Union internationale des télécommunications

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

X.1206

(04/2008)

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

Sécurité du cyberespace – Cybersécurité

Cadre indépendant du fournisseur de produits pour la notification automatique d'informations de sécurité et la diffusion automatique de mises à jour

Recommandation UIT-T X.1206

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX DE DONNÉES, COMMUNICATION ENTRE SYSTÈMES OUVERTS ET SÉCURITÉ

RÉSEAUX PUBLICS DE DONNÉES	X.1–X.199
INTERCONNEXION DES SYSTÈMES OUVERTS	X.200–X.299
INTERFONCTIONNEMENT DES RÉSEAUX	X.300–X.399
SYSTÈMES DE MESSAGERIE	X.400–X.499
ANNUAIRE	X.500–X.599
RÉSEAUTAGE OSI ET ASPECTS SYSTÈMES	X.600–X.699
GESTION OSI	X.700–X.799
SÉCURITÉ	X.800–X.849
APPLICATIONS OSI	X.850–X.899
TRAITEMENT RÉPARTI OUVERT	X.900–X.999
SÉCURITÉ DE L'INFORMATION ET DES RÉSEAUX	
Aspects généraux de la sécurité	X.1000–X.1029
Sécurité des réseaux	X.1030–X.1049
Gestion de la sécurité	X.1050–X.1069
Télébiométrie	X.1080–X.1099
APPLICATIONS ET SERVICES SÉCURISÉS	
Sécurité en multidiffusion	X.1100–X.1109
Sécurité des réseaux domestiques	X.1110–X.1119
Sécurité des télécommunications mobiles	X.1120–X.1139
Sécurité de la toile	X.1140–X.1149
Protocoles de sécurité	X.1150–X.1159
Sécurité d'homologue à homologue	X.1160–X.1169
Sécurité des identificateurs en réseau	X.1170–X.1179
Sécurité de la télévision par réseau IP	X.1180–X.1199
SÉCURITÉ DU CYBERESPACE	
Cybersécurité	**X.1200–X.1229**
Lutte contre le pollupostage	X.1230–X.1249
Gestion des identités	X.1250–X.1279
APPLICATIONS ET SERVICES SÉCURISÉS	
Communications d'urgence	X.1300–X.1309
Sécurité des réseaux de capteurs ubiquitaires	X.1310–X.1339

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

Recommandation UIT-T X.1206

Cadre indépendant du fournisseur de produits pour la notification automatique d'informations de sécurité et la diffusion automatique de mises à jour

Résumé

La Recommandation UIT-T X.1206 fournit un cadre pour la notification automatique d'informations de sécurité et la diffusion automatique de mises à jour. Ce cadre a pour caractéristique essentielle de ne pas dépendre du fournisseur de produits. Une fois qu'une ressource est enregistrée, des mises à jour d'informations de vulnérabilité, des corrections ou d'autres mises à jour concernant la ressource peuvent être automatiquement mises à la disposition des utilisateurs ou directement transmises aux applications.

Source

La Recommandation UIT-T X.1206 a été approuvée le 18 avril 2008 par la Commission d'études 17 (2005-2008) de l'UIT-T selon la procédure définie dans la Résolution 1 de l'AMNT.

AVANT-PROPOS

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

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

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

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

NOTE

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

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

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

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux développeurs de consulter la base de données des brevets du TSB sous http://www.itu.int/ITU-T/ipr/.

TABLE DES MATIÈRES

Recommandation UIT-T X.1206

Cadre indépendant du fournisseur de produits pour la notification automatique d'informations de sécurité et la diffusion automatique de mises à jour

1 Domaine d'application

La présente Recommandation fournit un cadre pour les flux bidirectionnels de notification et de diffusion automatiques d'informations de vulnérabilité et de diffusion de mises à jour et/ou de corrections. De plus, elle permet aux administrateurs de systèmes de connaître l'état de toute ressource relevant de leur responsabilité.

Les paragraphes 6 et 7 décrivent les problèmes liés à la maintenance des ressources du point de vue de l'identification des ressources, de la diffusion des informations et de la gestion des systèmes/réseaux.

Le paragraphe 8 donne un aperçu général d'un cadre indépendant du fournisseur de produits. Il donne un exemple de système pris en charge par l'adoption du cadre, décrit les comportements du cadre et donne un exemple de séquence d'échanges se déroulant dans le cadre. Il décrit aussi le niveau de sécurité à considérer dans le cadre indépendant du fournisseur de produits.

Le paragraphe 9 décrit les fonctionnalités et les caractéristiques de l'architecture recommandée.

Le paragraphe 10 donne les définitions des structures de données des composants du cadre.

Le paragraphe 11 contient le schéma XML défini et décrit dans le paragraphe 10.

La présente Recommandation fournit un cadre que tout fournisseur de produits peut utiliser pour la notification et la réception d'informations de vulnérabilité ainsi que pour la diffusion des corrections/mises à jour requises pour les ressources concernées. Elle définit le format des informations qui devraient être utilisées dans et entre les composants qui composent le cadre.

Elle ne définit pas les protocoles à utiliser pour la communication entre les composants étant donné que de nombreux protocoles sont pris en charge sans examen particulier.

Si certains rôles et responsabilités communs devront être définis pour une exploitation fondée sur le cadre indépendant du fournisseur de produits, un examen relatif à la définition et à la mise en œuvre des rôles possibles et de leurs responsabilités résultantes ne relève pas du domaine d'application de la présente Recommandation.

2 Références

Aucune.

3 Définitions

3.1 Termes définis dans la présente Recommandation

La présente Recommandation définit les termes suivants:

3.1.1 agent: mise en œuvre de la présente Recommandation fonctionnant en appui à une ressource installée sur un dispositif donné, en appui à une fonctionnalité de serveur ou en appui à une fonctionnalité de serveur local.

3.1.2 ressource: dispositif, équipement identifiable séparément, application, système d'exploitation ou instance de code exécutable.

3.1.3 client: dispositif qui demande des services à un autre dispositif.

3.1.4 **dispositif**: système agissant comme client, serveur, client/serveur ou serveur local.

3.1.5 **groupe**: ensemble de dispositifs exploités en tant qu'unité unique.

3.1.6 **serveur local**: client agissant en tant que nœud serveur pour d'autres clients aval.

3.1.7 **message**: demande d'une action spécifique à réaliser, par exemple une action générale telle que "Register" ("enregistrer") une ressource comme étant d'une version donnée et/ou contenant des composants de versions données, "Request" ("demander") des mises à jour, des corrections ou des informations de vulnérabilité existantes ou disponibles dans le futur, etc. Des messages étendant la fonctionnalité de la présente Recommandation peuvent être définis hors du domaine d'application de la présente Recommandation.

3.1.8 **données de message**: informations fournies en appui à un message donné. Parmi le nombre presque infini de possibilités, la présente Recommandation définit comme exemples spécifiques des données définissant des informations de version, des informations de vulnérabilité relatives à des versions données ainsi que des mises à jour ou des corrections de versions spécifiques.

3.1.9 **ensemble de message**: combinaison et association d'un identificateur unique universel, d'un message et de la définition des données de message associées au message, tous définis dans le cadre d'un schéma XML tiré, en l'étendant, de l'élément *Message_Core* défini dans la présente Recommandation.

3.1.10 **correction**: correction largement diffusée d'une vulnérabilité propre à un produit et relative à la sécurité. Méthode de mise à jour d'un fichier qui ne remplace que les parties modifiées et non la totalité du fichier.

3.1.11 **serveur**: dispositif utilisé pour répondre à des demandes d'autres dispositifs.

3.1.12 **vulnérabilité**: toute faiblesse, processus ou mécanisme administratif, ou exposition physique qui rend un ordinateur ou un réseau d'ordinateurs susceptible de subir les conséquences d'une menace.

4 Abréviations

La présente Recommandation utilise les abréviations et les acronymes suivants:

API interface de programmation d'application (*application programming interface*)

GUID identificateur unique globalement (*globally unique IDentifier*)

HTTP protocole de transfert hypertexte (*hypertext transfer protocol*)

ISIRT équipe d'intervention en cas d'incident touchant à la sécurité des informations (*information security incident response team*)

ISP fournisseur de services Internet (*Internet service provider*)

OS système d'exploitation (*operating system*)

POAS plate-forme/système d'exploitation/application/service (*platform/operating system/application/service*)

URI identificateur uniforme de ressource (*uniform resource Identifier*)

5 Conventions

Aucune.

6 Introduction

On constate qu'un nombre plus grand d'individus commencent à utiliser un ordinateur à domicile et au travail alors qu'ils sont moins nombreux à avoir une formation reconnue à l'informatique et encore moins aux problèmes de sécurité. On atteint rapidement un stade où il devient non seulement presque impossible de maintenir la sécurité mais aussi plus difficile pour les responsables de la sécurité de savoir dans quelles conditions fonctionnent les systèmes dont ils ont la responsabilité et qu'ils doivent surveiller avant qu'une infraction ou un incident soit constaté (lorsqu'il est donc déjà trop tard).

Ceci s'explique principalement par l'existence d'un très grand nombre d'ordinateurs différents, qui se trouvent dans des états différents de maintenance et de mise à jour. En ce qui concerne les questions de sécurité, la gestion de systèmes s'apparente bien moins à un processus préventif qu'à une sorte de processus de gestion "des catastrophes" et de rétablissement après interruption.

Bien qu'un certain nombre d'applications et même de systèmes d'exploitation (OS) disposent de leurs propres mécanismes de mise à jour, ils ont tous un certain nombre de problèmes en commun. L'un de ces problèmes est que tous les mécanismes de mise à jour doivent d'abord être activés pour fonctionner et qu'ils ne peuvent ensuite être autorisés à fonctionner que si la disponibilité d'une mise à jour est notifiée à l'utilisateur, à supposer que le processus de notification ait lui-même été activé par ce dernier.

Le plus regrettable sans doute est que ces problèmes ne laissent pour l'heure aucune place aux administrateurs de systèmes: à moins d'installer leurs propres systèmes de surveillance sur chacun des ordinateurs placés sous leur responsabilité, ceux-ci n'ont aucune idée du niveau général de sécurité dans les réseaux et les systèmes dont ils ont la charge.

