/**
```

5.6.8 Interface-façade ScopedFilteredHDScanner_F

```
Voir le paragraphe 4.3.8 pour le comportement.
*/
     interface ScopedFiltedHDScanner_F: AbstractHDScanner_F
     {
          MONameType baseGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);
```

```
          void baseSet
               (in MONameType name,
               in MONameType base)
               raises (itut_x780::ApplicationError);

          ScopeType scopeGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void scopeSet
               (in MONameType name,
               in ScopeType scope)
               raises (itut_x780::ApplicationError);

          FilterType filterGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void filterSet
               (in MONameType name,
               in FilterType filter)
               raises (itut_x780::ApplicationError);

          LanguageType languageGet
               (in MONameType name)
               raises (itut_x780::ApplicationError);

          void languageSet
               (in MONameType name,
               in LanguageType language)
               raises (itut_x780::ApplicationError);

     }; // interface ScopedFiltedHDScanner_F


/**
```

5.7 Notifications

```
Aucune notification spécifique du modèle n'est définie dans la présente
Recommandation.
*/


/**
```

5.8 Corrélations de noms

```
*/

/**
The following module contains name-binding information.
*/
     module NameBinding
     {

/**
```

5.8.1 Interface ThresholdData

```
Cette corrélation de noms est utilisée pour nommer l'objet ThresholdData en
fonction d'un objet d'élément géré.
```

```
This name binding is used to name the ThresholdData object to a Managed Element
object.
*/
          module ThresholdData_ManagedElement
          {
               const string  superiorClass =
                    "itut_m3120::ManagedElement";
               const boolean superiorSubclassesAllowed = TRUE;
               const string  subordinateClass =
                    "itut_q822d1::ThresholdData";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = TRUE;
               const DeletePolicyType deletePolicy =
                    itut_x780::deleteOnlyIfNoContainedObjects;
               const string  kind = "ThresholdData";
          }; // module ThresholdData_ManagedElement

/**
```

5.8.2 Interface HDFile

```
Cette corrélation de noms est utilisée pour nommer l'objet HDFile en fonction
d'un objet AbstractHDScanner.

*/
          module HDFile_AbstractHDScanner
          {
               const string  superiorClass =
                    "itut_q822d1::AbstractHDScanner";
               const boolean superiorSubclassesAllowed = TRUE;
               const string  subordinateClass =
                    "itut_q822d1::HDFile";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = FALSE;
               const DeletePolicyType deletePolicy =
                    itut_x780::deleteOnlyIfNoContainedObjects;
               const string  kind = "HDFile";
          }; // module HDFile_AbstractHDScanner

/**
```

5.8.3 Interface AbstractHDScanner

```
Cette corrélation de noms est utilisée pour nommer l'objet AbstractHDScanner en
fonction d'un élément géré.
*/
          module AbstractHDScanner_ManagedElement
          {
               const string  superiorClass =
                    "itut_m3120::ManagedElement";
               const boolean superiorSubclassesAllowed = TRUE;
               const string  subordinateClass =
                    "itut_q822d1::AbstractHDScanner";
               const boolean subordinateSubclassesAllowed = TRUE;
               const boolean managerCreatesAllowed = TRUE;
               const DeletePolicyType deletePolicy =
                    itut_x780::deleteOnlyIfNoContainedObjects;
               const string  kind = "HDScanner";
          }; // module AbstractHDScanner_ManagedElement
```

```
    }; // module NameBinding


/**
```

5.9 Module HDFileFormatConst

```
Ce module contient des valeurs constantes identifiant des formats de fichier de
données chronologiques.
*/
    module HDFileFormatConst
    {
        const string moduleName =
            "itut_q822d1::HDFileFormatConst";

        const short unknown = 0;

        const short q822d1FileText = 1;

        const short q822d1FileXML = 2;

    }; // HDFileFormatConst


}; // module itut_q822d1



#endif // _itut_q822_1_idl_
```

ANNEXE A

Format de fichier

La présente annexe décrit le format d'un fichier de données de mesure de qualité de fonctionnement, nommé q822d1FileText.

A.1 Formalisme BNF de format de fichier texte

Le formalisme BNF présenté dans la présente annexe définit le format du fichier de données de mesure de qualité de fonctionnement nommé "q822d1FileText".

