Groupe de travail Réseau
Request for Comments : 3460
RFC mise à jour : 3060
Catégorie : En cours de normalisation

B. Moore, éditeur, IBM
janvier 2003

Traduction Claude Brière de L'Isle

Extensions au modèle d'informations de cœur de politique (PCIM)

Statut de ce mémoire
Le présent document spécifie un protocole Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et des suggestions pour son amélioration. Prière de se reporter à l'édition actuelle du STD 1 "Normes des protocoles officiels de l'Internet" pour connaître l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document spécifie un certain nombre de changements au modèle d'informations de cœur de politique (PCIM, *Policy Core Information Model*) [RFC3060]. Deux types de changements sont inclus. D'abord, plusieurs éléments complètement nouveaux sont introduits, par exemple, les classes de filtrage d'en-tête, qui étendent PCIM dans des domaines qu'il ne couvrait pas précédemment. Ensuite, il y a des cas où des éléments de PCIM (par exemple, les priorités de règle de politique) sont déconseillés, et des éléments de remplacement sont définis (dans ce cas, les priorités liées aux associations qui se réfèrent aux règles de politique). Les deux types de changements sont faits d'une façon telle que, dans la mesure du possible, l'interopérabilité avec les mises en œuvre du modèle PCIM d'origine sont préservées. Le présent document met à jour la [RFC3060].

Table des Matières

1. Introduction

Le présent document spécifie un certain nombre de changements au modèle d'informations de cœur de politique (PCIM, *Policy Core Information Model*) [RFC3060]. Deux types de changements sont inclus. D'abord, plusieurs éléments complètement nouveaux sont introduits, par exemple, les classes pour le filtrage d'en-têtes, qui étendent PCIM sur des zones qu'il ne couvrait pas précédemment. Ensuite, il y a des cas où des éléments de PCIM (par exemple, les priorités de règle de politique) sont déconseillées et des éléments de remplacement sont définis (dans ce cas, les priorités liées aux associations qui se réfèrent aux règles de politique). Les deux types de changements sont faits d'une façon telle que, dans la mesure du possible, l'interopérabilité avec les mises en œuvre du modèle PCIM d'origine est préservée.

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

2. Changements depuis la RFC3060

Le paragraphe 3.2 contient une courte discussion des changements faits par le présent document au modèle d'informations de la RFC3060. Voici une brève liste de ces changements :

1. PolicyRepository et ses associations est déconseillé et remplacé.
2. Précision et développement des façons dont PolicyRules et PolicyGroups sont agrégés.
3. Changement de la façon de représenter la prioritisation des PolicyRules, et introduction des stratégies spécifiées par l'administrateur pour l'évaluation des règles.
4. Développement du rôle de PolicyRoles, et introduction de moyens d'associer un PolicyRole à une ressource.
5. Introduction de conditions et d'actions de politique composées dans le modèle.
6. Introduction de variables et de valeurs dans le modèle.
7. Introduction de sous classes de variables et de valeurs pour le filtrage d'en-tête de paquet.
8. Introduction de classes pour le filtrage d'en-tête de paquet au niveau appareil.

3. Présentation des changements

3.1 Comment changer un modèle d'information

Le modèle d'informations de cœur de politique est étroitement aligné sur le modèle de politique de cœur de CIM du DMTF. Comme il n'y a pas d'ensemble de règles documenté à part pour spécifier les modèles d'informations de l'IETF comme PCIM, il est raisonnable d'examiner dans les spécifications de CIM ce qui peut servir de lignes directrices sur la façon de modifier et étendre le modèle. Parmi les règles CIM pour changer un modèle d'informations on trouve les suivantes. Noter que tout ce qu'on dit ici des "classes" s'applique aux classes d'association (y compris les agrégations) ainsi qu'aux classes non association.

- Des propriétés peuvent être ajoutées aux classes existantes.
- Des classes, et des propriétés individuelles, peuvent être marquées comme DÉCONSEILLÉES. Si il y a une caractéristique de remplacement pour la classe ou propriété déconseillée, elle est identifiée explicitement. Autrement, la notation "Non valeur" est utilisée. Dans le présent document, la notation "DÉCONSEILLÉ POUR <nom-de-caractéristique>" est utilisée pour indiquer qu'une caractéristique a été déconseillée, et pour identifier sa caractéristique de remplacement.
- Des classes peuvent être insérées dans la hiérarchie d'héritage au dessus de classes existantes, et des propriétés des classes existantes peuvent alors être "remontées" dans les nouvelles classes. L'effet net est que les classes existantes ont exactement les mêmes propriétés que ce qu'elles avaient avant, mais les propriétés sont héritées plutôt que définies explicitement dans les classes.
- De nouvelles sous-classes peuvent être définies en dessous des classes existantes.

3.2 Liste des changements au modèle

Les paragraphes qui suivent font un très bref survol des changements à PCIM définis dans PCIMe. Dans plusieurs cas, l'origine du changement est notée, comme pour QPIM [RFC3644], ICPM [RFC3585], ou QDDIM [RFC3670].

3.2.1 Changements à PolicyRepository

À cause du potentiel de confusion avec le composant de cadre de politique Policy Repository (du tableau à quatre faces : Policy Management Tool, Policy Repository, PDP, PEP) "PolicyRepository" est un mauvais nom pour la classe PCIM qui représente un conteneur d'éléments de politique réutilisables. Donc, la classe PolicyRepository *(répertoire de politiques)* est remplacée par la classe ReusablePolicyContainer *(conteneur de politiques réutilisables)*. Pour réaliser ce changement, il est nécessaire de déconseiller la classe PCIM PolicyRepository et ses trois associations, et de les remplacer par une nouvelle classe ReusablePolicyContainer et de nouvelles associations. Par un autre changement, les associations pour ReusablePolicyContainer sont élargies, pour permettre à un ReusablePolicyContainer de contenir tout élément de politique réutilisable. Dans PCIM, les seules associations définies pour un PolicyRepository étaient qu'il contienne des conditions et des actions de politique réutilisables.

3.2.2 Associations supplémentaires et éléments supplémentaires réutilisables

Les agrégations PolicyRuleInPolicyRule et PolicyGroupInPolicyRule ont, en effet, été importées de QPIM. ("En effet" parce que ces deux agrégations, comme les deux agrégations PolicyGroupInPolicyGroup et PolicyRuleInPolicyGroup de PCIM, sont combinées en une seule agrégation PolicySetComponent.) Ces agrégations rendent possible de définir de plus gros "tronçons" de politique réutilisable à placer dans un ReusablePolicyContainer. Ces agrégations introduisent aussi une nouvelle sémantique représentant les implications contextuelles d'une PolicyRule qui s'exécute dans la portée d'une autre PolicyRule.

3.2.3 Priorités et stratégies de décision

Empruntant à la fois à QPIM et à ICPM, la propriété Priority a été déconseillée dans PolicyRule, et plutôt placée dans

l'agrégation PolicySetComponent. Les règles QPIM pour la résolution des priorités relatives à travers les PolicyGroups et PolicyRules incorporés ont aussi été incorporées dans PCIMe. Avec le retrait de la propriété Priority de PolicyRule, une nouvelle dépendance de modélisation est introduite. Afin d'établir les priorités d'un PolicyRule/PolicyGroup par rapport aux autres PolicyRules/PolicyGroups, les éléments à classer doivent tous résider dans un de ces trois lieux : dans un PolicyGroup commun, dans une PolicyRule commune, ou dans un System commun.

En l'absence de tout critère général clair pour détecter les conflits de politique, la restriction de PCIM qui déclare que les priorités ne sont pertinentes que dans le cas de conflits est supprimée. À la place, une propriété PolicyDecisionStrategy a été ajoutée aux classes PolicyGroup et PolicyRule. Cette propriété permet à un administrateur de politique de choisir un des deux comportements par rapport à l'évaluation de règle : soit effectuer les actions pour toutes les PolicyRules dont les conditions s'évaluent à VRAI, soit n'effectuer les actions que pour la PolicyRule de plus forte priorité dont les conditions s'évaluent à VRAI. (Ceci est fait en plaçant la propriété PolicyDecisionStrategy dans une classe abstraite PolicySet, d'où sont déduits PolicyGroup et PolicyRule.) Les règles QPIM pour appliquer les stratégies de décision à un ensemble incorporé de PolicyGroups et PolicyRules ont aussi été importées.

3.2.4 Rôles de politique

Le concept de rôle de politique est ajouté à PolicyGroups (étant déjà présent dans la classe PolicyRule). Cela est fait via une nouvelle superclasse pour les PolicyRules et les PolicyGroups - PolicySet. Pour les PolicyRules et PolicyGroups incorporés, tous les rôles associés à la règle ou au groupe externe sont automatiquement "hérités" par celui qui est incorporé. Des rôles supplémentaires peuvent être ajoutés au niveau d'une règle ou groupe incorporé.

Il a aussi été observé qu'il n'y a pas de mécanisme dans PCIM pour assigner les rôles aux ressources. Par exemple, bien qu'il soit possible dans PCIM d'associer une PolicyRule au rôle "FrameRelay&&WAN", il n'y a pas de moyen d'indiquer quelles interfaces satisfont à ce critère. Une nouvelle classe PolicyRoleCollection a été définie dans PCIMe, pour représenter la collection des ressources associées à un rôle particulier. La liaison entre une PolicyRule ou un PolicyGroup et un ensemble de ressources est alors représentée par une instance de PolicyRoleCollection. Des valeurs équivalentes devraient être définies dans la propriété PolicyRoles des PolicyRules et PolicyGroups, et dans la propriété PolicyRole dans PolicyRoleCollection.

3.2.5 CompoundPolicyConditions et CompoundPolicyActions

Le concept d'une CompoundPolicyCondition *(condition de politique composée)* a aussi été importé dans PCIMe de QPIM, et élargi pour inclure une CompoundPolicyAction parallèle. Dans les deux cas, l'idée est de créer des "tronçons" de politique réutilisables qui puissent exister comme éléments désignés dans un ReusablePolicyContainer. Les classes "Compound" et leurs associations incorporent la sémantique de condition et d'action que PCIM définissait au niveau PolicyRule : DNF/CNF pour les conditions, et l'ordre pour les actions.

Les conditions et actions Compound sont définies comme fonctionnant avec toutes les conditions et actions composantes. En d'autres termes, alors que les composants peuvent être des instances, respectivement, de SimplePolicyCondition et de SimplePolicyAction (exposées immédiatement ci-dessous), ils n'ont pas besoin de l'être.

3.2.6 Variables et valeurs

La structure SimplePolicyCondition / PolicyVariable / PolicyValue a été importée dans PCIMe de QPIM. Une liste de variables de niveau PCIMe est définie, ainsi qu'une liste de valeurs de niveau PCIMe. D'autres variables et valeurs peuvent, si nécessaire, être définies dans des sous modèles de PCIMe. Par exemple, QPIM définit un ensemble de variables implicites correspondant aux champs dans les flux RSVP.

Une structure correspondante SimplePolicyAction / PolicyVariable / PolicyValue est aussi définie. Bien que la sémantique d'une SimplePolicyCondition soit "variable MATCH valeur" *(la variable correspond à une valeur)*, une SimplePolicyAction a la sémantique "SET variable TO value" *(régler la variable à une certaine valeur)*.

3.2.7 Filtrage de paquet au niveau domaine

Pour le filtrage de paquet spécifié au niveau domaine, un ensemble de PolicyVariables et de PolicyValues est défini, correspondant aux champs dans un en-tête de paquet IP plus les champs d'en-tête de trame de couche 2 les plus courants. On s'attend à ce que les conditions de politique de niveau domaine qui filtrent sur ces champs d'en-tête soient exprimés en termes de CompoundPolicyConditions construites à partir des SimplePolicyConditions qui utilisent ces variables et valeurs. Une PolicyVariable supplémentaire, PacketDirection, est aussi définie, pour indiquer si un paquet filtré voyage en entrée ou en sortie sur une interface.