De plus, malgré la mise à jour des logiciels à la version la plus récente, il arrive souvent que la résolution de problèmes ne réside pas dans les seules mises à jour mais plutôt dans la mise en œuvre de pratiques évoluées ne faisant pas appel à ces mises à jour mais seulement à des informations reçues par l'utilisateur final. Même si divers applications et systèmes d'exploitation peuvent avoir mis en place des mécanismes de mise à jour, aucun d'eux n'utilise de méthode uniforme pour que les utilisateurs restent informés des meilleures pratiques les plus récentes conduisant à un fonctionnement sûr et continu.

Les méthodes utilisées pour diffuser les mises à jour sont également importantes. A l'heure actuelle, toutes les mises à jour se font à travers divers mécanismes via des canaux spécialisés (un par session de mise à jour). Mais si les mises à jour ou d'autres informations importantes mises à la disposition de divers centres de redistribution (fournisseurs de services Internet ou réseaux d'entreprise par exemple) étaient ensuite diffusées dans les réseaux d'une manière avérée et sûre, on pourrait gagner en efficacité et diviser par deux ou réduire fortement la largeur de bande requise pour la diffusion exactement comme pour les serveurs proxy HTTP.

Autre situation le plus souvent mal gérée, celle où des utilisateurs découvrent un problème concernant l'utilisation ou les actions d'une ressource donnée, en n'ayant personne à qui soumettre le problème. Même si les utilisateurs peuvent contacter les administrateurs de systèmes ou d'autres personnes chargées de l'assistance informatique, utiliser une telle méthode finit par s'apparenter à une espèce de jeu des "20 questions", caractérisé par un va-et-vient incessant entre les questions de l'utilisateur et celles encore plus nombreuses que son interlocuteur lui envoie généralement en réponse.

Il est souvent difficile, même pour une personne d'appui expérimentée, d'avoir une bonne compréhension de la constitution exacte de nombreuses ressources. Compte tenu des architectures logicielles modulaires et de l'existence de divers logiciels constitués de modules issus de différents fournisseurs, qui ont tous leurs propres systèmes de versions, il peut être difficile, même si une mise à jour ou une information pertinente concernant un produit ou un sous-module donné est disponible,

de savoir où et à quoi l'appliquer, avec pour conséquence que les personnes les moins informées ignorent généralement en grande partie ce dont la sécurité de leurs systèmes pourrait dépendre.

Au bout du compte, on finit par avoir des utilisateurs qui ne sont pas informés et des responsables de la sécurité des systèmes qui sont en grande partie laissés hors du champ d'intervention.

7 Situation actuelle concernant les informations de vulnérabilité

Des informations de vulnérabilité sont à présent diffusées par de nombreux fournisseurs de produits et de nombreux organismes de sécurité (équipe d'intervention en cas d'incident touchant à la sécurité des informations (ISIRT) par exemple) pour sensibiliser les utilisateurs aux questions de sécurité et proposer des mises à jour et des corrections lorsque cela est nécessaire. Toutefois, il arrive souvent que les utilisateurs finals n'utilisent pas les informations, les mises à jour ou les corrections, voire ne savent si ce qu'on leur propose les concerne.

Si diverses raisons expliquent cette situation, il faut d'abord chercher à comprendre pourquoi les utilisateurs finals pourraient avoir des difficultés à exploiter les informations mises à leur disposition.

Un utilisateur final utilise généralement des informations de version relatives aux ressources pour déterminer si elles concernent le système, le logiciel ou les composants touchés par la vulnérabilité venant d'être détectée. Utiliser ces informations pour déterminer le système, le logiciel ou les composants affectés est toutefois difficile.

- La notation de version peut être ambiguë

Les règles de notation de version varient d'un fournisseur de produits à l'autre, d'où le risque que l'interprétation donnée par l'utilisateur final aux informations de version soit différente de celle voulue par le fournisseur. Dans l'incapacité de comprendre un système de versions spécifique d'un fournisseur, l'utilisateur risque de ne pas se rendre compte qu'une mise à jour, une correction ou une information donnée le concerne.

Prenons par exemple le cas où il existe trois versions ("v1", "v1.1" et "v1.2") du système décrites comme cible dans les informations de vulnérabilité suivantes:

"Systèmes affectés

* XXXX v1 système affecté"

Deux interprétations sont alors possibles:

1) v1 désigne simplement la version v1.0;

2) v1 correspond à v1.x, c'est-à-dire à toutes les sous-versions de la version 1.

Tenter de normaliser la notation de version n'étant pas une chose commode il faut donc lire et comprendre les règles de notation de version spécifiques d'un fournisseur. Espérer que cette démarche sera celle des utilisateurs finals n'est toutefois pas réaliste.

- La version d'un produit donné ne peut pas toujours être utilisée comme critère pour déterminer les points réels de vulnérabilité

En ce qui concerne certains fournisseurs, la version des produits généralement connus des utilisateurs pourrait ne pas permettre de déterminer si ces produits présentent ou non des vulnérabilités (Figure 1).

Dans le cas d'un produit à composants, c'est en examinant la version des différents composants et non de celle du produit proprement dit que l'on peut déterminer la présence d'une vulnérabilité.

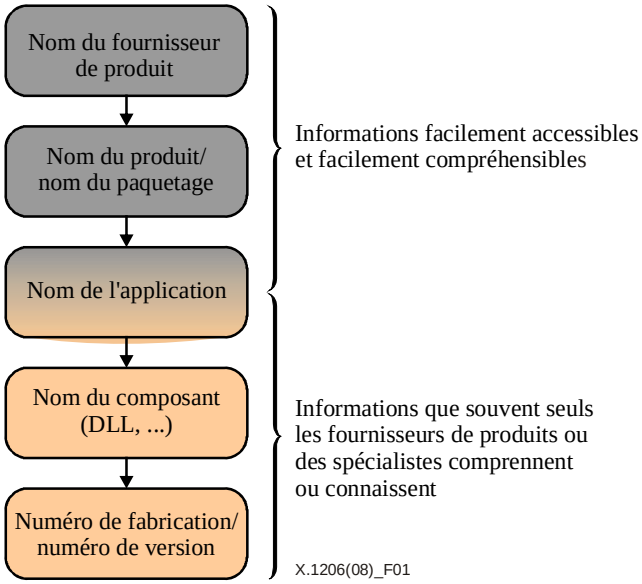


Figure 1 – Structure hiérarchique d'une "version"

- Certains produits comprennent comme composants des produits d'un autre fournisseur

Les informations de vulnérabilité font généralement référence à un produit donné. L'utilisateur final peut ne pas utiliser le produit spécifique auquel il est fait référence mais une autre application qui charge ou utilise un service du produit considéré comme vulnérable. De fait, avec les interfaces API (normes publiées "ouvertes"), un fournisseur dont les produits sont utilisés dans d'autres applications peut avoir peu d'informations, voire aucune, concernant l'endroit où sont utilisés ses produits ou sur les applications pour lesquelles ils peuvent être chargés ou utilisés.

- Les administrateurs de systèmes n'ont pas connaissance de l'état des ressources dont ils ont la responsabilité

Etant donnée que les administrateurs de systèmes dépendent de la façon dont les utilisateurs assurent la maintenance de leurs propres systèmes, à moins de vérifier par eux-mêmes chaque système un à un ou de s'en remettre aux utilisateurs pour le compte rendu des incidents, il leur est impossible de connaître l'état des systèmes dont ils assument en dernier ressort la responsabilité. Sans compte rendu des utilisateurs (souvent imprécis et incomplets pour les diverses raisons déjà mentionnées concernant la notation de version) ou applications de surveillance spécialement conçues (qui nécessitent des activités de développement et de maintenance), les responsables de la sécurité des réseaux d'entreprise ou ISP ont peu d'outils à leur disposition.

8 Aperçu général d'un cadre indépendant du fournisseur de produits

Le présent paragraphe donne un aperçu général d'un cadre indépendant du fournisseur de produits pour la diffusion d'informations de vulnérabilité, de mises à jour et de corrections.

En adoptant un tel cadre, on peut mettre en place des systèmes de diffusion d'informations de vulnérabilité, de mises à jour et de corrections tels que celui de la Figure 2 en utilisant un simple mécanisme d'abonnement. Chaque ressource, dispositif ou serveur local peut s'enregistrer auprès d'un ou de l'ensemble des serveurs disponibles pour recevoir des informations de vulnérabilité, des mises à jour et/ou des corrections demandées (diffusion) ou suggérées (extraction).