```
<File> := #file <FD> <FileType> <Newline>
    <Nodes> #endfile <Newline>

<FileType> := <FileFormat> <FD> <FormatVersion> <FD> <ModelParadigm>

<FileFormat> := <String>

<FormatVersion> := <String>

<ModelParadigm> := <String>

<Nodes> := <Node> <Nodes>
    | <Node>

<Node> := #node <FD> <NodeID> <FD> <MeasuredObjectIDPrefix> <Newline>
    <Tables> #endnode <Newline>
```

```
<NodeID> := <String>

<MeasuredObjectIDPrefix> := <ID>
     | <Empty>

<Tables> := <MeasuredObjectIDAliases> <Table> <Tables>
     | <MeasuredObjectIDAliases> <Table>

<MeasuredObjectIDAliases> := <MeasuredObjectIDAlias> <MeasuredObjectIDAliases>
     | <MeasuredObjectIDAlias>
     | <Empty>

<MeasuredObjectIDAlias> := #idalias <FD> <ShortID> <FD>     <LongID> <Newline>

<ShortID> := <ID>

<LongID> := <ID>

<Table> := #table <FD> <Header> <RecordGroups> #endtable <Newline>

<Header> := #header <FD> <MeasuredObjectType> <FD> <GranualarityPeriod>
     <Newline> <DataSetTypes> #endheader <Newline>

<MeasuredObjectType> := <String>

<GranualarityPeriod> := <TimePeriodValue>

<DataSetTypes> := <DataSetType> <DataSetTypes>
     | <DataSetType>

<DataSetType> := #dataset <FD> <DataSetTypeName> <FD> <DataSetTypeIndex>
     <Newline> suspect <FD> <ParameterTypes> <Newline>
     #enddataset <Newline>

<DataSetTypeName> := <String>

<DataSetTypeIndex> := <IntegerString>

<ParameterTypes> := <ParameterType> <ParameterTypes>
     | <ParameterType>

<ParameterType> := <String> <FieldSeparator>

<RecordGroups> := <RecordGroup> <RecordGroups>
     | <RecordGroup>

<RecordGroup> := #period <FD> <PeriodEndTime> <Newline>
     <Records> #endperiod <NewLine>

<PeriodEndTime> := <TimeValue>

<Records> := <Record> <Records>
     | <Record>

<Record> := <MeasuredObjectID> <FD> <DataSetValues> <Newline>

<MeasuredObjectID> := <ID>  // can be aliased and prefixed

<DataSetValues> := <DataSetValue> <DataSetValues>
     | <DataSetValue>

<DataSetValue> := <DataSetTypeIndex> <FD> <Suspect> <FD> <ParameterValues>

<Suspect> := <BooleanValue>
```

```
<ParameterValues> := <ParameterValue> <ParameterValues>
     | <ParameterValue>

<ParameterValue> := <Value> <FD>

<Value> := <CounterValue>
     | <GaugeValue>
     | <TidemarkValue>
     | <BooleanValue>
     | <EnumValue>
     | <TimeValue>
     | <TimePeriodValue>

<FD> := :     // field delimiter

<EscapeCharacter> := \

<Newline> := '\n'  // line delimiter

<ID> := <String>

<CounterValue> := <IntegerString>

<GaugeValue> := <FloatString>

<TidemarkValue> := <FloatString>

<BooleanValue> := <BooleanString>

<EnumValue> := <IntegerString>

<TimeValue> := <String> // UTC in format "YYYYMMDDHHMMSS.fffZ"

<TimePeriodValue> := <IntegerString> <TimePeriodType>

<BooleanString> := T | F    // true or false

<TimePeriodType> := days | hours | minutes | seconds

<String> := {ISO 8859-1(Latin-1) characters}

<IntegerString> := <IntegerString> <Digit>
     | <Digit>

<Digit> := 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

<FloatString> := <Mantissa> | <Mantissa> <Exponent>

<Mantissa> := <IntegerString> | <IntegerString> . | . <IntegerString>
         | <IntegerString> . <IntegerString>