3.2.8 Filtrage de paquet au niveau appareil

Pour le filtrage de paquet exprimé au niveau appareil, y compris les filtres de classificateur de paquet modélisés dans QDDIM, les variables et valeurs exposées au paragraphe 3.2.7 n'ont pas besoin d'être utilisées. Les classes de filtre dérivées de la hiérarchie de classe CIM FilterEntryBase sont disponibles pour être utilisées dans ces contextes. Ces dernières classes ont deux importantes différences avec les classes de niveau domaine :

- o Elles prennent en charge la spécification de filtres pour tous les champs dans un en-tête de protocole particulier dans une seule instance d'objet. Avec les classes de niveau domaine, des instances séparées sont nécessaires pour chaque champ d'en-tête.
- o Elles fournissent des représentations natives pour les valeurs de filtre, à l'opposé de la représentation de chaîne utilisée par les classes de niveau domaine.

Les classes de filtre de niveau appareil pour les en-têtes qui se rapportent à IP (IP, UDP, et TCP) et les en-têtes MAC 802 sont respectivement définis aux paragraphes 6.19 et 6.20.

4. Hiérarchies mises à jour de classe et de classe d'association

La figure suivante montre la hiérarchie d'héritage de classe pour PCIMe. Les changements par rapport à la hiérarchie PCIM sont notés entre parenthèses.

```
ManagedElement (abstraite)
    |
    +--Policy (abstraite)
    |   |
    |   +---PolicySet (abstraite -- nouveau - 5.3)
    |   |    |
    |   |    +---PolicyGroup (déplacé - 5.3)
    |   |    |
    |   |    +---PolicyRule (déplacé - 5.3)
    |   |
    |   +---PolicyCondition (abstraite)
    |   |    |
    |   |   +---PolicyTimePeriodCondition
    |   |    |
    |   |    +---VendorPolicyCondition
    |   |    |
    |   |    +---SimplePolicyCondition (nouveau - 5.8.1)
    |   |    |
    |   |    +---CompoundPolicyCondition (nouveau - 5.7.1)
    |   |         |
    |   |         +---CompoundFilterCondition (nouveau - 5.9)
    |   |
    |   +---PolicyAction (abstraite)
    |   |    |
    |   |    +---VendorPolicyAction
    |   |    |
    |   |    +---SimplePolicyAction (nouveau - 5.8.4)
    |   |    |
    |   |    +---CompoundPolicyAction (nouveau - 5.7.2)
    |   |
    |   +---PolicyVariable (abstract -- nouveau - 5.8.5)
    |   |    |
    |   |    +---PolicyExplicitVariable (nouveau - 5.8.6)
    |   |    |
    |   |    +---PolicyImplicitVariable (abstraite -- nouveau - 5.8.7)
    |   |         |
    |   |         +---(sous arbre de classes plus spécifiques-- nouveau - 6.12)
    |   |
    |   +---PolicyValue (abstraite -- nouvelle - 5.8.10)
    |    |
    |    +---(sous arbre de classes plus spécifiques-- nouveau - 6.14)
    |
```

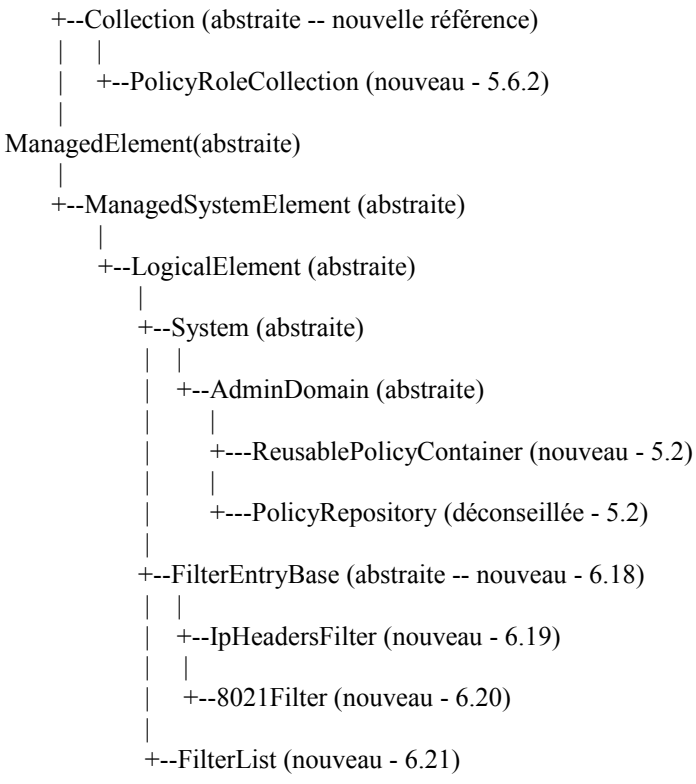
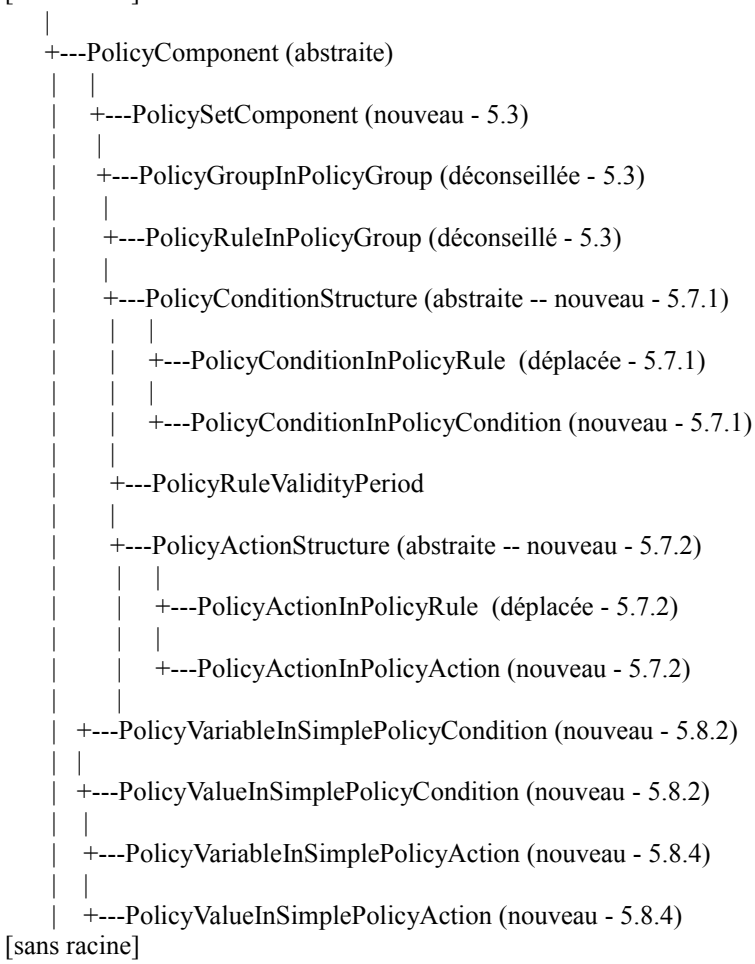


Figure 1 : Hiérarchie d'héritage de classe pour PCIMe

La figure suivante montre la hiérarchie de classe d'association pour PCIMe. Comme précédemment, les changements par rapport à PCIM sont notés entre parenthèses.



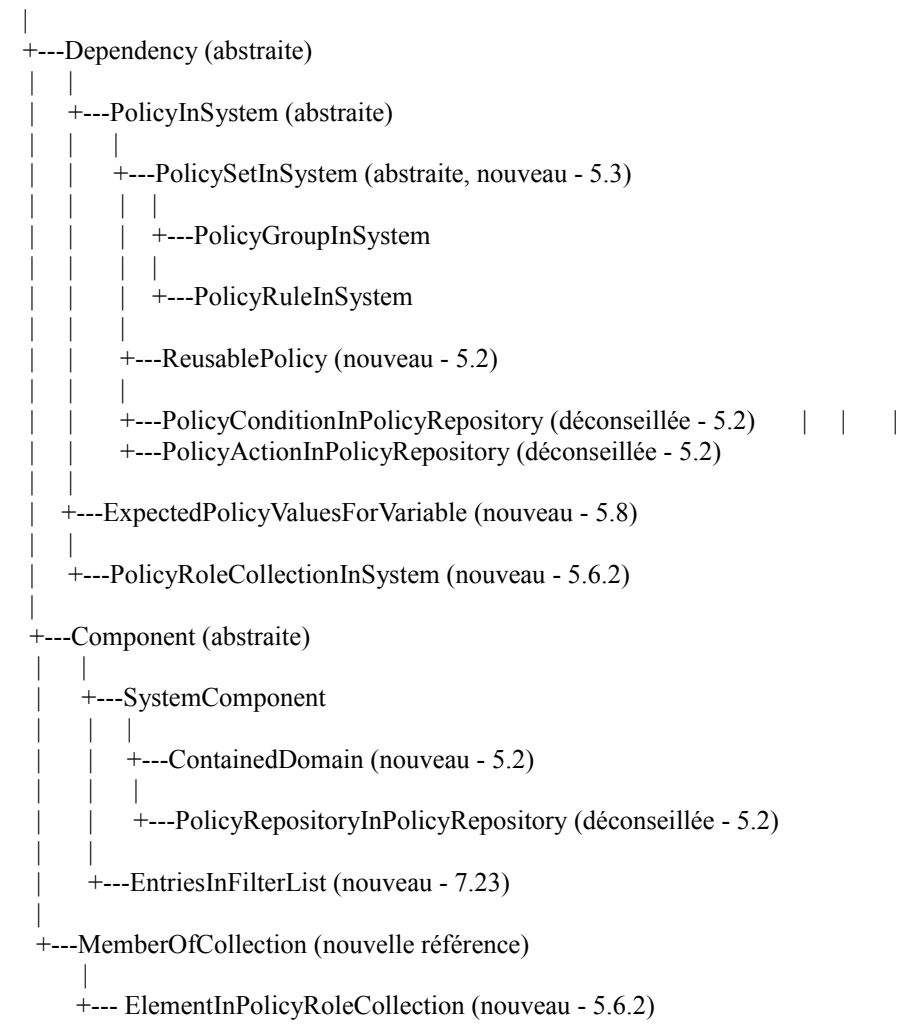


Figure 2 : Hiérarchie d'héritage de classe d'association pour PCIMe

En plus de ces changements qui se manifestent au niveau de la classe et de la classe d'association, il y a d'autres changements par rapport à PCIM qui impliquent des propriétés de classe individuelle. Dans certains cas, de nouvelles propriétés sont introduites dans des classes existantes, et dans d'autres cas, des propriétés existantes sont déconseillées (sans déconseiller les classes qui les contiennent).

5. Zones d'extension à PCIM

Les paragraphes qui suivent décrivent chacune des zones pour lesquelles sont définies des extensions à PCIM.

5.1 Portée de politique

Les portées de politique peuvent être vues dans deux dimensions :
1) le niveau d'abstraction de la spécification de politique et
2) l'applicabilité des politiques à un ensemble de ressources gérées.