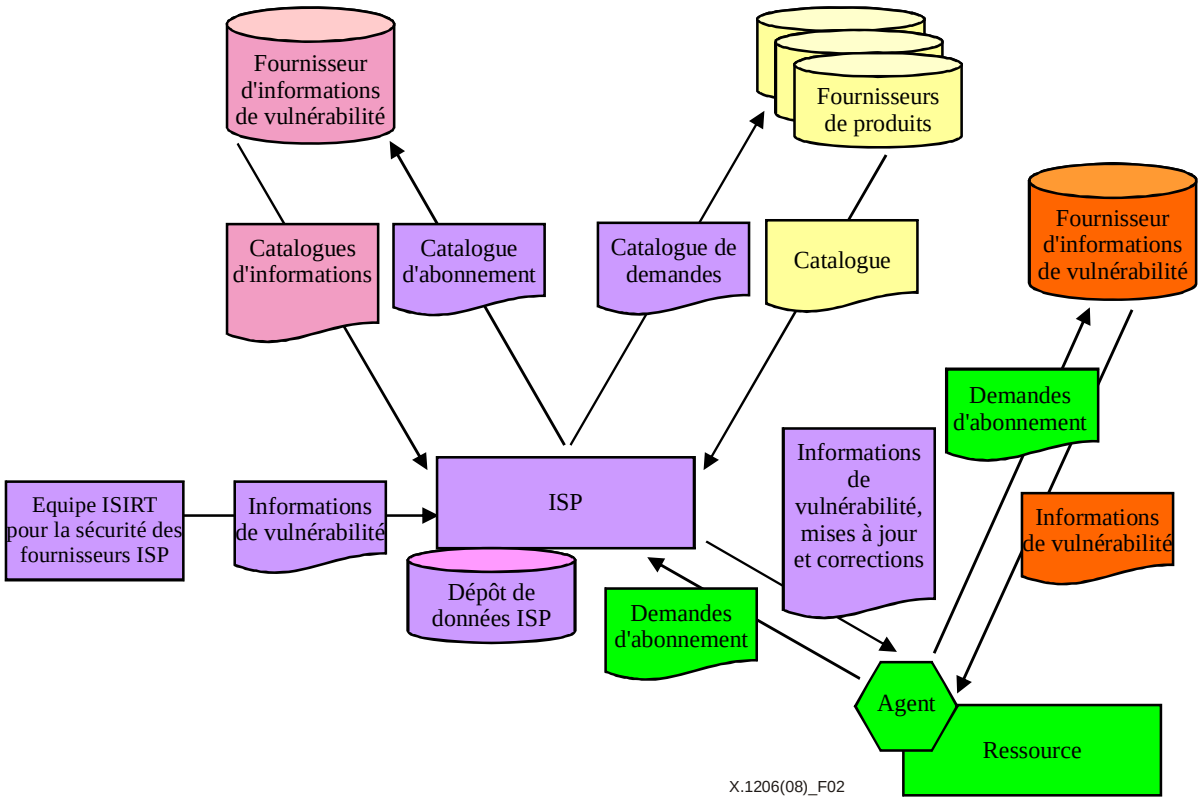


Figure 2 – Exemple d'architecture d'application

8.1 Sources multiples d'informations de vulnérabilité, de mises à jour et de corrections

Grâce à l'utilisation d'un même mécanisme de messages, toute entité peut demander des informations de vulnérabilité, des mises à jour et/ou des corrections à toute autre entité et/ou lui en fournir. Dans l'exemple de la Figure 2, un fournisseur ISP fait office de point de collecte pour agir en tant que source vers tous ses abonnés. De plus, chaque abonné peut s'enregistrer auprès de sources tierces indépendantes pour obtenir des mises à jour d'informations, des analyses de code ou même des mises à jour ou des corrections de code. Il faudrait bien sûr que toute application tentant d'appliquer une mise à jour ou une correction donnée l'accepte comme étant autorisée; cependant, la description de ce processus d'autorisation (si ce n'est en indiquant qu'elle se fait par une certaine utilisation de l'inclusion de signatures prise en charge) ne relève pas de la présente Recommandation.

8.2 Exemple de fonctionnement d'application

8.2.1 Processus pour les demandes et la fourniture d'informations et de mises à jour

La ressource (par exemple un éditeur de textes) est installée. Pendant son déroulement, l'installation est notifiée à un agent précédemment installé. Sur la base des informations qui lui ont été fournies durant l'installation de la ressource, l'agent fait une demande d'abonnement auprès du fournisseur ISP de l'utilisateur pour toutes les informations de vulnérabilité, les mises à jour et/ou les corrections.

Avant ou après avoir reçu la demande d'abonnement envoyée par l'agent de l'utilisateur, le fournisseur ISP contacte différentes sources d'informations de vulnérabilité, de mises à jour et de corrections sur la base des informations fournies par l'Agent concernant la source ou le fournisseur de l'application nouvellement installée.

De plus, l'agent de l'utilisateur a, dans cet exemple, la capacité d'interroger séparément d'autres sources d'informations de vulnérabilité et de mettre toute information reçue à la disposition de

l'utilisateur pour visualisation. Ce dernier processus pourrait être lancé par le biais d'un site web mettant en valeur les services d'une source d'informations de vulnérabilité et par le biais d'un lien de téléchargement, ou par la fourniture d'un message conforme à la présente Recommandation à partir duquel l'agent pourra agir.

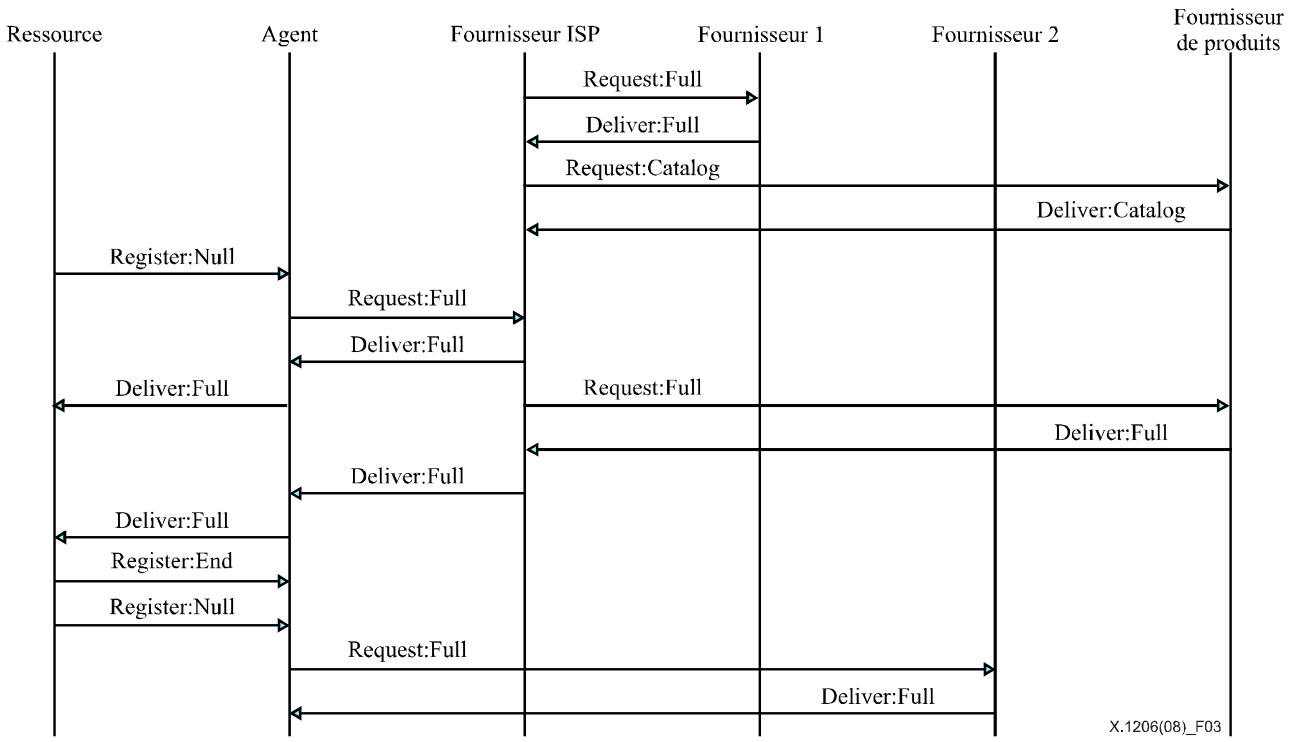


Figure 3 – Exemple de messages d'application

8.2.2 Messages pour les demandes et la fourniture d'informations et de mises à jour

Un fournisseur ISP (il pourrait tout aussi bien s'agir d'un administrateur de réseau d'entreprise ou privé) adresse des demandes à plusieurs fournisseurs d'informations de vulnérabilité, de mises à jour et de corrections. Il peut demander le contenu intégral de ce qui est disponible, pour agir comme dépôt local dès le début de l'exploitation, ou simplement demander une liste (Catalogue) des informations de vulnérabilité, des mises à jour et des corrections disponibles dont il peut ultérieurement demander des éléments particuliers dans leur intégralité. Bien que cela ne soit pas indiqué sur la Figure 3, une demande de contenu intégral pourrait recevoir comme réponse "*Major Message Not Supported*" ("message majeur non pris en charge") ou "*Minor Message Not Supported*" ("message mineur non pris en charge"), auquel cas le service demandeur essaierait de demander une liste (un catalogue). Il est toutefois préférable que les fournisseurs fassent connaître publiquement leurs politiques de demandes pour que la négociation de demandes ne soit pas nécessaire.

Dans l'exemple particulier considéré, le fournisseur ISP demande uniquement un catalogue au Fournisseur de produits étant donné que le recueil complet ("dépôt") des informations de vulnérabilité, des mises à jour et/ou des corrections du fournisseur peut être très volumineux. Toutefois, comme une demande de catalogue fait également office d'abonnement pour toutes les futures informations, mises à jour et corrections, le fournisseur ISP peut être sûr que tout ce qu'il fournit à des agents agissant au nom de leurs utilisateurs correspondra aux données disponibles les plus récentes. Cependant, il peut être commode de demander l'intégralité de ce qui est disponible étant donné que le fournisseur d'informations de vulnérabilité (par exemple le Fournisseur 1) a probablement moins de données en ligne.

Durant l'installation d'une ressource, le processus d'installation peut transmettre un message *Request:Full* (types de message Majeur: Mineur) à un agent installé localement pour demander toutes les informations de vulnérabilité, les mises à jour et les corrections actuelles et futures à mesure qu'elles sont disponibles pour la version spécifique de la ressource ou de l'un quelconque de ses composants.

L'agent fait ensuite une demande *Request:Full* au fournisseur ISP au nom de la ressource nouvellement installée.

Etant donné que, dans cet exemple, il achemine déjà des informations de vulnérabilité en provenance du Fournisseur 1 version dépôt, le fournisseur ISP peut fournir ces informations instantanément ou attendre de recevoir une réponse du fournisseur de produits et que toutes les informations soient associées puis fournies.

Ayant déterminé que le fournisseur de produits est une source d'informations pertinente pour la ressource, sur la base de son catalogue stockée, le fournisseur ISP vérifie sa mémoire cache pour vérifier si les informations sont disponibles localement. Si ce n'est pas le cas, il demande au fournisseur toutes les informations disponibles, qu'il fournit ensuite à l'agent et met éventuellement en mémoire cache dans son dépôt.

L'utilisateur peut, par le biais d'une interface vers l'agent et de façon indépendante, localiser une source utile d'informations de vulnérabilité et demander à l'agent, en utilisant éventuellement une demande déjà conçue adressée à un fournisseur (par exemple le fournisseur 2) et qu'il peut avoir téléchargée, de demander un catalogue ou les informations de vulnérabilité déjà disponibles ou dès qu'elles le seront dans l'avenir.