<Exponent> := <Exp> <Sign> <IntegerString>

<Sign> := + | -

<Exp> := E | e
```

NOTE 1 – L'espace vide dans le fichier est sans signification. Le générateur de fichiers ne doit pas insérer d'espace vide dans le fichier. L'analyseur de fichier doit omettre et ignorer tout espace vide rencontré dans le fichier.

NOTE 2 – Un analyseur de fichier rétrocompatible doit omettre et ignorer d'éventuelles données situées entre des balises #<tag> et #end<tag> inconnues.

NOTE 3 – L'extension vers ce format de fichier nécessite de nouvelles définitions de balises #<tag>.

A.2 Description du format du fichier texte

Les données chronologiques sont contenues dans un fichier de mesures de qualité de fonctionnement. Ce fichier se compose d'une définition du type de fichier et d'un ensemble de données métrologiques. L'ordre des données n'est pas significatif. Le fichier doit contenir au moins un ensemble de données. Sinon, il n'est pas créé.

Le fichier de mesures de qualité de fonctionnement doit suivre les règles suivantes pour sa production (voir la Figure 7, qui décrit la structure du fichier):

1) le type de fichier est défini par le format du fichier, par la version de ce format et par le paradigme de modélisation. L'objet FileFormat indique le format adopté pour le fichier. L'objet FormatVersion spécifie une version du format particulier du fichier. La valeur de l'objet FileFormat dans la présente Recommandation est "q822d1FileText" et la valeur de l'objet FormatVersion est "version1";
2) le paradigme de modélisation est celui du système de gestion qui produit le fichier. Si celui-ci est produit à partir d'un système de gestion CORBA conforme au cadre CORBA de l'UIT-T, la valeur de l'objet ModelParadigm doit être "ITUTCORBA". Sinon, une valeur appropriée doit être fournie: par exemple "CMIP", "SNMP", "TL1", "abcCORBA", etc.;
3) la spécification de gestion de qualité de fonctionnement pour un domaine de modélisation d'information spécifique doit indiquer la sémantique et la convention de nommage des objets ModelParadigm, NodeID, MeasuredObjectType, DataSetTypeName, ParameterType, ID, MeasuredObjectIDPrefix, MeasuredObjectID;
4) l'objet MeasuredObjectIDPrefix définit le préfixe commun des identificateurs MeasuredObjectID dans une section nodale;
5) l'objet MeasuredObjectIDAlias définit le pseudonyme court MeasuredObjectID pour un identificateur long MeasuredObjectID de nom FDN. Le domaine d'une définition de pseudonyme va jusqu'à la fin du fichier sauf redéfinition par un autre objet MeasuredObjectIDAlias;