5.1.1 Niveaux d'abstraction : politiques de niveau domaine et appareil

Les politiques varient en niveau d'abstraction, depuis l'expression au niveau commercial des accords de niveau de service (SLA, *service level agreement*) à la spécification d'un ensemble de règles qui s'appliquent aux appareils dans un réseau. Ces dernières politiques peuvent elles-mêmes être classées en au moins deux groupes : les politiques consommées par un point de décision de politique (PDP, *Policy Decision Point*) qui spécifient les règles pour un domaine administratif et fonctionnel, et les politiques consommées par un point d'application de politique (PEP, *Policy Enforcement Point*) qui spécifient les règles spécifiques d'un appareil pour un domaine fonctionnel. Les règles de niveau supérieur consommées par un PDP, appelées

politiques de niveau domaine, peuvent avoir des variables de lien non spécifiées, ou spécifiées par une classification, tandis que les règles de niveau appareil vont vraisemblablement avoir moins de liens non résolus.

Il y a une relation entre ces niveaux de spécification de politique qui sort du domaine d'application de la présente spécification, mais qui est nécessaire dans le développement et le déploiement d'un système de configuration fondé sur la politique. Une transformation de politique de niveau SLA en politique de niveau domaine peut être vue comme analogue à une construction visuelle qui prend une entrée humaine et développe une spécification de règle de programmation. La relation entre la politique de niveau domaine et la politique de niveau appareil peut être vue comme analogue à celle d'un compilateur et d'un éditeur de liens qui traduit les règles en instructions spécifiques qui peuvent être exécutées sur un type spécifique de plateforme.

PCIM et PCIMe peuvent être utilisés pour spécifier des règles à tous ces niveaux d'abstraction. Cependant, aux différents niveaux d'abstraction, des mécanismes différents peuvent être plus ou moins appropriés.

5.1.2 Portées administratives et fonctionnelles

Les portées administratives pour la politique sont représentées dans PCIM et dans ces extensions à PCIM comme des instances de sous classe de système. Normalement, une politique de niveau domaine va recevoir sa portée d'une instance de AdminDomain (ou par une hiérarchie d'instances de AdminDomain) tandis qu'une politique de niveau appareil peut recevoir sa portée d'une instance de système qui représente le PEP (par exemple, une instance de ComputerSystem, voir CIM [CIM]). En plus de collecter les politiques dans un domaine administratif, ces classes de système peuvent aussi agréger les ressources auxquelles les politiques s'appliquent.

Les portées fonctionnelles (parfois appelées domaines fonctionnels) sont généralement définies par les sous modèles déduits de PCIM et PCIMe, et correspondent au service ou services auxquels les politiques s'appliquent. Ainsi, par exemple, la qualité de service peut être vue comme une portée fonctionnelle, ou Diffserv et Intserv peuvent chacun être vus comme des portées fonctionnelles. Ces décisions de portées sont représentées par la structure des sous modèles dérivés de PCIM et PCIMe, et peuvent être reflétés dans le nombre et les types de clients de politique de PEP, les services, et l'interaction entre les politiques. Les politiques dans différentes portées fonctionnelles sont organisées en ensembles disjoints de règles de politique. Des domaines fonctionnels différents peuvent partager certains rôles, certaines conditions, et même certaines actions. Les règles provenant de domaines fonctionnels différents peuvent même être mises en application à la même ressource gérée, mais pour les besoins de l'évaluation de politique, elles sont séparées. Voir plus d'informations au paragraphe 5.5.3.

Les portées fonctionnelles PEUVENT être reflétées dans les portées administratives. C'est-à-dire que les déploiements de politique peuvent avoir des portées administratives différentes pour des portées fonctionnelles différentes, mais il n'y a pas d'exigence qu'il en soit ainsi.

5.2 Éléments de politique réutilisables

Dans PCIM, une distinction a été faite entre les PolicyConditions et PolicyActions réutilisables et celles spécifiques d'une règle. La classe PolicyRepository a aussi été définie, pour servir de conteneur pour ces éléments réutilisables. Le nom de "PolicyRepository" s'est révélé être un chois malheureux pour la classe qui sert de conteneur pour les éléments de politique réutilisables. Ce terme est déjà utilisé dans des documents comme le cadre de politique, pour noter la localisation à partir de laquelle le PDP restitue toutes les spécifications de politique, et dans laquelle l'outil de gestion de politique (PMT, *Policy Management Tool*) place toutes les spécifications de politique. Par conséquent, la classe PolicyRepository est déconseillée, en faveur d'une nouvelle classe ReusablePolicyContainer.

Lorsque une classe est déconseillée, toute association qui s'y réfère doit aussi être déconseillée. De sorte que des solutions de remplacement sont nécessaires pour les deux associations PolicyConditionInPolicyRepository *(condition de politique dans un répertoire de politique)* et PolicyActionInPolicyRepository *(action de politique dans un répertoire de politique)*, ainsi que pour l'agrégation PolicyRepositoryInPolicyRepository *(répertoire de politique dans un répertoire de politique)*. En plus de renommer la classe PolicyRepository en ReusablePolicyContainer, PCIMe a cependant aussi élargi les types d'éléments de politique qui peuvent être réutilisables. Par conséquent, plutôt que de remplacer une à une les deux associations, une seule association ReusablePolicy de niveau supérieur est définie. Cette nouvelle association permet de placer tout élément de politique (c'est-à-dire, une instance de toute sous classe de la classe abstraite Policy) dans un ReusablePolicyContainer.

En résumé, les changements suivants aux Sections 6 et 7 sont le résultat de cet élément :
- La classe ReusablePolicyContainer est définie.
- La classe PolicyRepository de PCIM est déconseillée.
- L'association ReusablePolicy est définie.
- L'association PolicyConditionInPolicyRepository de PCIM est déconseillée.
- L'association PolicyActionInPolicyRepository de PCIM est déconseillée.

- o L'agrégation ContainedDomain est définie.
- o L'agrégation PolicyRepositoryInPolicyRepository de PCIM est déconseillée.

5.3 Ensembles de politique

Une "politique" peut être vue comme un ensemble cohérent de règles pour administrer, gérer, et contrôler l'accès aux ressources du réseau ("Terminologie de politique", [RFC3198]). La structuration de ces ensembles cohérents de règles en sous ensembles est améliorée dans le présent document. Au paragraphe 5.4, on expose les nouvelles options pour l'hébergement des règles de politique.

Une nouvelle classe abstraite, PolicySet *(ensemble de politiques)*, est introduite pour fournir une forme abstraite pour un ensemble de règles. Elle est déduite de Policy, et est insérée dans la hiérarchie d'héritage au dessus de PolicyGroup et PolicyRule. Cela reflète la souplesse structurelle et la capacité sémantique supplémentaire des deux sous classes.

Deux propriétés sont définies dans PolicySet : PolicyDecisionStrategy et PolicyRoles. La propriété PolicyDecisionStrategy *(stratégie de décision de politique)* est incluse dans PolicySet pour définir les relations d'évaluation entre les règles dans l'ensemble de politique. Voir des informations complémentaires au paragraphe 5.5. La propriété PolicyRoles *(rôles de politique)* est incluse dans PolicySet pour caractériser les ressources auxquelles s'applique le PolicySet. Voir les détails au paragraphe 5.6.

Avec la définition de la classe PolicySet, une nouvelle classe d'agrégation concrète est définie qui sera aussi exposée dans les paragraphes qui suivent. PolicySetComponent *(composant d'ensemble de politique)* est défini comme une sous classe de PolicyComponent ; elle fournit une relation de contenance pour un PolicySet dans un PolicySet. PolicySetComponent remplace les deux agrégations PCIM de PolicyGroupInPolicyGroup et PolicyRuleInPolicyGroup, de sorte que ces deux agrégations sont déconseillées.

La relation d'un PolicySet avec un AdminDomain ou autre système de portée administrative (par exemple, un ComputerSystem) est représentée par l'association abstraite PolicySetInSystem *(ensemble de politiques dans le système)*. Cette nouvelle association est dérivée de PolicyInSystem, et les associations PolicyGroupInSystem et PolicyRuleInSystem sont maintenant déduites de PolicySetInSystem au lieu de découler directement de PolicyInSystem. La propriété PolicySetInSystem.Priority est exposée au paragraphe 5.5.3.

5.4 Règles de politique incorporées

Comme on l'a exposé précédemment, la politique est décrite par un ensemble de règles de politique qui peuvent être groupées en sous ensembles. Dans ce paragraphe, on introduit la notion de règles incorporées, ou la capacité à définir des règles au sein de règles. Les règles incorporées sont aussi appelées des sous règles, et on utilise les deux termes de façon interchangeable dans le présent document. L'agrégation PolicySetComponent *(composant d'ensemble de politiques)* est utilisée pour représenter l'incorporation d'une règle de politique dans une autre règle de politique.

5.4.1 Règles d'usage pour les règles incorporées

Les relations entre les règles et les sous règles sont définies comme suit :

- o La clause de condition de la règle parente est une condition pour l'évaluation de toutes les règles incorporées ; c'est-à-dire que les conditions du parent sont ajoutées logiquement (ET) aux conditions des sous règles. Si la clause de condition de la règle parente s'évalue comme FAUX, les sous règles PEUVENT être sautées car elles s'évaluent aussi à FAUX.
- o Si la condition de la règle parente s'évalue à VRAI, l'ensemble de sous règles DEVRA être évalué conformément à la stratégie de décision et de priorités exposée au paragraphe 5.5.
- o Si la condition de la règle parente s'évalue à VRAI, l'ensemble d'actions de la règle parente est exécuté AVANT l'exécution des actions des sous règles. Les actions de la règle parente ne doivent pas être confondues avec les actions par défaut. L'action par défaut est celle qui ne doit être exécutée que si aucune des sous règles plus spécifiques n'est exécutée. Si une action par défaut a besoin d'être spécifiée, elle doit être définie comme une action qui fait partie d'une sous règle de fourre tout associée à la règle parente. L'association qui relie la ou les actions par défaut dans cette sous règle spéciale devrait avoir la plus basse priorité par rapport à toutes les autres associations de sous règles :

  si condition parente alors action de la règle parente
  si condition A alors action A
  si condition B alors action B
  si VRAI, alors action par défaut

  De telles actions par défaut fonctionnent par défaut lorsque des stratégies de décision FirstMatching *(la première qui correspond)* sont utilisées (voir au paragraphe 5.5). Si AllMatching *(toutes correspondances)* s'applique, l'action "par

défaut" est toujours effectuée.

- Les règles de politique ont un contexte dans lequel elles sont exécutées. Le moteur de règles évalue et applique les règles de politique dans le contexte des ressources gérées qui sont identifiées par les rôles de politique (ou par une association explicite). Des sous modèles PEUVENT ajouter un contexte supplémentaire aux règles de politique sur la base de la structure de règle ; tout contexte supplémentaire est défini par la sémantique des classes d'action du sous modèle.

5.4.2 Motivation

L'incorporation de règles améliore la lisibilité, l'expressivité et la réutilisabilité de la politique. La capacité à incorporer les règles de politique et à former des sous règles est importante pou la gestion et l'adaptabilité, car elle permet de construire des règles de politique complexes à partir de plusieurs règles de politique plus simples.

Ces améliorations facilitent la tâche des outils de gestion de politique, en permettant d'exprimer les règles de politique d'une façon plus proche de celle dont pensent les êtres humains.

Bien que l'incorporation de règles puisse être utilisée pour suggérer des optimisations de la façon dont les règles de politique sont évaluées, comme exposé au paragraphe 5.5.2 "Effets collatéraux", l'incorporation ne spécifie ni n'exige aucun ordre particulier de l'évaluation des conditions. L'optimisation de l'évaluation de règles peut être faite dans le PDP ou dans le PEP par un code dédié. Ceci est similaire à la relation entre un langage de programmation de haut niveau comme le langage C et le code machine. Un optimiseur peut créer un code machine plus efficace que toute optimisation faite par le programmeur au sein du code source. Néanmoins, si le PEP ou le PDP ne fait pas d'optimisation, l'administrateur qui écrit la politique peut être capable d'influencer l'évaluation des règles de politique pour une exécution utilisant l'incorporation de règles.