Après la mise à niveau d'une ressource ou d'un de ses composants, les demandes de mises à jour de la version précédente sont annulées à l'aide du message *Register:End* et les demandes de mises à jour des informations pertinentes pour les nouvelles versions peuvent être faites via le message *Register:null*. De même, si une ressource est supprimée du service, de futures mises à jour d'informations peuvent être annulées par le biais également du message *Register:End*.

8.3 Considérations liées à la sécurité et au respect de la vie privée

Dans le cadre de la présente Recommandation, les informations d'utilisateur ne sont pas transmises du côté fournisseur de produits. Toutefois, il est nécessaire de procéder à une authentification et à une vérification d'intégrité des informations fournies, via les méthodes suggérées suivantes:

- Les sources devraient s'identifier en utilisant les signatures dans les messages.
- Les agents devraient vérifier l'authenticité et l'intégrité des messages fournis.
- Les mises à jour ne doivent pas être individualisées (personnalisées), pour qu'aucune information propre au système d'un utilisateur final ne soit transmise dans le réseau.
- Le téléchargement ou l'installation sur un équipement d'utilisateur final nécessite l'aval de l'utilisateur final.

9 Architecture recommandée

9.1 Couche cœur de message

La couche cœur de message permet une communication indépendante du POAS (plate-forme/système d'exploitation/application/service) entre un serveur conforme et des applications clients conçues pour répondre aux prescriptions POAS. Elle renvoie à des mises en œuvre spécifiques d'un message les complexités liées à la manière de réaliser une opération donnée, l'administrateur n'ayant alors plus qu'à décider du choix des opérations à réaliser.

9.2 Couche message/application

La couche message/application comprend deux éléments principaux. Le premier correspond à des processus ou des fonctions indépendants du POAS généralement effectués durant les opérations de ressource ou de dispositif (par exemple pour la demande d'informations de vulnérabilité, de mises à jour ou de corrections). Il s'agit d'opérations de haut niveau qui s'appliquent aux systèmes et aux fonctions sans qu'intervienne la façon dont un système donné est mis en œuvre. Par exemple, tous les systèmes nécessitent périodiquement des mises à jour ou des corrections ainsi que des informations de vulnérabilité qui les concernent et dont les utilisateurs devraient avoir connaissance, à mesure qu'elles deviennent disponibles.

Le problème est que pratiquement tous les systèmes réalisent exactement les mêmes fonctions de haut niveau de façons totalement différentes du point de vue de la mise en œuvre. Dans l'architecture de la présente Recommandation, c'est au niveau de cette couche, lorsqu'il y a conflit entre les prescriptions indépendantes du POAS et les mises en œuvre de niveau inférieur, que l'interface principale est créée.

En général, un identificateur est attribué aux opérations/processus de haut niveau qui doivent être exécutés sur un système indépendamment de son type. Des systèmes de types différents utilisent ensuite ces identificateurs pour échanger des instructions. Après avoir reçu une instruction, le système récepteur la transmet simplement, avec toute donnée associée, à un gestionnaire spécifique de système attribué chargé de la traiter.

Par bien des aspects, il n'y a pas de différence entre le traitement et la restitution d'un flux vidéo compressé et la lecture d'un fichier tel que celui du présent document sur une multiplicité de plates-formes différentes. Chaque lecteur vidéo ou lecteur d'un fichier document dépendant de la plate-forme est responsable du traitement des données reçues sur la base des normes considérées.

9.3 Echelonnabilité

L'échelonnabilité des systèmes mis en œuvre conformément à la présente Recommandation est prise en charge via l'architecture d'identification serveur/client topographique qui permet à un client d'un serveur d'être un serveur d'autres clients situés au-dessous de lui. De fait, tout groupement topographique de serveurs et de clients pouvant être décrit par des nœuds d'une arborescence est pris en charge. De plus, la capacité de grouper des messages à fournir selon l'appartenance à un client ou un groupe spécifique permet à tout serveur d'un niveau quelconque au-dessus d'un client donné de pouvoir communiquer et interagir avec lui.

- Les clients et les serveurs sont identifiés par un identificateur GUID à 128 bits qui leur est attribué.
- Un identificateur GUID peut être préattribué à un client et/ou lui être réattribué lors de son enregistrement auprès d'un serveur donné.
- Un client peut selon la politique topographique suivie se voir attribuer/utiliser différents identificateurs GUID pour interagir avec différents serveurs.
- Dans le cas où un client est lui-même serveur d'autres clients (en tant que serveur local), ce serveur local peut "occulter" le véritable identificateur GUID du serveur auquel il rend compte (et inversement).
- Les clients peuvent faire rapport/être attribués à plusieurs serveurs tous chargés de fournir un éventail de services spécifiques.

9.4 Extensibilité

On peut ajouter simplement des nouveaux plates-formes, applications, fonctions et services en créant un schéma déduit, en l'étendant, du schéma cœur de message de la présente Recommandation.

A partir d'un schéma de message défini, on peut ensuite créer des modules pour une plate-forme quelconque souhaitée en étant assuré d'une interopérabilité intégrale.

En se fondant sur des structures de messages et de données de message d'un nouveau schéma de message, on peut facilement créer des interfaces d'utilisateur "sur-le-champ" comme nécessaire ou souhaité.

On peut "réutiliser" des messages communs pour de nouveaux ensembles de messages en important la définition de message existante, la structure de données de message ou les deux (selon qu'il convient).

9.5 Indépendance vis-à-vis de la plate-forme

- La présente Recommandation est l'interface "normative" entre toutes les entités participantes.
- Des modules mettant en œuvre la présente Recommandation sont créés en fonction du POAS considéré pour mettre en œuvre les messages, les fonctions et les services requis.
- Tout dispositif participant peut émettre un message ou une demande spécifié quelconque vers n'importe quel autre dispositif participant indépendamment des plates-formes sur lesquelles ces dispositifs sont mis en œuvre. Si, dans certains cas, des données de message contenues peuvent contenir ou contiendront probablement un contenu spécifique de la plate-forme considérée, la présente Recommandation vise dans tous les cas à supprimer la nécessité de connaître les autres plates-formes. Elle définit seulement ce qu'un dispositif peut demander de faire à un autre dispositif, mais ne décrit pas la façon de le faire.

9.6 Communication client/serveur

9.6.1 Protocole public

- De type XML

9.6.2 Modes de communication serveur-client

- Diffusion par le serveur ("Server Push")
  - Le serveur envoie au client les notifications et les mises à jour pour lesquelles celui-ci s'est enregistré, à mesure que ces notifications et mises à jour sont disponibles.
- Extraction par le serveur ("Server Pull")
  - Le serveur envoie des demandes d'opérations, de notifications et de mises à jour au client.
- Diffusion par le client ("Client Push")
  - Le client envoie des notifications et des mises à jour au serveur.
- Extraction par le client ("Client Pull")
  - Le client envoie des demandes d'opérations, de notifications et de mises à jour au serveur.

10 Composants du cadre

La présente Recommandation est définie en utilisant le langage XML et comprend une architecture de messagerie, qui prend en charge l'adressage des sources et des destinations de messages ainsi que le transport de messages et des données de message associées.

Le présent paragraphe fournit les définitions du format de données des principaux composants: le conteneur de messages, le message d'inscription et le message de version.

10.1 Conteneur de messages

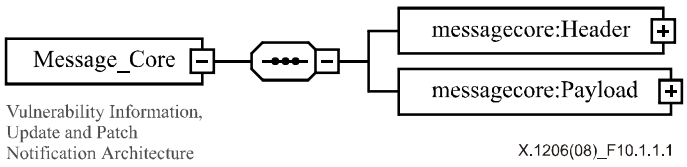
Le conteneur de messages sert à la fois d'enveloppe pour le transport, le routage et la fourniture de messages et de réponses ainsi que de modèle à partir duquel les autres messages et réponses peuvent être créés par extension et déduction.

10.1.1 Message_Core

Tous les messages utilisés dans l'architecture de la présente Recommandation sont définis par importation de *Message_Core*, extension et déduction.

Message_Core est l'élément racine de la structure de message.

10.1.1.1 Syntaxe



10.1.1.2 Sémantique

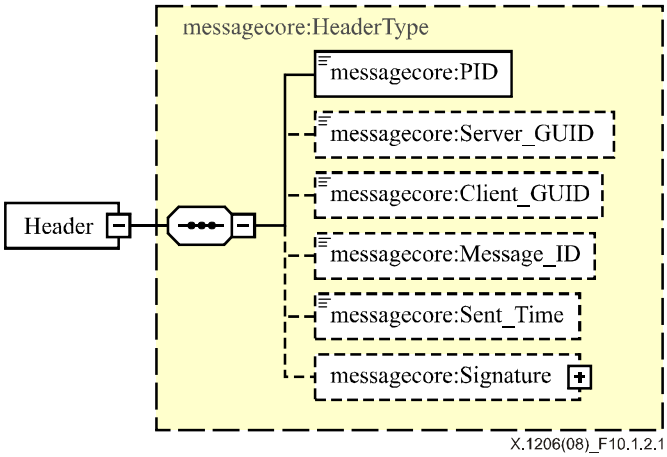
Header – L'en-tête contient des informations générales de routage et d'identification de message. [OBLIGATOIRE]

Payload – La charge utile contient un **choix** XML d'un ou plusieurs messages *Message_Core* groupés, ou un seul message. A utiliser lorsque les messages peuvent se trouver en mémoire cache au niveau d'un nœud donné (serveur local) ou lorsqu'un serveur assurant le contrôle/le service d'un serveur local peut souhaiter recevoir des messages ou des données de message en provenance de clients vis-à-vis desquels le serveur local assure un contrôle/un service. [OBLIGATOIRE]

10.1.2 Header

L'en-tête de message contient un identificateur de protocole à des fins d'extensibilité et d'interopérabilité, des informations d'adressage telles que source(s) et destination(s) pour un message donné, un champ pour acheminer un identificateur de message et un champ pour acheminer des informations sur la date d'origine du message.