6) l'indice DataSetTypeIndex d'une valeur DataSetValue indique le type correspondant d'ensemble de données tel que défini dans l'en-tête de table;
7) toutes les données métrologiques contenues dans une table ont la même période de granularité et représentent des mesures concernant le même type d'objet mesuré;
8) s'il n'y a pas de données pour une période particulière, le fichier doit quand même contenir les balises #period et #endperiod avec un champ d'information vide pour cette période.

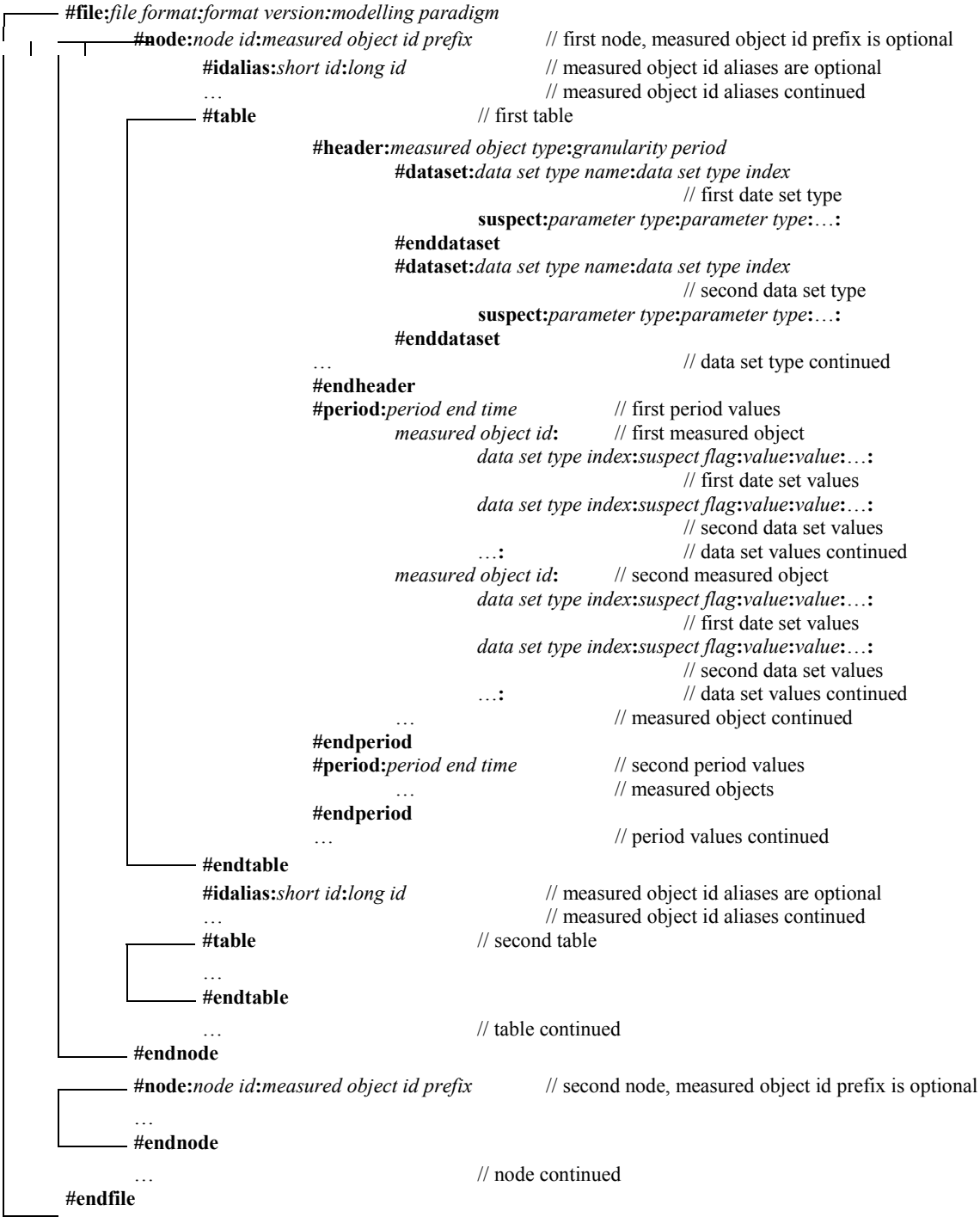

Figure 7/Q.822.1 – Illustration du format de fichier PM

ANNEXE B

Convention de format de fichier CORBA

La présente annexe décrit la sémantique et la convention de nommage des objets de type ModelParadigm, NodeID, MeasuredObjectType, DataSetTypeName, ParameterType, ID, MeasuredObjectIDPrefix, MeasuredObjectName dans le cadre CORBA de l'UIT-T.

1) *ModelParadigm* – doit être "ITUTCORBA".
2) *NodeID* – Nom FDN d'un élément géré ou nom spécifique d'une application d'un élément géré.
3) *MeasuredObjectType* – Nom d'interface détecté d'un objet géré.
4) *DataSetTypeName* – Nom d'interface détecté de la classe CurrentData ou de sa sous-classe.
5) *ParameterType* – Nom d'attribut dans la classe HistoryDataValueType ou dans son sous-type de valeur.
6) *ID* – Nom FDN d'un nom CORBA d'objet géré.
7) *MeasuredObjectIDPrefix* – Nom RDN d'un nom CORBA d'objet géré.
8) *MeasuredObjectID* – Nom FDN d'un nom CORBA d'objet géré mesuré.

APPENDICE I

Exemple d'utilisation de l'héritage des classes CurrentDataValueType et HistoryDataValueType

Comme décrit dans le § 4.3.2, la classe HistoryDataValueType est utilisée comme mémoire chronologique en lecture seulement de données actuelles. Ce type de valeur sera accessible par certaines méthodes à l'interface CurrentData. Lorsqu'une donnée actuelle est sous-classée, il est prévu que l'objet HistoryDataValueType correspondant soit sous-classé avec elle.

Le présent appendice donne un exemple de la manière de définir un objet HistoryDataValueType en fonction d'une interface CurrentData et de la manière d'accéder aux valeurs de données chronologiques au moyen de l'instance de l'objet CurrentData correspondant.

Dans cet exemple, l'interface TrafficControlCurrentData est utilisée afin de surveiller l'efficacité d'une régulation de trafic. Cette interface est décrite ci-dessous.