Les règles incorporées ne sont pas conçues pour l'optimisation de la restitution de répertoires de politique. On suppose que toutes les règles et groupes qui sont assignés à un rôle sont restitués par le PDP ou le PEP à partir du répertoire de politique et mis en application. L'optimisation du nombre de règles restituées devrait être faite par une sélection plus habile des rôles.

5.5 Priorités et stratégies de décision

Une "stratégie de décision" est utilisée pour spécifier la méthode d'évaluation pour les politiques dans un PolicySet. Deux stratégies de décision sont définies : "FirstMatching" et "AllMatching". La stratégie FirstMatching est utilisée pour causer l'évaluation des règles dans un ensemble tel que les seules actions mises en application sur un certain examen du PolicySet soient celles pour la première règle (c'est-à-dire, la règle qui a la plus forte priorité) qui a ses conditions évaluées à VRAI. La stratégie AllMatching est utilisée pour causer l'évaluation de toutes les règles dans un ensemble ; pour toutes les règles dont les conditions s'évaluent à VRAI, les actions sont mises en application. Les mises en œuvre DOIVENT prendre en charge la stratégie de décision FirstMatching ; les mises en œuvre PEUVENT prendre en charge la stratégie de décision AllMatching.

Comme exposé précédemment, les sous classes de PolicySet sont PolicyGroup et PolicyRule : les deux sous classes peuvent contenir des PolicySets de l'une ou l'autre sous classe. Les boucles, y compris le cas anormal d'un PolicySet qui se contient lui-même, ne sont pas permises lorsque les PolicySets contiennent d'autres PolicySets. La relation de contenance est spécifiée en utilisant l'agrégation PolicySetComponent.

La priorité relative au sein d'un PolicySet est établie par la propriété Priority de l'agrégation PolicySetComponent des instances contenues de PolicyGroup et PolicyRule. L'utilisation de la propriété PolicyRule.Priority de PCIM est déconseillée en faveur de cette nouvelle propriété. La séparation de la propriété de priorité de la règle présente deux avantages. D'abord, elle généralise le concept de priorité, de sorte qu'elle peut être utilisée pour les groupes et les règles. Ensuite, elle place la priorité sur la relation entre l'ensemble de politique parent et le groupe ou règle de politique subordonné. L'assignation d'une valeur de priorité devient ensuite beaucoup plus facile, en ce que la valeur n'est utilisée qu'en relation avec d'autres priorités dans le même ensemble.

Ensemble, le PolicySet.PolicyDecisionStrategy et le PolicySetComponent.Priority déterminent le traitement pour les règles contenues dans un PolicySet. Comme précédemment, la plus forte valeur de priorité représente la plus forte priorité. À la différence de la définition antérieure, PolicySetComponent.Priority DOIT avoir une valeur unique lorsque elle est comparée à d'autres définies pour le même PolicySet agrégateur. Donc, l'évaluation des règles au sein d'un ensemble est spécifiée de façon déterministe.

Pour une stratégie de décision FirstMatching, la première règle (c'est-à-dire, celle qui a la plus forte priorité) dans l'ensemble qui s'évalue à VRAI, est la seule règle dont les actions sont mises en application pour une passe d'évaluation particulière à travers le PolicySet.

Pour une stratégie de décision AllMatching, toutes les règles correspondantes sont mises en application. La priorité relative des

règles est utilisée pour déterminer l'ordre dans lequel les actions doivent être exécutées par le point d'application : les actions des règles de plus forte priorité sont exécutées les premières. Comme les actions des règles de plus forte priorité sont exécutées les premières, les règles de priorité inférieure qui correspondent aussi peuvent avoir le "dernier mot", et donc produire un résultat contre intuitif. Ainsi, par exemple, si deux règles s'évaluent toutes deux à VRAI, et si la règle de plus forte priorité établit le DSCP à 3 et si la règle de priorité inférieure règle le DSCP à 4, l'action de la règle de priorité inférieure sera exécutée en dernier et donc va "gagner" dans cet exemple, en réglant le DSCP à 4. Donc, les conflits entre règles sont résolus par cet ordre d'exécution.

Une mise en œuvre du moteur de règles n'a pas besoin de fournir le séquençage des actions mais les actions DOIVENT être séquencées par le PEP ou PDP en son nom. Ainsi, par exemple, le moteur de règles peut fournir une liste ordonnée des actions à exécuter par le PEP et toute sérialisation nécessaire est alors fournie par le service configuré par le moteur de règles. Voir au paragraphe 5.5.2 une discussion des effet collatéraux.

5.5.1 Structuration des stratégies de décision

Comme exposé aux paragraphes 5.3 et 5.4, les instances de PolicySet peuvent être incorporées de façon arbitraire. Pour une stratégie de décision FirstMatching sur un PolicySet, tout PolicySet contenu qui correspond satisfait aux critères de terminaison pour la stratégie FirstMatching. Un PolicySet est considéré correspondre si il est une PolicyRule et si ses conditions s'évaluent à VRAI, ou si le PolicySet est un PolicyGroup et si au moins un de ses PolicyGroup ou PolicyRule contenus correspond. La priorité associée aux PolicySets contenus détermine alors quand terminer l'évaluation de règle dans l'ensemble structuré de règles.

Dans l'exemple montré à la Figure 3, les priorités relatives pour les règles incorporées, de forte à faible, sont 1A, 1B1, 1X2, 1B3, 1C, 1C1, 1X2 et 1C3. (Noter que la PolicyRule 1X2 est incluse dans les PolicyGroup 1B et PolicyRule 1C, mais avec une priorité différente.) Bien sûr, la règle à appliquer dépend aussi de quelle règle correspond, s'il en est.

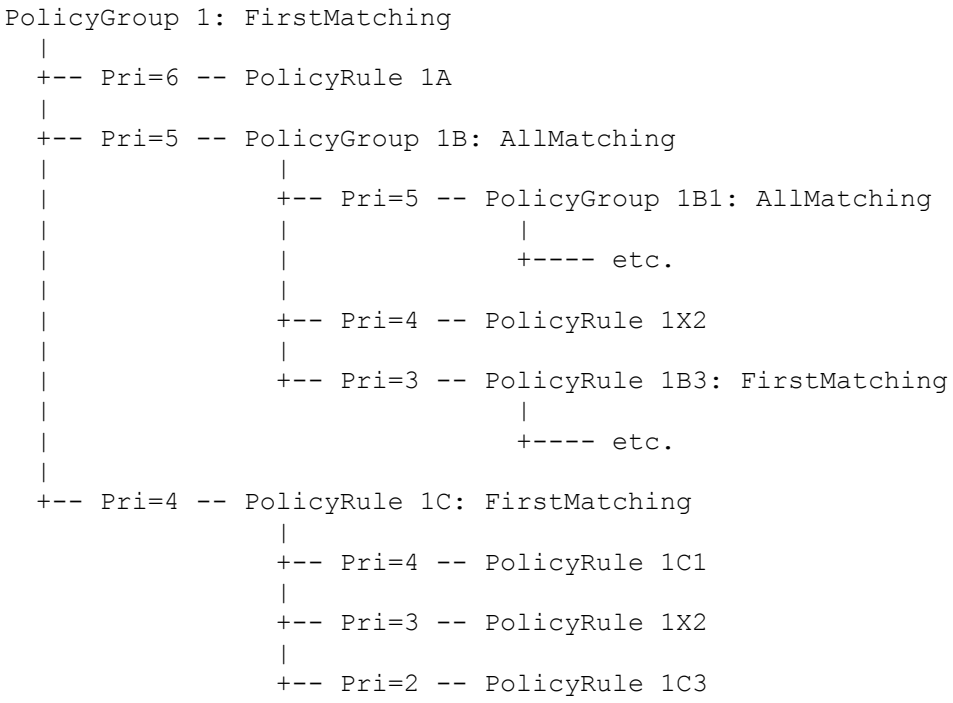


Figure 3 : Ensembles de politiques incorporées avec des stratégies de décision différentes

- Comme PolicyGroup 1 a une stratégie de décision FirstMatching, si les conditions de PolicyRule 1A correspondent, ses actions sont mises en application et l'évaluation s'arrête.
- Si elles ne correspondent pas, PolicyGroup 1B est évalué en utilisant une stratégie AllMatching. Comme PolicyGroup 1B1 a aussi une stratégie AllMatching, toutes les règles et groupes de règles contenus dans PolicyGroup 1B1 sont évalués et mis en application comme approprié. PolicyRule 1X2 et PolicyRule 1B3 sont aussi évaluées et mises en application comme approprié. Si une des sous règles dans les sous arborescences de PolicyGroup 1B s'évalue à VRAI, alors PolicyRule 1C n'est pas évaluée parce que la stratégie FirstMatching de PolicyGroup 1 a été satisfaite.
- Si ni PolicyRule 1A ni PolicyGroup 1B ne donnent de correspondance, alors PolicyRule 1C est évaluée. Comme c'est une première correspondance, les règles 1C1, 1X2, et 1C3 sont évaluées jusqu'à la première correspondance, s'il y en a une.

5.5.2 Effets collatéraux

Bien que l'évaluation des conditions soit parfois présentée comme un ensemble ordonné d'opérations, le moteur de règles n'a pas besoin d'être mis en œuvre comme un interpréteur de langage procédural. Un effet collatéral de l'évaluation de condition ou de l'exécution des actions NE DOIT PAS affecter le résultat de l'évaluation des autres conditions évaluées par le moteur de règles dans le même passage d'évaluation. C'est-à-dire que la mise en œuvre d'un moteur de règles PEUT évaluer toutes les conditions dans n'importe quel ordre avant d'appliquer la priorité et de déterminer quelles actions sont à exécuter.

Donc, sans considération de la façon dont un moteur de règles est mis en œuvre, il NE DOIT PAS inclure d'effets collatéraux d'évaluation des conditions dans l'évaluation de conditions pour l'une ou l'autre des stratégies de décision. Aussi bien pour la stratégie de décision AllMatching que pour l'incorporation de règles au sein de règles (soit directement, soit indirectement) lorsque les actions de plus d'une règle peuvent être mises en application, tout effet collatéral de la mise en application des actions DOIT ne pas être inclus dans l'évaluation de condition sur le même passage d'évaluation.

5.5.3 Plusieurs arborescences de PolicySet pour une ressource

Comme montré dans l'exemple de la Figure 3, les arborescences de PolicySet sont définies par les instances de sous classes de PolicySet et les instances d'agrégation de PolicySetComponent entre elles. Chaque arborescence de PolicySet a un ensemble défini de stratégies de décision et de priorités d'évaluation. Au paragraphe 5.6, on expose des améliorations de l'utilisation des PolicyRoles qui font que le PolicySet.PolicyRoles parent est à appliquer à toutes les instances de PolicySet contenues. Cependant, une certaine ressource peut quand même avoir plusieurs arborescences de PolicySet disjointes sans considération de la façon dont elles sont collectées. Ces instances de PolicySet de niveau supérieur sont appelées "sans racine" par rapport à la ressource considérée.