10.1.2.1 Syntaxe



10.1.2.2 Sémantique

PID – Identificateur de protocole. Valeur codée sur 128 bits utilisée pour identifier une version et/ou une extension spécifique du protocole de message ou remplacement complet de ce dernier avec au minimum importation de *Message_Core*, *Header* et *PID*. Pour signaler la conformité à la

présente Recommandation, les applications peuvent utiliser la valeur #h00000001. [OBLIGATOIRE]

Server_GUID – Valeur codée sur 128 bits utilisée par les clients pour identifier de manière unique un serveur ou un serveur local auquel ils se réfèrent directement. [FACULTATIF]

Client_GUID – Valeur codée sur 128 bits utilisée par les serveurs et les serveurs locaux pour identifier de manière unique les clients qui s'adressent à eux. [FACULTATIF]

Message_ID – Valeur codée sur 128 bits utilisée pour identifier de manière unique un message donné. Bien que la méthode utilisée pour sélectionner un identificateur *Message_ID* approprié ne soit pas normative, un système homologue donné devrait utiliser la même valeur *Message_ID* dans les réponses ultérieures. [FACULTATIF]

Sent_Time – Contient l'heure à laquelle un message donné a été envoyé. [FACULTATIF]

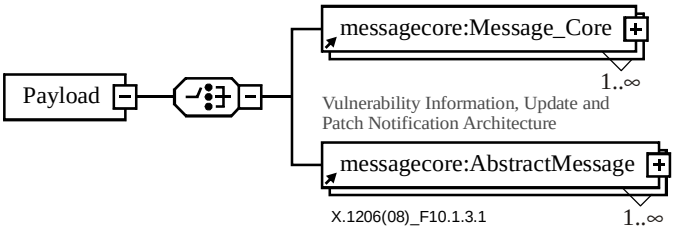
Signature – Signature XML enveloppée. [FACULTATIF]

10.1.3 Payload

La fonctionnalité associée au contenu d'une charge utile *Payload* donnée est définie par le biais de la définition et de l'association d'un identificateur *UUID* et/ou *CID* avec un élément *Message* et un élément *Message_Data* associé à l'élément *Message*. Les éléments *UUID/CID*, *Message* et *Message_Data* sont les éléments clés qui assurent l'extensibilité de l'architecture de la présente Recommandation et lui confèrent son indépendance vis-à-vis des plates-formes considérées.

En définissant et en associant un identificateur *UUID* et/ou un identificateur *CID* à la définition d'un schéma XML d'ensemble de message, qui importe et étend le schéma *Message_Core* et, en mettant en œuvre un module pour un ensemble de message donné qui inclut les schémas *Message* et *Message_Data*, on peut charger des modules dans toute application conforme à la présente Recommandation avec une garantie d'interopérabilité intégrale.

10.1.3.1 Syntaxe



10.1.3.2 Sémantique

Message_Core – Assure la capacité de grouper et de fournir plusieurs messages en provenance de plusieurs sources. Si choisi [OBLIGATOIRE].

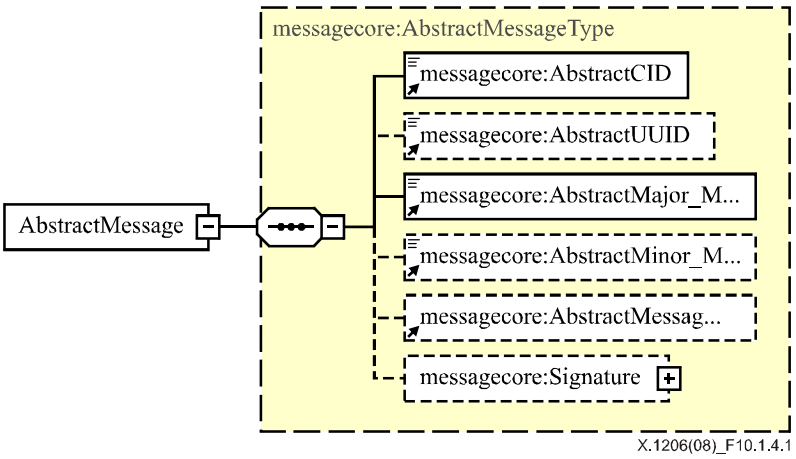
AbstractMessage – Message et identificateur de module appropriés et indication de type de message spécifique ainsi que données spécifiques de message. [OBLIGATOIRE]

10.1.4 AbstractMessage

La structure de données AbstractMessage est utilisée pour acheminer des informations en vue d'identifier exactement le message et le mode de message demandés mais aussi d'acheminer toutes les données nécessaires pour permettre l'acheminement du message demandé.

Cette structure de données est censée être étendue par tout message spécifique défini via un schéma normatif. Du fait de sa mise en œuvre dans le schéma central en tant que classe abstraite, elle ne peut pas être utilisée sans une extension de définition.

10.1.4.1 Syntaxe



10.1.4.2 Sémantique

AbstractCID – Valeur codée sur 128 bits utilisée pour identifier de manière unique une fonctionnalité donnée unique. Un module mis en œuvre donné, identifié par un identificateur *UUID* donné, peut permettre la prise en charge de plusieurs types de fonctionnalités identifiées par un identificateur *CID*. Inversement, plusieurs modules peuvent mettre en œuvre une même fonctionnalité mais ceci pour différentes applications (par exemple un module de version de traitement de texte et un module de version d'éditeur vidéo). [OBLIGATOIRE]

AbstractUUID – Valeur codée sur 128 bits utilisée pour identifier de manière unique un module mis en œuvre capable de traiter des *Messages* auquel son identificateur est associé. [FACULTATIF]

AbstractMajor_Message – Chaîne utilisée pour acheminer des instructions de serveurs vers des clients ou inversement. Ces instructions sont définies dans les schémas qui importent et étendent le schéma *Message_Core* et peuvent inclure, sans s'y limiter, les messages "Register", "Request", etc. [OBLIGATOIRE]

AbstractMinor_Message – Chaîne d'identificateurs de message additionnelle pour définir plus précisément le mode du message indiqué. Ces instructions sont définies dans les schémas qui importent et étendent le schéma et les modes *Message_Core* mais ne sont pas limitées à "Full", "Catalog", etc. [FACULTATIF]

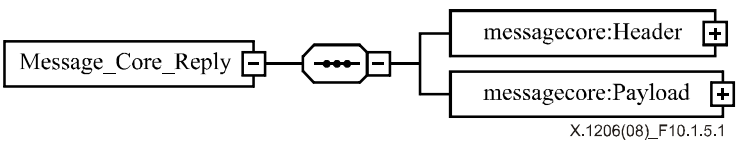
AbstractMessage_Data – Données requises pour le traitement du message demandé. [FACULTATIF]

Signature – Signature XML enveloppée. [FACULTATIF]

10.1.5 Message_Core_Reply

La structure de données *Message_Core_Reply* est utilisée pour acheminer une réponse relative à l'état et toute donnée additionnelle souhaitée vers toute entité à l'origine d'un flux de communication donné.

10.1.5.1 Syntaxe



10.1.5.2 Sémantique

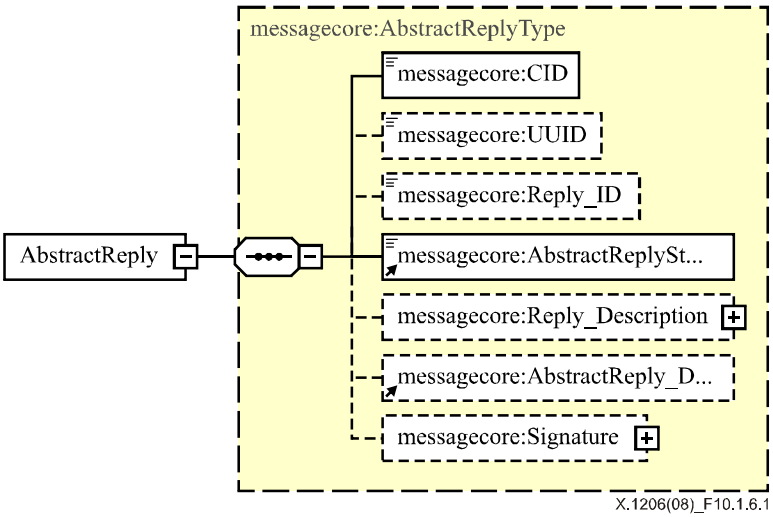
Header – Structure, données et prescriptions identiques à celles de Message_*Core:Header*.

Payload – Contient un **choix** XML d'un ou plusieurs messages *Message_Core_Reply* groupés ou un seul message *Message_Core_Reply*. A utiliser lorsque les messages peuvent être mis en mémoire cache au niveau d'un nœud donné (serveur local) ou lorsqu'un serveur assurant le contrôle/le service d'un serveur local peut souhaiter recevoir des messages ou des données de message de la part de clients vis-à-vis desquels le serveur local assure un contrôle/un service. [OBLIGATOIRE]

10.1.6 Abstract_Reply

La structure de données *Abstract_Reply* est utilisée pour acheminer une réponse relative à l'état et toute donnée additionnelle souhaitée vers toute entité à l'origine d'un flux de communication donné.