```
interface TrafficControlCurrentData: itut_q822d1::CurrentData
{
/**
the number of calls which are actually affected by the traffic control instance.
*/
     short callsAffectedByTrafficControlGet ()
          raises (itut_x780::ApplicationError);

/**
the number of calls which were offered to the traffic control instance.
*/
     short callsOfferedToTrafficControlGet ()
          raises (itut_x780::ApplicationError,
               NOcallsOfferedPackage);
}; // interface TrafficControlCurrentData
```

Sur la base du cadre NM de l'architecture CORBA, une interface TrafficControlCurrentData doit être définie comme suit afin de recevoir tous les attributs pris en charge par une instance de l'interface TrafficControlCurrentDataValueType:

```
valuetype TrafficControlCurrentDataValueType: itut_q822d1::CurrentDataValueType
{
     public short callsAffectedByTrafficControl;
          // trafficControlCurrentDataPackage
          // GET
     public short callsOfferedToTrafficControl;
          // conditional
          // callsOfferedPackage
          // GET
}; // valuetype TrafficControlCurrentDataValueType
```

Et afin de sauvegarder les mesures collectées dans une instance de l'interface TrafficControlCurrentData, un objet TrafficControlHistoryDataValueType doit être défini comme suit:

```
valuetype TrafficControlHistoryDataValueType: itut_q822d1::HistoryDataValueType
{
     public short callsAffectedByTrafficControl;
          // GET
     public short callsOfferedToTrafficControl;
          // GET
}; // valuetype TrafficControlHistoryDataValueType
```

Compte tenu de la définition de l'interface TrafficControlCurrentData et de l'objet TrafficControlHistoryDataValueType, une instance de l'interface TrafficControlCurrentData doit être créée lors de l'exécution du programme afin de surveiller l'efficacité d'une commande de trafic particulière de part et d'autre de l'interface TrafficControlCurrentDataFactory:

```
interface TrafficControlCurrentDataFactory: itut_x780::ManagedObjectFactory
{
     itut_x780::ManagedObject create
               (in NameBindingType nameBinding,
               in MONameType superior,
               in string reqID,   // auto naming if null
               out MONameType name,
               in IstringSetType packageNameList,
               in short historyRetention,
                    // Mandatory now (used to be conditional)
                    // GET-REPLACE
               in AdministrativeStateType administrativeState,
                    // GET-REPLACE
               in TimePeriodType granularityPeriod,
                    // GET-REPLACE
               in MONameSetType thresholdDataInstanceList,
                    // conditional
                    // thresholdPackage
                    // GET-REPLACE ADD-REMOVE
               in short maxSuppressedIntervals,
                    // conditional
                    // zeroSuppressionPackage
                    // GET-REPLACE
               in GeneralizedTimeType periodSynchronizationTime)
                    // conditional
                    // periodSynchronizationPackage
                    // GET-REPLACE
               raises (itut_x780::ApplicationError,
                    itut_x780::CreateError);

}; // interface TrafficControlCurrentDataFactory
```

A la fin de chaque période de granularité, un enregistrement de l'objet TrafficControlHistoryDataValueType est produit et les mesures callsAffectedByTrafficControl et callsOfferedToTrafficControl, collectées dans l'instance de cet objet, doivent être recopiées dans l'enregistrement chronologique.

Afin d'accéder aux valeurs de données chronologiques de type TrafficControl, une des deux méthodes suivantes est appelée à partir de l'instance de l'interface TrafficControlCurrentData: getMostRecent() et getBetween().

Afin d'accéder aux valeurs de données chronologiques dans un domaine de visibilité plus vaste, l'on utilisera une instance de l'interface HistoryDataScanner.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, circuits téléphoniques, télégraphie, télécopie et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	**Commutation et signalisation**
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse
Genève, 2002