Ainsi, une instance de PolicySet est définie comme à racine ou sans racine dans le contexte d'un élément géré particulier ; la relation à l'élément géré est usuellement établie par les rôles de politique de l'instance de PolicySet et de l'élément géré (voir le paragraphe 5.6 "Rôles de politique"). Une instance de PolicySet est sans racine dans ce contexte si et seulement si il n'y a pas d'association PolicySetComponent avec un PolicySet parent, c'est-à-dire aussi en rapport avec le même élément géré. Ces agrégations de PolicySetComponent traversent toute l'arborescence sans considération de la façon dont une instance de PolicySet en est venue à être en relation avec l'élément géré. La Figure 4 montre un exemple où l'instance A a le rôle A, l'instance B a le rôle B et ainsi de suite. Dans cet exemple, dans le contexte de l'interface X, les instances B et C sont sans racine et les instances D, E, et F sont toutes à racine. Dans le contexte de l'interface Y, l'instance A est sans racine et les instances B, C, D, E et F sont toutes à racine.

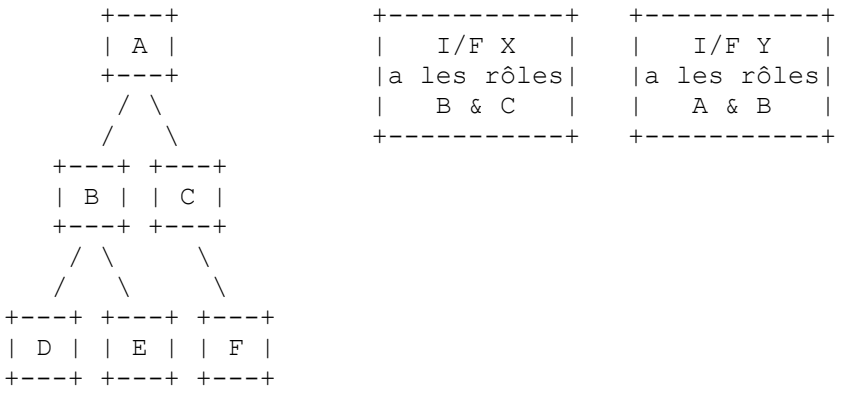


Figure 4 : Instances de PolicySet sans racine

Pour les cas où il y a plusieurs instances de PolicySet sans racine qui s'appliquent à la même ressource gérée (c'est-à-dire, qui n'est pas dans une arborescence commune de PolicySetComponent) la stratégie de décision parmi ces instances disjointes de PolicySet est la stratégie FirstMatching. La priorité utilisée avec cette stratégie FirstMatching est définie dans l'association PolicySetInSystem. Les instances de sous classe de PolicySetInSystem sont présentes pour toutes les instances de PolicySet (c'est une association exigée) mais la priorité n'est utilisée que par défaut pour les instances de PolicySet sans racine dans un certain contexte de ManagedElement.

La stratégie FirstMatching est utilisée parmi toutes les instances de PolicySet sans racine qui s'appliquent à une certaine ressource pour un certain domaine fonctionnel. Ainsi, par exemple, les instances de PolicySet qui sont utilisées pour la politique de QS et les instances qui sont utilisées pour la politique de IKE, bien qu'elles soient disjointes, ne sont pas jointes dans une stratégie de décision FirstMatching. Elles sont plutôt évaluées indépendamment l'une de l'autre.

5.5.4 Décisions déterministes

Comme exposé précédemment, les valeurs de PolicySetComponent.Priority DOIVENT être uniques au sein d'un PolicySet contenant, et les valeurs de PolicySetInSystem.Priority DOIVENT être uniques pour un système associé. Chaque PolicySet a alors un comportement déterministe fondé sur la stratégie de décision et une priorité définie univoque.

Il y a certainement des cas où les règles n'ont pas besoin d'avoir une valeur de priorité unique (c'est-à-dire, où la priorité d'évaluation et d'exécution n'est pas importante). Cependant, on pense que la souplesse offerte par cette capacité n'est pas suffisamment bénéfique pour justifier de possibles variations du comportement des mises en œuvre et la confusion résultante qui pourrait s'ensuivre.

5.6 Rôles de politique

Un rôle de politique est défini dans la [RFC3198] comme "une caractéristique administrativement spécifiée d'un élément géré (par exemple, une interface). C'est un sélecteur des règles de politique et des classes d'approvisionnement (PRC) pour déterminer l'applicabilité de la règle/PRC pour un élément géré particulier".

Dans PCIMe, un rôle de politique est défini comme une propriété d'un PolicySet, qui est hérité par les règles de politique et les groupes de politique. Dans le présent document, on ajoute aussi le PolicyRole comme le nom identifiant une collection de ressources (PolicyRoleCollection), où chaque élément de la collection a les caractéristiques de rôle spécifiées.

5.6.1 Comparaison des rôles dans PCIM et des rôles dans snmpconf

Dans la gestion de configuration de la MIB de gestion fondée sur la politique du groupe de travail SNMP (snmpconf) [RFC4011], les règles de politique sont de la forme :

si <policyFilter> alors <policyAction>

où <policyFilter> est un ensemble de conditions qui sont utilisées pour déterminer si la politique s'applique ou non à une instance d'objet. Le filtre de politique peut effectuer des opérations de comparaison sur les variables SNMP déjà définies dans les MIB (par exemple, "ifType == ethernet").

La MIB de gestion de politique définie dans la [RFC4011] définit un tableau des rôles qui permet d'associer les rôles aux éléments, où les rôles ont la même sémantique que dans PCIM. Donc, comme le policyFilter dans une politique permet de définir des conditions sur la base de la comparaison des valeurs des variables SNMP, on peut filtrer les éléments sur la base de leurs rôles tels que définis dans le groupe de rôles.

Cette approche diffère de celle adoptée dans PCIM par les aspects suivants. D'abord, dans PCIM, un ensemble de rôles est associé à une règle de politique comme les valeurs de la propriété PolicyRoles d'une règle de politique. La sémantique de rôle est alors supposée être mise en œuvre par le PDP (c'est-à-dire, les politiques sont appliquées aux éléments avec les rôles appropriés). Dans la [RFC4011], cependant, aucun traitement spécial n'est exigé pour réaliser la sémantique des rôles ; les rôles sont traités juste comme toute autre variable SNMP et les comparaisons des valeurs de rôle peuvent être incluses dans le filtre de politique d'une règle de politique. Ensuite, dans PCIM, il n'y a pas de façon formellement définie d'associer un rôle à une instance d'objet, tandis que dans la [RFC4011] ceci est fait via l'utilisation des tableaux de rôles (pmRoleESTable et pmRoleSETable). Les tableaux de rôles associent les valeurs de rôle aux éléments.

5.6.2 Ajout de PolicyRoleCollection à PCIMe

Pour remédier à cette dernière lacune de PCIM (l'absence de moyen d'associer un rôle à une instance d'objet) PCIMe a une nouvelle classe PolicyRoleCollection dérivée de la classe Collection de CIM. Les ressources qui partagent un rôle commun sont agrégées par une instance de PolicyRoleCollection, via l'agrégation ElementInPolicyRoleCollection. Le rôle est spécifié dans la propriété PolicyRole de l'instance agrégeante PolicyRoleCollection.

Une PolicyRoleCollection existe toujours dans le contexte d'un système. Comme il était fait dans PCIM pour les PolicyRules et les PolicyGroups, une association, PolicyRoleCollectionInSystem, capture cette relation. On se souvient que dans CIM, System est une classe de base pour décrire les appareils du réseau et les domaines administratifs.

L'association entre une PolicyRoleCollection et un système devrait être cohérente avec les associations qui délimitent les règles/groupes de politique qui sont appliqués aux ressources dans cette collection. Précisément, une PolicyRoleCollection devrait être associée au même système que les PolicyRules et/ou PolicyGroups applicables, ou à un système plus élevé dans l'arborescence formée par l'association de SystemComponent. Lorsque un PEP appartient à plusieurs systèmes (c'est-à-dire,

AdminDomains), et que la délimitation par un seul domaine est impraticable, une alternative existe : soit limiter arbitrairement son appartenance de domaine à un System/AdminDomain, soit définir un AdminDomain plus global qui inclut simplement les autres, et/ou qui couvre le secteur commercial ou l'entreprise.

Par exemple, supposons qu'il y ait vingt circuits de trafic dans un réseau, et qu'un administrateur veuille allouer trois d'entre eux pour fournir un service "or". L'administrateur a aussi défini plusieurs règles de politique qui spécifient comment le service "or" est fourni. Pour ces règles, la propriété PolicyRoles (héritée de PolicySet) est réglée à "Service or".

Afin d'associer trois circuits de trafic au service "or", une instance de la classe PolicyRoleCollection est créée et sa propriété PolicyRole est aussi réglée à "Service or". À la suite de cela, l'administrateur associe trois circuits de trafic à la nouvelle instance de PolicyRoleCollection via l'agrégation ElementInPolicyRoleCollection. Cela permet à un PDP de déterminer que les règles de politique "Service or" s'appliquent aux trois circuits de trafic agrégés.

Noter que les rôles sont utilisés pour optimiser la restitution de politique. Il n'est pas obligatoire de mettre en œuvre des rôles ou, si ils ont été mis en œuvre, de grouper les éléments dans une PolicyRoleCollection. Cependant, si des rôles sont utilisés, l'approche de la collection devrait être mise en œuvre, ou des éléments devraient être capables de rapporter leur rôle "préprogrammé" (comme cela est fait dans COPS).

5.6.3 Rôles pour PolicyGroups

Dans PCIM, les rôles sont seulement associés à des règles de politique. Cependant, il peut être souhaitable d'associer des rôles à des groupes de règles de politique. Par exemple, un administrateur de réseau peut vouloir définir un groupe de règles qui ne s'appliquent qu'aux interfaces Ethernet. Un groupe de politique peut être défini avec une combinaison de rôle ="Ethernet", et toutes les règles de politique pertinentes peuvent être placées dans ce groupe de politique. (Noter que dans PCIMe, les rôles sont rendus disponibles aux PolicyGroup aussi bien qu'aux PolicyRule en déplaçant la propriété PolicyRoles de PCIM dans la nouvelle classe abstraite PolicySet. La propriété est alors héritée par les deux PolicyGroup et PolicyRule.) Toute règle de politique dans ce groupe de politique hérite alors implicitement de cette combinaison de rôles du groupe de politique contenant. Un héritage implicite similaire s'applique aux groupes de politique incorporés.

Il n'y a pas de copie explicite des rôles de l'identité contenante à l'entité contenue. Évidemment, cet héritage implicite de rôles conduit à la possibilité de définir des rôles incohérents (comme l'explique l'exemple ci-dessous) ; le traitement de telles incohérences sort du domaine d'application de PCIMe.