10.1.6.1 Syntaxe



10.1.6.2 Sémantique

CID – Valeur codée sur 128 bits utilisée pour identifier une version et/ou une extension spécifique du protocole de message ou remplacement complet de ce dernier avec au minimum importation de *Message_Core*, *Header* et *PID*. Pour signaler la conformité à la présente Recommandation, les applications peuvent utiliser la valeur #h00000001. [OBLIGATOIRE]

UUID – Valeur codée sur 128 bits utilisée pour identifier de manière unique un module mis en œuvre capable de traiter des réponses auxquelles son identificateur est associé. [FACULTATIF]

Reply_ID – Valeur codée sur 128 bits utilisée pour identifier de manière unique une réponse donnée. Bien que la méthode utilisée pour sélectionner un identificateur *Reply_ID* approprié ne soit pas normative, un système homologue donné devrait utiliser la même valeur *Message_ID* dans les demandes reçues dont ce message est la réponse. [FACULTATIF]

AbstractReplyString – Elément étendu par les messages important le schéma *Message_Core* et qui contient des chaînes indiquant le statut de la demande précédente pour laquelle la réponse est envoyée. Ces chaînes peuvent être, mais sans s'y limiter, les suivantes: "Failed", "Succeeded", "Message not supported", etc. [OBLIGATOIRE]

Reply_Description – Texte simple ou code HTML donnant des informations supplémentaires concernant la chaîne *AbstractReplyString*. [FACULTATIF]

AbstractReply_Data – Toute mise en œuvre dépendante des données requises. [FACULTATIF]

Signature – Signature XML enveloppée. [FACULTATIF]

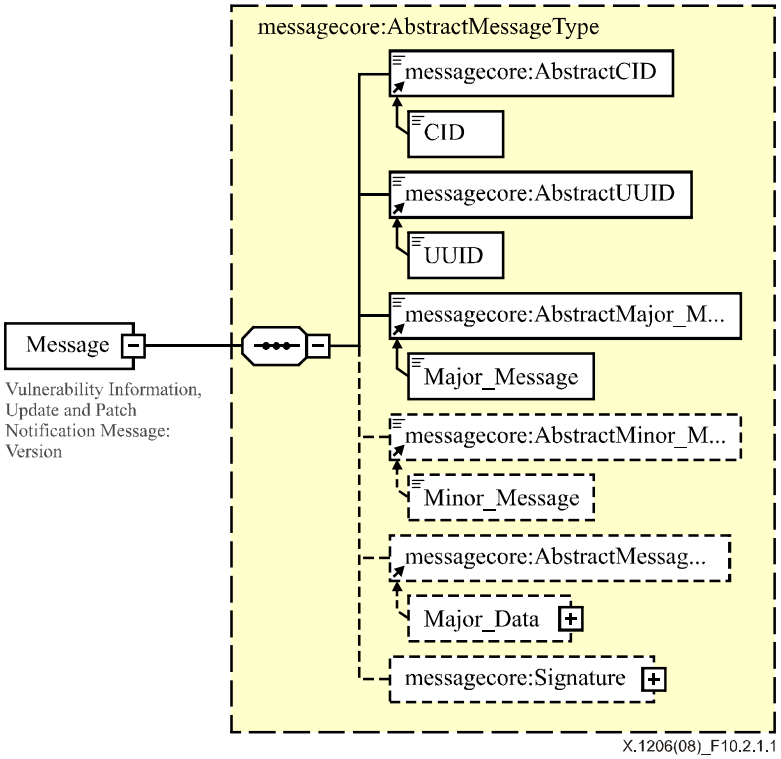
10.2 Message de version

Le message de version est utilisé entre composants d'un système pour demander et mettre à disposition des informations de vulnérabilité ainsi que des mises à jour et/ou des corrections.

10.2.1 Version:Message

Version:Message doit permettre un remplacement direct partout où *AbstractMessage* existe et sa structure de données est restreinte et étendue pour prendre en charge de façon unique le processus de mise à disposition, de demande et de fourniture d'informations de vulnérabilité, de mises à jour et/ou de corrections.

10.2.1.1 Syntaxe



10.2.1.2 Sémantique

CID – supplante et restreint la classe de base *Message_Core:AbstractCID* par le biais d'une valeur assignée spécifique #h02. Il identifie de manière unique ce message comme appartenant à la classe message de version. [OBLIGATOIRE]

UUID – Supplante la classe de base *Message_Core:AbstractUUID* du message et est utilisée pour identifier de manière unique un module mis en œuvre capable de traiter les *Messages* auxquels son identificateur est associé. [OBLIGATOIRE]

Major_Message – Supplante et restreint la classe de base *Message_Core:AbstractMajor_Message* du message à l'aide d'une liste énumérant les valeurs possibles "Register", "Request", "Deliver" et "Analyse". [OBLIGATOIRE]

Les valeurs sont utilisées comme suit:

- "Register" – Notifier au client récepteur que la ressource et sa version décrite dans *Version:Message_Data* sont utilisées et que les informations de vulnérabilité, les mises à jour et les corrections devraient être fournies dès qu'elles sont disponibles.
- "Request" – Demander les informations de vulnérabilité, les mises à jour et les corrections disponibles pour tout produit qui au minimum répond aux prescriptions spécifiées dans *Version:Message_Data*.
- "Deliver" – Les informations de vulnérabilité ainsi qu'une ou plusieurs mises à jour ou corrections sont incluses dans *Version:Message_Data.*

- *"Analyse"* – Les informations figurant dans *Version:Message_Data* sont demandées pour être analysées. Le motif de la demande peut être connu *a priori* ou une explication peut être donnée dans *version:Info – version:Description ou version:Info – version:Container – version:Description*, suivant ce qui convient le plus pour l'application considérée.

Minor_Message – Supplante et restreint la classe de base *Message_Core:AbstractMinor_Message* par le biais d'une liste énumérant les valeurs possibles "Full", "Catalog" et "End". [FACULTATIF]

Les valeurs utilisées sont les suivantes:

- "Full" – Demander toutes les mises à jour et/ou corrections disponibles et applicables.
- "Catalog" – Fournir une liste d'informations de vulnérabilité, de mises à jour et/ou de corrections disponibles pour les produits spécifiés dans *Version:Message_Data*.
- "End" – Mettre fin à un abonnement conformément aux informations qui figurent dans *Version:Message_Data*.

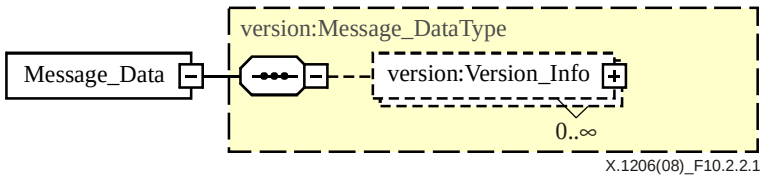
Message_Data – Supplante et restreint la classe de base *Message_Core:AbstractMessage_Data* à l'aide d'une structure de données spécifique utilisée pour acheminer l'identification de la version de produit et acheminer toutes les informations de vulnérabilité, mises à jour et/ou corrections pertinentes. [FACULTATIF]

Signature – Signature XML enveloppée. [FACULTATIF]

10.2.2 Version:Message_Data

L'élément *Version:Message_Data* est conçu pour permettre un remplacement direct partout où l'élément *AbstractMessage_Data* existe et sa structure de données est à la fois restreinte et étendue pour prendre en charge de façon unique le processus d'acheminement de la version du produit, des informations de vulnérabilité, des mises à jour et/ou des corrections.

10.2.2.1 Syntaxe



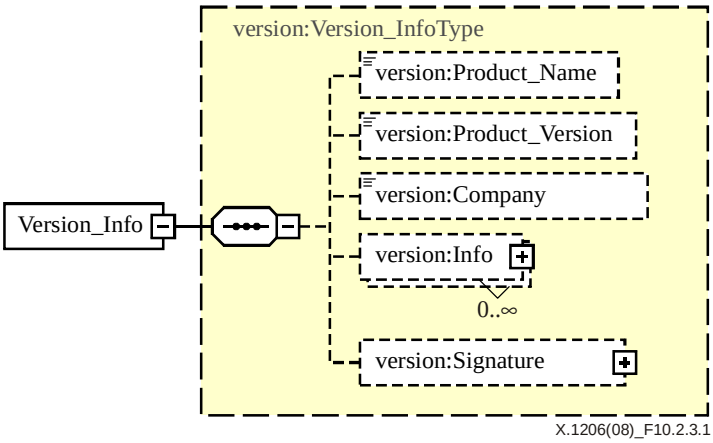
10.2.2.2 Sémantique

Version_Info – Informations utilisées pour identifier, au niveau de spécificité requis, les produits et leurs versions et prendre en charge l'acheminement des informations de vulnérabilité, des mises à jour et/ou des corrections.

10.2.3 Version_Info

Cet élément est utilisé non seulement pour identifier les produits, avec leur fournisseur, leur version et les informations relatifs à la version, mais aussi pour acheminer des informations de vulnérabilité, des mises à jour et des corrections aux applications qui en ont fait la demande.

10.2.3.1 Syntaxe



10.2.3.2 Sémantique

Product_Name – Nom du produit de haut niveau auquel se rapportent les informations de version identifiées (par exemple, dans le cas d'un lecteur de DVD qui contient une version de système flash spécifique, *Product_Name* ferait référence au nom du produit de lecteur DVD). [FACULTATIF]

Product_Version – Version du produit de haut niveau à laquelle se rapportent les informations de version identifiées (dans l'exemple précédent, *Product_Version* correspondrait à la version spécifique du lecteur DVD).

Company – Fournisseur de produits, fournisseur ou auteur du produit ou de la ressource identifiée. [FACULTATIF]

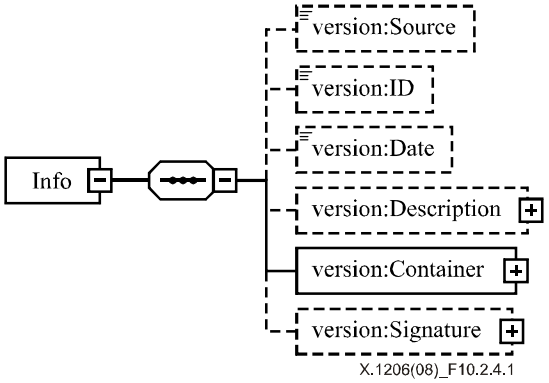
Info – Structure de données utilisée pour acheminer des informations de vulnérabilité pour des mises en œuvre effectives de produits ou de ressources comme indiqué par les valeurs des champs précédents. [FACULTATIF]

Signature – Signature XML enveloppée. [FACULTATIF]

NOTE – Les informations contenues dans le champ Info sont toujours utilisées pour identifier une ressource donnée, les champs *Product_Name* et *Product_Version* n'étant utilisés que si l'on en ressent le besoin. Toutefois, lorsque la ressource identifiée dans le champ Info est un produit de haut niveau, l'utilisation des champs *Product_Name* et *Product_Version* est conseillée pour indiquer clairement qu'il n'existe pas de "parent" pour la ressource identifiée.