Par exemple, supposons qu'il y a un groupe de politique PG1 qui contient trois règles de politique, PR1, PR2, et PR3. Supposons que PG1 a les rôles "Ethernet" et "Fast". Supposons aussi que les règles de politique contenues ont leurs rôles montrés ci-dessous :

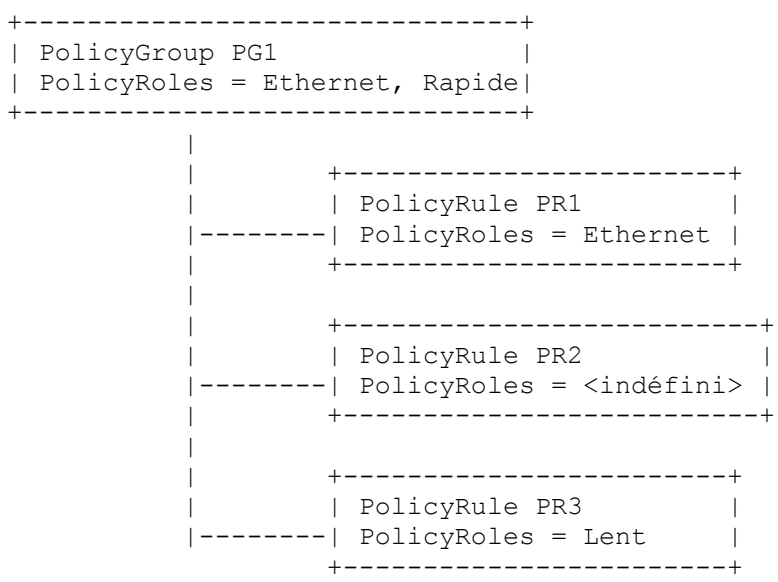


Figure 5 : héritage des rôles

Dans cet exemple, la valeur de la propriété PolicyRoles pour PR1 est cohérente avec la valeur dans PG1, et en fait, n'a pas besoin d'être redéfinie. La valeur de PolicyRoles pour PR2 est indéfinie. Son rôle est implicitement hérité de PG1. Enfin, la valeur de PolicyRoles pour PR3 est "Lent". Cela paraît entrer en conflit avec le rôle "Rapide," défini dans PG1. Cependant, il n'est pas clair que ces rôles soient réellement en conflit. Dans un scénario, l'administrateur de politique peut avoir seulement

voulu les règles "Rapide"- "Ethernet" dans le groupe de politiques. Dans un autre scénario, l'administrateur peut vouloir indiquer que PR3 s'applique à toutes les interfaces "Ethernet" sans considérer si elles sont "Rapides" ou "Lentes". C'est seulement dans le premier scénario (seulement les règles "Rapide"- "Ethernet" dans le groupe de politique) qu'il y a un conflit de rôles.

Noter qu'il est possible d'outrepasser les rôles hérités implicitement via des conditions appropriées sur une PolicyRule. Par exemple, supposons que ci-dessus PR3 ait défini les conditions suivantes :

(interface n'est pas "Rapide") et (interface est "Lent")

Il en résulte une sémantique non ambiguë pour PR3.

5.7 Conditions de politique composée et actions de politique composée

Les conditions de politique composée et les actions de politique composée sont introduites pour fournir des "tronçons" de politique réutilisables supplémentaires.

5.7.1 Conditions de politique composée

Une CompoundPolicyCondition est une PolicyCondition qui représente une combinaison booléenne de conditions plus simples. Les conditions qui sont combinées peuvent être des SimplePolicyConditions (exposées au paragraphe 6.4) mais l'utilité de conditions de combinaisons de politique n'est pas nécessairement limitée au cas où les conditions des composants sont simples.

Les extensions à PCIM pour introduire des conditions de politique composées sont relativement directes. Comme l'objet de l'extension est d'appliquer la logique DNF / CNF à partir de l'agrégation PolicyConditionInPolicyRule de PCIM à une condition composée qui agrége de plus simples conditions, les changements suivants sont requis :

- Créer une nouvelle agrégation PolicyConditionInPolicyCondition, avec les mêmes propriétés GroupNumber et ConditionNegated que PolicyConditionInPolicyRule. La façon la plus nette de faire cela est de déplacer les propriétés sur une nouvelle super classe d'agrégation abstraite PolicyConditionStructure, à partir de laquelle l'agrégation PolicyConditionInPolicyRule existante et une nouvelle agrégation PolicyConditionInPolicyCondition sont déduites. Pour l'instant il n'est pas nécessaire de re-documenter les propriétés elles-mêmes, car elles sont déjà documentées dans PCIM au titre de la définition de l'agrégation PolicyConditionInPolicyRule.

- Il est aussi nécessaire de définir une sous classe concrète CompoundPolicyCondition de PolicyCondition, pour introduire la propriété ConditionListType. Cette propriété a la même fonction, et fonctionne exactement de la même façon, que la propriété correspondante actuellement définie dans PCIM pour la classe PolicyRule.

Les définitions de classe et de propriété pour représenter les conditions de politique composées sont à la Section 6.

5.7.2 Actions de politique composée

Une action composée est une construction pratique pour représenter une séquence d'actions à appliquer comme une seule action atomique au sein d'une règle de politique. Dans de nombreux cas, les actions sont en relation les unes avec les autres et devraient être regardées comme des sous actions d'une action "logique". Un exemple d'une telle action logique est "formater & marquer" (c'est-à-dire, formater un certain flux en un ensemble de caractéristiques prédéfinies de bande passante et ensuite marquer ces paquets avec une certaine valeur de DSCP). Cette action logique est en fait composée de deux différentes actions de QS, qui devraient être effectuées dans un ordre bien défini et comme un ensemble complet.

La construction CompoundPolicyAction permet de créer une relation logique entre un certain nombre d'actions, et de définir la logique d'activation associée à cette action logique.

La construction CompoundPolicyAction permet la réutilisation de ces actions complexes, en les mémorisant dans un ReusablePolicyContainer et en les réutilisant dans différentes règles de politique. Noter qu'une action composée peut aussi être agrégée par une autre action composée.

Comme c'était le cas avec la CompoundPolicyCondition, les extensions à PCIM pour introduire les actions de politique composées sont relativement directes. Cette fois le but est d'appliquer la propriété ActionOrder à partir de l'agrégation PolicyActionInPolicyRule de PCIM en une action composée qui agrège de plus simples actions. Les changements suivants sont requis :

- Créer une nouvelle agrégation PolicyActionInPolicyAction, avec la même propriété ActionOrder que PolicyActionInPolicyRule. La façon la plus propre pour faire cela est de déplacer la propriété dans une nouvelle super classe abstraite d'agrégation PolicyActionStructure, à partir de laquelle l'agrégation existante PolicyActionInPolicyRule et une nouvelle agrégation PolicyActionInPolicyAction sont déduites.

- Il est aussi nécessaire de définir une sous classe concrète CompoundPolicyAction de PolicyAction, pour introduire la propriété SequencedActions. Cette propriété a la même fonction, et travaille exactement de la même façon que la propriété correspondante actuellement définie dans PCIM pour la classe PolicyRule.

- Finalement, une nouvelle propriété ExecutionStrategy est nécessaire pour les deux classes PCIM PolicyRule et la nouvelle classe CompoundPolicyAction. Cette propriété permet à l'administrateur de politique de spécifier comment le PEP devrait se comporter dans le cas où plusieurs actions sont agrégées par une PolicyRule ou par une CompoundPolicyAction.

Les définitions de classe et de propriété pour représenter les actions de politique composées sont à la Section 6.

5.8. Variables et valeurs

Les paragraphes qui suivent introduisent plusieurs concepts en rapport, incluant les variables de politique *(PolicyVariables)* et les valeurs de politique *(PolicyValues* SimplePolicyConditions, et SimplePolicyActions*)* (et leurs nombreuses sous classes).

5.8.1 Conditions de politique simples

La classe SimplePolicyCondition modèle les expressions booléennes élémentaires de la forme "(<variable> MATCH <valeur>)". La relation 'MATCH', qui est implicite dans le modèle, est interprétée sur la base de la variable et de la valeur. Le paragraphe 5.8.3 explique la sémantique de l'opérateur 'MATCH'. Des expressions booléennes d'une complexité arbitraire peuvent être formées en enchaînant à la suite un nombre quelconque de conditions simples en utilisant des opérateurs relationnels. Les conditions simples individuelles peuvent aussi être niées. Des expressions booléennes de complexité arbitraire sont modélisées par la classe CompoundPolicyCondition (décrite au paragraphe 5.7.1).

Par exemple, l'expression "SourcePort == 80" *(accès de source = 80)* peut être modélisée par une condition simple. Dans cet exemple, 'SourcePort' est une variable, '==' est l'opérateur relationnel qui note la relation d'égalité (qui est généralisée par PCIMe en une relation "MATCH") et '80' est une valeur d'entier. L'interprétation complète d'une condition simple dépend des liens de la variable. Le paragraphe 5.8.5 décrit les variables et leurs règles de liaison.

La classe SimplePolicyCondition précise la structure de base de la classe PolicyCondition définie dans PCIM en utilisant la paire (<variable>, <valeur>) pour former la condition. Noter que l'opérateur entre la variable et la valeur est toujours impliqué dans PCIMe : il ne fait pas partie de la notation formelle.

La variable spécifie l'attribut d'un objet qui devrait correspondre lors de l'évaluation de la condition. Par exemple, pour un modèle de QS, cet objet pourrait représenter le flux, c'est-à-dire, être soumis à des conditions. Un ensemble de variables prédéfinies qui couvre les attributs de réseau couramment utilisés pour le filtrage est introduit dans PCIMe, pour favoriser l'interopérabilité. Cette liste couvre les attributs IP de couche 3 tels que les adresses réseau IP, les protocoles et les accès, ainsi qu'un ensemble d'attributs de couche 2 (par exemple, les adresses MAC).

La variable de lien est confrontée à une valeur pour produire le résultat booléen. Par exemple, dans la condition "L'adresse IP de source du flux appartient au sous réseau 10.1.x.x", une variable d'adresse IP de source est confrontée à une valeur de sous réseau de 10.1.x.x.

5.8.2 Utilisation des conditions de politique simples

Les conditions simples peuvent être utilisées directement dans les règles de politique, ou comme blocs de construction pour la création de conditions de politique composées.

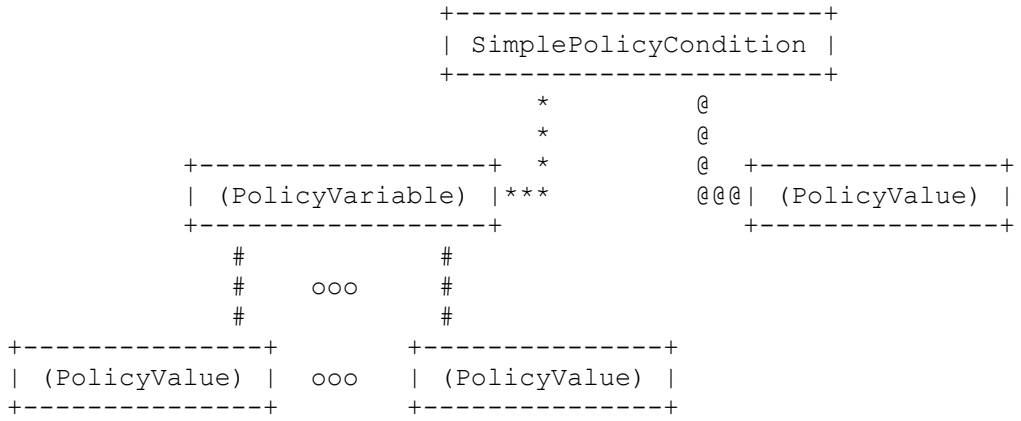
Une composition de condition simple DOIT appliquer la règle de conformité de type de données suivante : la propriété ValueTypes de la variable doit être compatible avec le type de la classe de valeur utilisée. La façon la plus simple (et la plus facile, du point de vue de l'utilisateur) de le faire est d'égaler le type de la classe de valeur avec le nom de la classe. En s'assurant que la propriété ValueTypes de la variable correspond au nom de la classe de valeur utilisée, on sait que les valeurs d'instances de variable et de valeur sont compatibles les unes avec les autres.

Composer une condition simple exige qu'une instance de la classe SimplePolicyCondition soit créée, et qu'existent des

instances de la variable et des classes de valeur qu'elle utilise. Noter que les instances de variable et/ou valeur peuvent déjà exister comme objets réutilisables dans un ReusablePolicyContainer approprié.

Deux agrégations sont utilisées afin de créer la paire (<variable>, <valeur>). L'agrégation PolicyVariableInSimplePolicyCondition met en rapport une SimplePolicyCondition avec une seule instance de variable. De même, l'agrégation PolicyValueInSimplePolicyCondition met en rapport une SimplePolicyCondition avec une seule instance de valeur. Les deux agrégations sont définies dans le présent document.

La Figure 6 décrit une SimplePolicyCondition avec sa variable et valeur associées. Deux instances de PolicyValue sont aussi montrées qui identifient les valeurs que la variable peut assumer.



Légende :
**** PolicyVariableInSimplePolicyCondition
@@@@ PolicyValueInSimplePolicyCondition
ExpectedPolicyValuesForVariable

Figure 6 : SimplePolicyCondition

Note : Les noms de classe entre parenthèses notent les sous classes. Les classes nommées dans la figure sont abstraites, et ne peuvent donc pas être elles-mêmes instanciées.

5.8.3 Opérateur Condition simple

Une condition simple modèle une expression booléenne élémentaire de la forme "variable MATCH valeur". Cependant, la notation formelle de SimplePolicyCondition, avec ses associations, modèle seulement une paire (<variable>, <valeur>). L'opérateur 'MATCH' n'est pas directement modélisé – il est implicite. De plus, cet opérateur 'MATCH' implicite porte une sémantique surchargée.

Par exemple, dans la condition simple "DestinationPort MATCH '80'", l'interprétation de l'opérateur 'MATCH' est l'égalité (l'opérateur 'égal'). En clair, une interprétation différente est nécessaire dans les cas suivants :
o "DestinationPort MATCH {'80', '8080'}" -- l'opérateur est 'EST ÉTABLI COMME MEMBRE'.
o "DestinationPort MATCH {'1 to 255'}" -- l'opérateur est 'DANS UNE GAMME D'ENTIERS'.
o "SourceIPAddress MATCH 'MyCompany.com'" -- l'opérateur est 'ADRESSE IP COMME RÉSOLUE PAR LE DNS'.

Les exemples ci-dessus illustrent la nature implicite, dépendante du contexte de l'opérateur 'MATCH'. L'interprétation dépend des instances réelles de variable et de valeur dans la condition simple. L'interprétation est toujours déduite de l'instance de variable et valeur liées associée à la condition simple. Le texte qui accompagne la classe de valeur et la définition implicite de variable est utilisée pour interpréter la sémantique de la relation 'MATCH'. Dans la liste qui suit, on définit des correspondances génériques (indépendantes du type).

Les PolicyValues peuvent être sur plusieurs champs, et chaque champ peut contenir une gamme de valeurs. La même égalité tient pour les PolicyVariables. Fondamentalement, on doit traiter avec des valeurs seules (singleton), des gammes ([limite inférieure .. limite supérieure]), et des ensembles (a,b,c). Donc, indépendamment du type de variable et de valeur, l'ensemble suivant de règles génériques correspondantes est défini pour l'opérateur 'MATCH'.

o un singleton correspond à un singleton -> la règle de correspondance est définie dans le type.

o un singleton correspond à une gamme [limite inférieure .. limite supérieure] -> la correspondance évalue la vérité, si le

singleton correspond à la limite inférieure ou à la limite supérieure ou à une valeur intermédiaire.

- un singleton correspond à un ensemble -> la confrontation s'évalue à vrai si la valeur du singleton correspond à un des composants de l'ensemble, où un composant peut être un singleton ou une autre gamme.

- une gamme [A..B] correspond à un singleton -> c'est vrai si A correspond à B correspond à un singleton.

- une gamme [A..B] correspond à une gamme [X..Y] -> la confrontation s'évalue à vrai si toutes les valeurs de la gamme [A..B] sont aussi dans la gamme [X..Y]. Par exemple, [3..5] confronté à [1..6] s'évalue à vrai, tandis que [3..5] confronté à [4..6] s'évalue à faux.
- une gamme [A..B] correspond à un ensemble (a,b,c, ...) -> la confrontation s'évalue à vrai si toutes les valeurs dans la gamme [A..B] font partie de l'ensemble. Par exemple, la gamme [2..3] confrontée à l'ensemble ([1..2],3) s'évalue à vrai, ainsi que la confrontation de la gamme [2..3] à l'ensemble (2,3), et de la gamme [2..3] à l'ensemble ([1..2],[3..5]).

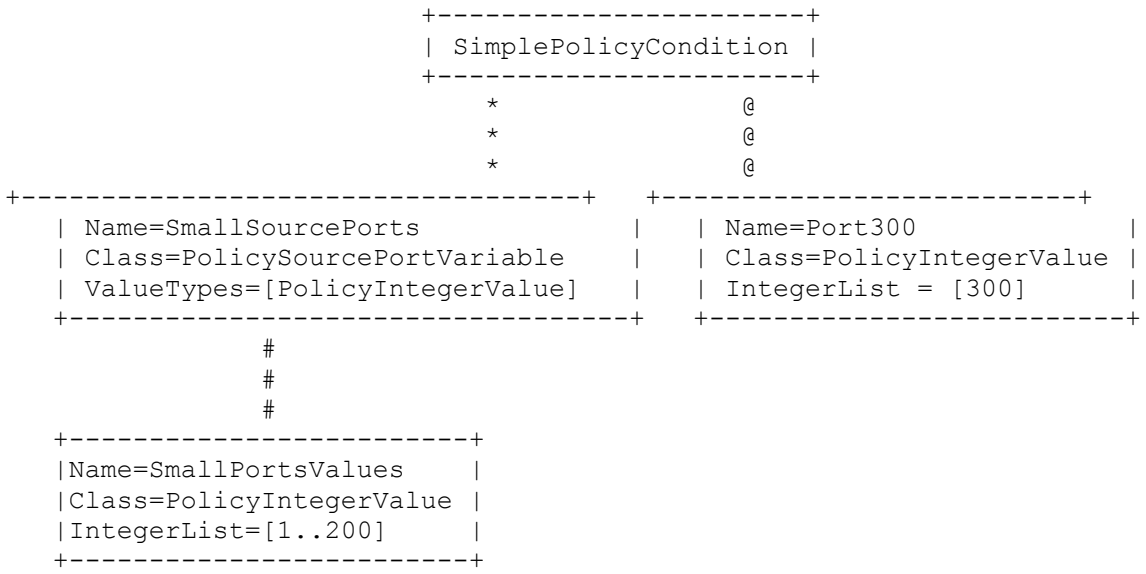
- l'ensemble (a,b,c, ...) correspond à un singleton -> vrai si a correspond à b qui correspond à c qui correspond à ... qui correspond à un singleton.

- un ensemble correspond à une gamme -> la confrontation s'évalue à vrai si toutes les valeurs dans l'ensemble font partie de la gamme. Par exemple, la confrontation de l'ensemble (2,3) à la gamme [1..4] s'évalue à vrai.

- un ensemble (a,b,c,...) correspond à un ensemble (x,y,z,...) -> la confrontation s'évalue à vrai si toutes les valeurs de l'ensemble (a,b,c,...) font partie de l'ensemble (x,y,z,...). Par exemple, l'ensemble (1,2,3) confronté à l'ensemble (1,2,3,4) s'évalue à vrai. La confrontation de l'ensemble (1,2,3) à l'ensemble (1,2) s'évalue à faux.

Les variables peuvent contenir divers types (paragraphe 6.11.1). Sauf mention contraire, le type de la valeur liée à la variable au moment de l'évaluation de la condition et le type de la valeur de l'instance de PolicyValue doivent être les mêmes. Si ils diffèrent, la condition s'évalue alors à FAUX.

L'association ExpectedPolicyValuesForVariable *(valeurs de politique attendues pour une variable)* spécifie l'ensemble attendu des valeurs qui peuvent correspondre pour une variable dans une simple condition. En utilisant cette association, un accès de source ou de destination peut être limité à la gamme 0-200, une adresse IP de source ou de destination peut être limitée à une liste spécifiée de valeurs d'adresses IPv4, etc.



Légende :
**** : PolicyVariableInSimplePolicyCondition
@@@@ : PolicyValueInSimplePolicyCondition
: ExpectedPolicyValuesForVariable

Figure 7 : SimplePolicyCondition invalide

La capacité à exprimer ces limitations apparaît dans le modèle pour prendre en charge la validation d'une SimplePolicyCondition avant son déploiement sur un point de mise en application. Un outil de gestion de politique, par exemple, NE DEVRAIT PAS accepter la SimplePolicyCondition montrée à la Figure 7. Cependant, si une règle de politique

contenant cette condition apparaît à un point de mise en application, les valeurs attendues ne jouent aucun rôle dans la détermination de l'évaluation de la condition à vrai ou faux. Donc, dans cet exemple, la SimplePolicyCondition s'évalue à vrai si l'accès de source pour le paquet considéré est 300, et elle s'évalue à faux autrement.

5.8.4 SimplePolicyActions

La classe SimplePolicyAction modèle le fonctionnement de l'ensemble élémentaire. "SET <variable> TO <valeur>". L'opérateur SET DOIT écraser une vieille valeur de la variable. Dans le cas où la variable à mettre à jour est multi valeurs, la seule opération de mise à jour définie est un remplacement complet de toutes les valeurs précédentes par un nouvel ensemble. En d'autres termes, il n'y a pas d'opération Add *(ajout)* ou Remove *(suppression)* [à/de l'ensemble des valeurs] définie pour SimplePolicyActions.

Par exemple, l'action "set DSCP to EF" *(régler DSCP à EF)* peut être modélisée par une simple action. Dans cet exemple, 'DSCP' est une variable implicite qui se réfère au champ DSCP de l'en-tête du paquet IP. 'EF' est une valeur d'entier ou de chaîne binaire (6 bits). L'interprétation complète d'une action simple dépend du lien de la variable.

La classe SimplePolicyAction précise la structure de base de la classe PolicyAction définie dans PCIM, en spécifiant le contenu de l'action en utilisant la paire (<variable>, <valeur>) pour former l'action. La variable spécifie les attributs d'un objet. La valeur de cet attribut est réglée à la valeur spécifiée dans <valeur>. Le choix de l'objet est une fonction du type de variable impliquée. Voir aux paragraphes 5.8.6 et 5.8.7, respectivement, les détails sur le choix d'objet pour les variables de politique liées explicitement et implicitement.

SimplePolicyActions peut être utilisé directement dans des règles de politique, ou comme éléments de construction pour créer des CompoundPolicyActions *(actions de politique composées)*.

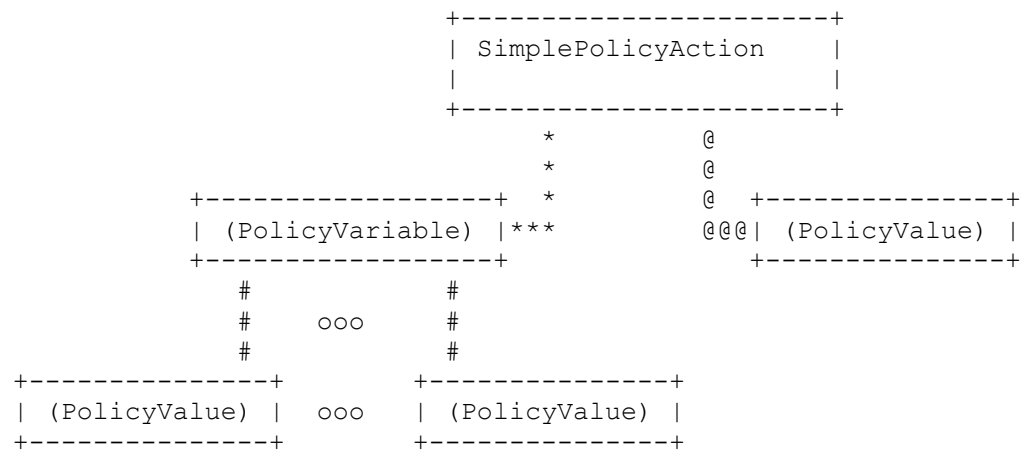
L'opération SET n'est valide que si la liste des types de la variable (propriété ValueTypes de PolicyImplicitVariable) inclut le type spécifié de la valeur. La conversion des valeurs d'une représentation dans une autre n'est pas définie. Par exemple, une variable du type IPv4Address ne peut pas être réglée à une chaîne contenant un nom DNS. Les conversions font partie d'une transposition spécifique de la mise en œuvre du modèle.