10.2.4 Info

10.2.4.1 Syntaxe



10.2.4.2 Sémantique

Source – Description textuelle (en utilisant par exemple un nom) du fournisseur de produits ou du fournisseur d'informations de vulnérabilité fournissant le code inclus ou les informations de vulnérabilité. [FACULTATIF]

ID – Identificateur grâce auquel l'ensemble de données *Info* considéré peut être identifié. Même si aucune méthode d'assignation n'est spécifiée, il est proposé que les identificateurs assignés ne soient uniques que dans le cadre d'une série pour une ressource donnée en provenance d'une *Source* donnée. [FACULTATIF]

Date – Valeur de date XML relative aux données *Info*. [FACULTATIF]

Description – Description textuelle ou HTML du code contenu dans une instance facultative du champ *Info_Container* ou d'une référence d'identificateur URI de ce dernier pour la version de ressource spécifique décrite dans une instance spécifique de cette structure de données. [FACULTATIF]

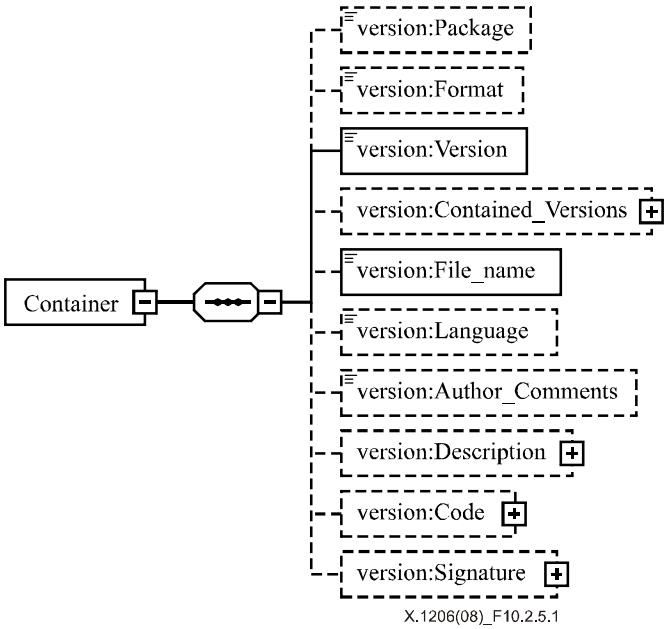
Container – Structure de données pour l'acheminement du code sous la forme d'une mise à jour ou d'une correction. [OBLIGATOIRE]

Signature – Signature XML enveloppée. [FACULTATIF]

10.2.5 Conteneur

Il contient des informations d'identification de version ainsi que des informations de vulnérabilité, une mise à jour ou une correction.

10.2.5.1 Syntaxe



10.2.5.2 Sémantique

Package – Méthode de paquetage utilisée pour la version spécifique et/ou le contenu de *Code.* [FACULTATIF]

Format – Format de la version spécifique. [FACULTATIF]

Version – Version applicable, incluse pour les demandes ou utilisée pour identifier le contenu du champ *Version_Control* et/ou du champ *Code*. [OBLIGATOIRE]

Contained_Versions – Liste, définie en utilisant la structure de données *Version_Info* (voir ci-dessus), qui contient la description de tous les composants de la ressource ou du composant considéré. [FACULTATIF]

File_Name – Nom du fichier. [FACULTATIF]

Language – Dans le cas de ressources pour lesquelles il existe différentes versions pour différents pays sur la base de la langue utilisée, ce champ est utilisé pour identifier la langue. [FACULTATIF]

Author_Comments – Tout contenu textuel souhaité. [FACULTATIF]

Description – Description textuelle ou HTML de la version spécifique contenue dans le champ *Code* ou description textuelle ou HTML des informations de vulnérabilité relatives à la ressource identifiée par le champ *Version* et/ou le champ *File_Name* utilisés par ou dans les champs *Version_Info: Product_Name* et *Version_Info: Product_Version*.

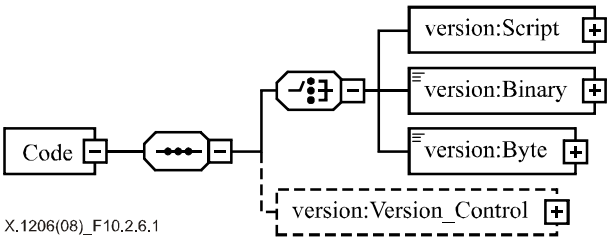
Code – Conteneur du code exécutable et des informations de contrôle de version.

Signature – Signature XML enveloppée. [FACULTATIF]

10.2.6 Code

Conteneur du code ou des chaînes d'octets effectifs (en tant qu'image mémoire par exemple) ou des références URL vers ces éléments, ainsi que d'informations utiles pour le contrôle des mises à jour de version.

10.2.6.1 Syntaxe



10.2.6.2 Sémantique

Script – Code sous forme de script. [EVENTUELLEMENT OBLIGATOIRE]

Binary – Code binaire compilé. [EVENTUELLEMENT OBLIGATOIRE]

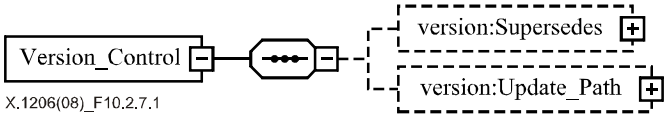
Byte – Données fournies pour l'analyse. [EVENTUELLEMENT OBLIGATOIRE]

Version_Control – Structure de données utilisée pour indiquer les liens entre les versions successives et leurs chemins de mise à jour possibles. [FACULTATIF]

10.2.7 Version_Control

Il s'agit d'un conteneur pour l'acheminement de listes de séquences de versions remplacées par une version donnée ainsi qu'une liste identifiant les versions auxquelles la mise à jour ou la correction identifiée peut être appliquée.

10.2.7.1 Syntaxe



10.2.7.2 Sémantique

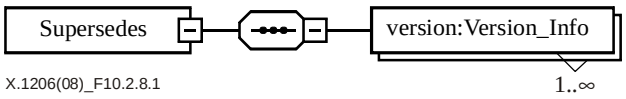
Supersedes – Liste séquencée de versions supplantées par *Version_Info*. [FACULTATIF]

Update_Path – Liste de versions auxquelles la mise à jour ou la correction actuelle peut être appliquée. [FACULTATIF]

10.2.8 Supersedes

Liste de ressources supplantées par la version spécifiée.

10.2.8.1 Syntaxe



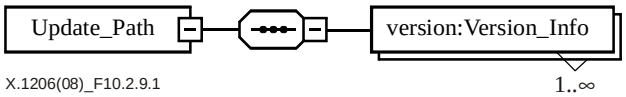
10.2.8.2 Sémantique

Version_Info – (voir *info*).

10.2.9 Update_Path

Liste de ressources auxquelles la mise à jour ou la correction contenue peut être appliquée.

10.2.9.1 Syntaxe



X.1206(08)_F10.2.9.1

10.2.9.2 Sémantique

Version_Info – (voir *info*).

10.2.10 ReplyStatus

10.2.10.1 Syntaxe

X.1206(08)_F10.2.10.1

10.2.10.2 Sémantique

ReplyStatus – Contient l'un des éléments suivants:

"Failed" – L'opération demandée n'a pu être achevée.

"Succeeded" – L'opération demandée a réussi.

"Major Message Not Supported" – L'identificateur de message majeur demandé ou son utilisation dans le contexte donné n'est pas pris en charge.

"Minor Message Not Supported" – L'identificateur de message mineur demandé ou son utilisation dans le contexte donné n'est pas pris en charge.

"Handler not available" – Le type de message identifié n'est pas pris en charge.

11 Schémas

11.1 Message_Core

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:messagecore="http://www.itu.int//xml-namespace/itu-t/x.1206/CORE/" xmlns:ns1="http://www.w3.org/2000/09/xmldsig#" targetNamespace=" http://www.itu.int//xml-namespace/itu-t/x.1206/CORE/" elementFormDefault="qualified" attributeFormDefault="unqualified">

    <xs:import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="xmldsig-core-schema.xsd"/>

    <xs:element name="Message_Core">

        <xs:annotation>

            <xs:documentation>Vulnerability Information, Update and Patch Notification Architecture</xs:documentation>

        </xs:annotation>

        <xs:complexType>

            <xs:sequence>

                <xs:element name="Header" type="messagecore:HeaderType"/>

                <xs:element name="Payload">

                    <xs:complexType>

                        <xs:choice>

                            <xs:element ref="messagecore:Message_Core" maxOccurs="unbounded"/>

                            <xs:element ref="messagecore:AbstractMessage" maxOccurs="unbounded"/>

                        </xs:choice>

                    </xs:complexType>

                </xs:element>
```

```
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="HeaderType">
        <xs:sequence>
            <xs:element name="PID" type="xs:string"/>
            <xs:element name="Server_GUID" type="xs:string" minOccurs="0"/>
            <xs:element name="Client_GUID" type="xs:string" minOccurs="0"/>
            <xs:element name="Message_ID" type="xs:string" minOccurs="0"/>
            <xs:element name="Sent_Time" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="Signature" type="ns1:SignatureType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="AbstractMessage" type="messagecore:AbstractMessageType" abstract="true"/>
    <xs:complexType name="AbstractMessageType">
        <xs:sequence>
            <xs:element ref="messagecore:AbstractCID"/>
            <xs:element ref="messagecore:AbstractUUID" minOccurs="0"/>
            <xs:element ref="messagecore:AbstractMajor_Message"/>
            <xs:element ref="messagecore:AbstractMinor_Message" minOccurs="0"/>
            <xs:element ref="messagecore:AbstractMessage_Data" minOccurs="0"/>
            <xs:element name="Signature" type="ns1:SignatureType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="AbstractCID" type="messagecore:AbstractCIDType" abstract="true"/>
    <xs:complexType name="AbstractCIDType">
        <xs:simpleContent>
            <xs:extension base="xs:string"/>
        </xs:simpleContent>
    </xs:complexType>
    <xs:element name="AbstractUUID" type="messagecore:AbstractUUIDType" abstract="true"/>
    <xs:complexType name="AbstractUUIDType">
        <xs:simpleContent>
            <xs:extension base="xs:string"/>
        </xs:simpleContent>
    </xs:complexType>
    <xs:element name="AbstractMajor_Message" type="messagecore:AbstractMajorMessageType" abstract="true"/>
    <xs:complexType name="AbstractMajorMessageType">
        <xs:simpleContent>
            <xs:extension base="xs:string"/>
        </xs:simpleContent>
    </xs:complexType>
    <xs:element name="AbstractMinor_Message" type="messagecore:AbstractMinorMessageType" abstract="true"/>
    <xs:complexType name="AbstractMinorMessageType">
        <xs:simpleContent>
            <xs:extension base="xs:string"/>
        </xs:simpleContent>
    </xs:complexType>
```

```
    <xs:element name="AbstractMessage_Data" type="messagecore:AbstractMessage_DataType" abstract="true"/>
    <xs:complexType name="AbstractMessage_DataType"/>
    <xs:element name="Message_Core_Reply">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Header" type="messagecore:HeaderType"/>
                <xs:element name="Payload">
                    <xs:complexType>
                        <xs:choice>
                            <xs:element ref="messagecore:Message_Core_Reply" maxOccurs="unbounded"/>
                            <xs:element ref="messagecore:AbstractReply"/>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="AbstractReply" type="messagecore:AbstractReplyType"/>
    <xs:complexType name="AbstractReplyType">
        <xs:sequence>
            <xs:element name="CID" type="messagecore:AbstractCIDType"/>
            <xs:element name="UUID" type="messagecore:AbstractUUIDType" minOccurs="0"/>
            <xs:element name="Reply_ID" type="xs:string" minOccurs="0"/>
            <xs:element ref="messagecore:AbstractReplyStatus"/>
            <xs:element name="Reply_Description" type="messagecore:Description_Type" minOccurs="0"/>
            <xs:element ref="messagecore:AbstractReply_Data" minOccurs="0"/>
            <xs:element name="Signature" type="ns1:SignatureType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="AbstractReplyStatus" type="messagecore:AbstractReplyStatusType" abstract="true"/>
    <xs:complexType name="AbstractReplyStatusType">
        <xs:simpleContent>
            <xs:extension base="xs:string"/>
        </xs:simpleContent>
    </xs:complexType>
    <xs:element name="AbstractReply_Data" type="messagecore:AbstractReply_DataType" abstract="true"/>
    <xs:complexType name="AbstractReply_DataType"/>
    <xs:complexType name="Description_Type">
        <xs:attribute name="ref" type="xs:anyURI" use="optional"/><![CDATA[]]></xs:complexType>
</xs:schema>
```

11.2 Message_Version

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:messagecore="http://www.itu.int//xml-namespace/itu-t/x.1206/CORE/" xmlns:version="http://www.itu.int//xml-namespace/itu-t/x.1206/CORE/MESSAGE/VERSION/" xmlns:xmldsig="http://www.w3.org/2000/09/xmldsig#" targetNamespace="http://www.itu.int//xml-namespace/itu-t/x.1206/CORE/MESSAGE/VERSION/" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:import namespace="http://www.example.com/CORE" schemaLocation="Message_Core.xsd"/>
    <xs:import namespace="http://www.w3.org/2000/09/xmldsig#" schemaLocation="xmldsig-core-schema.xsd"/>
```

```
<xs:element name="Message" type="messagecore:AbstractMessageType" substitutionGroup="messagecore:AbstractMessage">
    <xs:annotation>
        <xs:documentation>Vulnerability Information, Update and Patch Notification Message : Version</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="CID" type="messagecore:AbstractCIDType" substitutionGroup="messagecore:AbstractCID"/>
<xs:element name="UUID" type="messagecore:AbstractUUIDType" substitutionGroup="messagecore:AbstractUUID"/>
<xs:element name="Major_Message" type="version:Major_MessageType" substitutionGroup="messagecore:AbstractMajor_Message"/>
<xs:complexType name="Major_MessageType">
    <xs:simpleContent>
        <xs:restriction base="messagecore:AbstractMajorMessageType">
            <xs:enumeration value="Register"/>
            <xs:enumeration value="Request"/>
            <xs:enumeration value="Deliver"/>
            <xs:enumeration value="Analyse"/>
        </xs:restriction>
    </xs:simpleContent>
</xs:complexType>
<xs:element name="Minor_Message" type="version:Minor_MessageType" substitutionGroup="messagecore:AbstractMinor_Message"/>
<xs:complexType name="Minor_MessageType">
    <xs:simpleContent>
        <xs:restriction base="messagecore:AbstractMinorMessageType">
            <xs:enumeration value="Full"/>
            <xs:enumeration value="Catalog"/>
            <xs:enumeration value="End"/>
        </xs:restriction>
    </xs:simpleContent>
</xs:complexType>
<xs:element name="Message_Data" type="version:Message_DataType" substitutionGroup="messagecore:AbstractMessage_Data"/>
<xs:complexType name="Message_DataType">
    <xs:complexContent>
        <xs:extension base="messagecore:AbstractMessage_DataType">
            <xs:sequence>
                <xs:element name="Version_Info" type="version:Version_InfoType" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:element name="Reply" type="messagecore:AbstractReplyType" substitutionGroup="messagecore:AbstractReply"/>
<xs:element name="ReplyStatus" substitutionGroup="messagecore:AbstractReplyStatus">
    <xs:complexType>
        <xs:simpleContent>
            <xs:restriction base="messagecore:AbstractReplyStatusType">
                <xs:enumeration value="Failed"/>
                <xs:enumeration value="Succeeded"/>
                <xs:enumeration value="Major Message Not Supported"/>
                <xs:enumeration value="Minor Message Not Supported"/>
                <xs:enumeration value="Handler not available"/>
```

```
            </xs:restriction>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="Reply_Data" type="messagecore:AbstractReply_DataType" substitutionGroup="messagecore:AbstractReply_Data"/>
<xs:complexType name="Version_InfoType">
    <xs:complexContent>
        <xs:extension base="messagecore:AbstractReply_DataType">
            <xs:sequence>
                <xs:element name="Product_Name" type="xs:string" minOccurs="0"/>
                <xs:element name="Product_Version" type="xs:string" minOccurs="0"/>
                <xs:element name="Company" type="xs:string" minOccurs="0"/>
                <xs:element name="Info" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Source" type="xs:string" minOccurs="0"/>
                            <xs:element name="ID" type="xs:string" minOccurs="0"/>
                            <xs:element name="Date" type="xs:string" minOccurs="0"/>
                            <xs:element name="Description" minOccurs="0">
                                <xs:complexType>
                                    <xs:attribute name="ref" type="xs:anyURI" use="optional"/><![CDATA[]]></xs:complexType>
                            </xs:element>
                            <xs:element name="Container">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Package" type="xs:string" minOccurs="0"/>
                                        <xs:element name="Format" type="xs:string" minOccurs="0"/>
                                        <xs:element name="Version" type="xs:string"/>
                                        <xs:element name="Contained_Versions" type="version:Version_InfoType" minOccurs="0"/>
                                        <xs:element name="File_name" type="xs:string" minOccurs="0"/>
                                        <xs:element name="Language" type="xs:string" minOccurs="0"/>
                                        <xs:element name="Author_Comments" type="xs:string" minOccurs="0"/>
                                        <xs:element name="Description" type="messagecore:Description_Type" minOccurs="0"/>
                                        <xs:element name="Code" minOccurs="0">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:choice>
                                                        <xs:element name="Script">
                                                            <xs:complexType>
                                                                <xs:attribute name="ref" type="xs:anyURI" use="optional"/>
                                                            </xs:complexType>
                                                        </xs:element>
                                                        <xs:element name="Binary">
                                                            <xs:complexType>
                                                                <xs:simpleContent>
                                                                    <xs:extension base="xs:base64Binary">
                                                                        <xs:attribute name="ref" type="xs:anyURI" use="optional"/>
                                                                    </xs:extension>
```

```
                                        </xs:simpleContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="Byte">
                                    <xs:complexType>
                                        <xs:simpleContent>
                                            <xs:extension base="xs:base64Binary">
                                                <xs:attribute name="ref" type="xs:anyURI" use="optional"/>
                                            </xs:extension>
                                        </xs:simpleContent>
                                    </xs:complexType>
                                </xs:element>
                            </xs:choice>
                            <xs:element name="Version_Control" minOccurs="0">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Supersedes" minOccurs="0">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element name="Version_info"
type="version:Version_InfoType" maxOccurs="unbounded"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="Update_Path" minOccurs="0">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element name="Version_Info"
type="version:Version_InfoType" maxOccurs="unbounded"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Signature" type="xmldsig:SignatureType" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Signature" type="xmldsig:SignatureType" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Signature" type="xmldsig:SignatureType" minOccurs="0"/>
</xs:sequence>
</xs:extension>
```

```
            </xs:complexContent>
        </xs:complexType>
    </xs:schema>
```

Bibliographie

[b-UIT-T X.667] Recommandation UIT-T X.667 (2004) | ISO/CEI 9834-8:2005, *Technologies de l'information – Interconnexion des systèmes ouverts – Procédures opérationnelles des organismes d'enregistrement de l'OSI: génération et enregistrement des identificateurs uniques universels (UUID) et utilisation de ces identificateurs comme composants d'identificateurs d'objets ASN.1*.

[b-IETF RFC 3023] IETF RFC 3023 (2001), *XML Media Types*. <http://www.ietf.org/rfc/rfc3023.txt?number=3023>

[b-IETF RFC 3075] IETF RFC 3075 (2001), *XML-Signature Syntax and Processing*. <http://www.ietf.org/rfc/rfc3075.txt?number=3075>

[b-XML Datatypes] W3C Datatypes:2001, *XML Schema Part 2: Datatypes*, W3C Recommendation, Copyright © [2 May 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2001/RECxmlschema-2-20010502/.

[b-XML Signature] W3C Signature Schema:2001, *XML Signature Schema*, W3C Recommendation, Copyright © [1 March 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/xmldsigcore/xmldsig-core-schema.xsd.

[b-XML Structures] W3C XML Schema Part 1:2001, *XML Schema Part 1: Structures*, W3C Recommendation, Copyright © [2 May 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2001/RECxmlschema-1-20010502/.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
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	Gestion des télécommunications y compris le RGT et maintenance des réseaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Terminaux et méthodes d'évaluation subjectives et objectives
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, communication entre systèmes ouverts et sécurité**
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de prochaine génération
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse
Genève, 2009