Comme c'était le cas avec SimplePolicyConditions, le rôle des valeurs attendues pour les variables qui apparaissent dans SimplePolicyActions est pour la validation, avant le moment d'exécution d'une action. Les valeurs attendues ne jouent aucun rôle dans l'exécution de l'action.

Composer une action simple exige qu'une instance de la classe SimplePolicyAction soit créée, et aussi qu'existent des instances des classes de variable et de valeur qu'elle utilise. Noter que les instances de variable et/ou de valeur peuvent déjà exister comme objets réutilisables dans un ReusablePolicyContainer approprié.

Deux agrégations sont utilisées pour créer la paire (<variable>, <valeur>). L'agrégation PolicyVariableInSimplePolicyAction met en rapport une SimplePolicyAction avec une seule instance de variable. De même, l'agrégation PolicyValueInSimplePolicyAction met en rapport une SimplePolicyAction avec une seule instance de valeur. Les deux agrégations sont définies dans le présent document.

La Figure 8. décrit une SimplePolicyAction avec sa variable et sa valeur associées.



Légende des agrégations :
**** PolicyVariableInSimplePolicyAction
@@@@ PolicyValueInSimplePolicyAction
ExpectedPolicyValuesForVariable

Figure 8 : SimplePolicyAction

5.8.5 Variables de politique

Une variable représente de façon générique des informations qui changent (ou "varient") c'est-à-dire réglées ou évaluées par un logiciel. En matière de politique, les conditions et actions peuvent abstraire des informations comme des "variables de politique" pour être évaluées dans des expressions logiques, ou établies par des actions.

PCIMe définit deux types de PolicyVariables, PolicyImplicitVariables et PolicyExplicitVariables. La différence sémantique entre ces classes se fonde sur le contexte de modélisation. Les variables explicites sont liées à des constructions exactes du modèle, tandis que les variables implicites sont définies et évaluées en dehors d'un modèle. Par exemple, on peut imaginer de vérifier une PolicyCondition pour voir si la propriété Status d'un ManagedSystemElement *(élément de système géré)* CIM a la valeur "Erreur". La propriété Status est une variable de politique explicitement définie (c'est-à-dire, elle est définie dans le contexte du schéma de CIM, et évaluée dans le contexte d'une instance spécifique). D'un autre côté, les paquets du réseau ne sont pas explicitement modélisés ou instanciés, dans la mesure ou (pour l'instant) on ne s'est pas intéressé à la gestion au niveau du paquet. Donc, une condition de politique peut ne faire aucune référence explicite à une construction d'un modèle qui représente l'adresse de source d'un paquet du réseau. Dans ce cas, une variable de politique implicite est définie, pour permettre l'évaluation ou la modification de l'adresse de source d'un paquet.

5.8.6 Variables de politique explicitement limitées

Les variables de politique explicitement limitées indiquent les noms de classe et de propriété des constructions du modèle à évaluer ou établir. Le schéma CIM définit et fixe les contraintes des valeurs "appropriées" pour les variables (c'est-à-dire, les propriétés du modèle) en utilisant les types de données et d'autres informations comme les qualificatifs de classe/propriété.

Une PolicyExplicitVariable est "explicite" parce que la sémantique de son modèle est exactement définie. Elle N'EST PAS explicite à cause d'un lien exact à une instance d'objet particulière. Si les PolicyExplicitVariable étaient liées à des instances (via soit des associations, soit une propriété d'identification d'objet dans la classe elle-même) on transgresserait alors les règles spécifiques de l'élément. D'un autre côté, si on spécifie seulement le contexte du modèle de l'objet (nom de classe et de propriété) mais on laisse le fait de lier au cadre de la politique (par exemple, en utilisant des rôles de politique) une plus grande souplesse résulte pour les règles soit générales soit spécifiques de l'élément.

Par exemple, une règle spécifique d'un élément est obtenue par une condition (paire (<variable>, <valeur>)) qui définit dans CIM LogicalDevice DeviceID="12345". Autrement, si un rôle de politique d'une règle de politique est "appareil de bordure" et si la condition (paire (<variable>, <valeur>)) est Status="Error", une règle générale résulte en une erreur pour tous les appareils de bordure.

Actuellement, le seul lien pour une PolicyExplicitVariable définie dans PCIMe est avec les instances choisies par les rôles de politique. Pour une telle instance, une SimplePolicyCondition qui agrège la PolicyExplicitVariable ne s'évalue à Vrai que si et seulement si TOUT ce qui suit est vrai :

- L'instance choisie est de la classe identifiée par la propriété ModelClass de la variable, ou d'une sous classe de cette classe.
- L'instance choisie a la propriété identifiée par la propriété ModelProperty de la variable.
- La valeur de cette propriété dans l'instance correspond à la valeur spécifiée dans la valeur de politique agrégée par la condition.

Dans tous les autres cas, la SimplePolicyCondition s'évalue à Faux.

Pour le cas où une SimplePolicyAction agrège une PolicyExplicitVariable, la propriété indiquée dans l'instance choisie est réglée à la valeur représentée par la PolicyValue que la SimplePolicyAction agrège aussi. Cependant, si l'instance choisie n'est pas de la classe identifiée par la propriété ModelClass de la variable, ou d'une sous classe de cette classe, l'action n'est alors pas effectuée. Dans ce cas, la SimplePolicyAction n'est traitée ni comme une action exécutée avec succès (pour la stratégie d'exécution Faire jusqu'à la réussite) ni comme une action échouée (pour la stratégie d'exécution Faire jusqu'à l'échec). Au lieu de cela, les actions restantes pour la règle de politique, si il y en a, sont exécutées comme si cette SimplePolicyAction n'était pas présente du tout dans la liste des actions agrégées par la règle.

Les variables explicites seraient plus puissantes si elles pouvaient atteindre les instances qui s'y rapportent au delà des instances choisies par les rôles de politique. Cependant, représenter une règle de politique impliquant des telles variables d'une manière générale exigerait quelque chose qui commencerait à ressembler beaucoup à un langage complet de politique. Il est clair qu'un tel langage sort du domaine d'application de PCIMe, bien qu'il puisse faire l'objet d'un futur document.

En restreignant beaucoup de leur généralité, il serait possible que les variables explicites dans PCIMe aillent légèrement au delà de l'instance choisie. Par exemple, si une instance choisie était exactement en rapport avec une instance d'une autre classe via une classe d'association particulière, et si le but de la règle de politique était à la fois de vérifier une propriété de cette instance en rapport et d'établir une propriété de cette même instance, il serait alors possible de représenter la condition et l'action de la règle en utilisant PolicyExplicitVariables. Plutôt que de traiter ce cas spécifique avec des variables explicites, il a

cependant été décidé de le grouper avec le cas plus général, et de le traiter si et quand un langage de politique sera défini.

Se référer au paragraphe 6.10 pour la définition formelle de la classe PolicyExplicitVariable.

5.8.7 Variables de politique implicitement limitées

Les variables de politique implicitement liées définissent le type de données et la sémantique d'une variable. Cela détermine comment la variable est liée à une valeur dans une condition ou une action. D'autres instructions sont fournies pour spécifier les contraintes sur le type de données et/ou la valeur pour les variables implicitement liées.

PCIMe introduit une classe abstraite, PolicyImplicitVariable, pour modéliser les variables implicitement liées. Cette classe est dérivée de la classe abstraite PolicyVariable aussi définie dans PCIMe. Chacune des variables implicitement liées introduites par PCIMe (et celles qui sont introduites par les sous modèles spécifiques du domaine) DOIT être dérivée de la classe PolicyImplicitVariable. La raison de l'utilisation de ce mécanisme pour la modélisation est expliquée au paragraphe 5.8.9.

Un modèle d'informations de politique spécifique du domaine qui étend PCIMe peut définir des variables implicitement liées supplémentaires soit en les déduisant directement de la classe PolicyImplicitVariable, soit en précisant une classe de variables existante telle que SourcePort. Lorsque on précise une classe telle que SourcePort, les règles existantes de liaison, les contraintes de type ou de valeur peuvent être restreintes.

5.8.8 Structure et usage de variables prédéfinies

Une classe déduite de PolicyImplicitVariable pour modéliser une variable implicitement liée particulière DEVRAIT être construite d'une façon telle que son nom décrive la signification de la variable. Par exemple, une classe définie pour modéliser l'accès de source d'un flux TCP/UDP DEVRAIT avoir 'SourcePort' dans son nom.

PCIMe définit une association et un mécanisme d'objet général qui ensemble caractérisent chacune des variables implicitement liées qu'ils introduisent :

1. L'association ExpectedPolicyValuesForVariable définit l'ensemble de classes de valeurs qui pourraient correspondre à cette variable.
2. La liste des contraintes sur les valeurs que la PolicyVariable peut contenir (c'est-à-dire, les valeurs que la variable doit satisfaire) est définie par les propriétés appropriées d'une classe PolicyValue associée.

Dans l'exemple présenté ci-dessus, une PolicyImplicitVariable représente le SourcePort du trafic entrant. La propriété ValueTypes d'une instance de cette classe va contenir le nom de classe PolicyIntegerValue. Ceci contraint par lui-même le type de données de l'instance de SourcePort à être un entier. Cependant, on peut contraindre encore plus les valeurs particulières que peut contenir la variable SourcePort en entrant des gammes valides dans la propriété IntegerList de l'instance de PolicyIntegerValue (0 - 65535 dans le présent document).

La combinaison du VariableName et de l'association ExpectedPolicyValuesForVariable donne un ensemble cohérent et extensible de métadonnées qui définit la sémantique des variables utilisées pour former des conditions de politique. Comme l'association ExpectedPolicyValuesForVariable pointe sur une instance de PolicyValue, toute valeur exprimable dans la classe PolicyValue peut être utilisée pour contraindre les valeurs que peut porter PolicyImplicitVariable. Par exemple :

- La propriété ValueTypes peut être utilisée pour s'assurer que seules les classes appropriées sont utilisées dans l'expression. Par exemple, la variable SourcePort ne sera permise pour aucun type de PolicyIPv4AddrValue, car les accès de source ont une sémantique différente de celle des adresses IP et ne peuvent pas y correspondre. Cependant, des types de valeur d'entier sont permises car la propriété ValueTypes contient la chaîne "PolicyIntegerValue", qui est le nom de classe des valeurs d'entier.
- L'association ExpectedPolicyValuesForVariable assure aussi que la sémantique spécifique de la variable est mise en application (par exemple, la variable SourcePort peut inclure une contrainte associée à un objet valeur qui définit une gamme d'entiers spécifique qui devrait être satisfaite).

5.8.9 Raisons du modelage de variables implicites comme classes

Une variable implicitement liée peut être modélisée de plusieurs façons, incluant une seule classe avec un énumérateur pour chaque variable implicitement liée individuelle et une classe abstraite étendue pour chaque variable individuelle. Les raisons de l'utilisation d'un mécanisme d'héritage de classe pour spécifier des variables implicitement liées individuelles sont :

1. Il est facile à étendre. Un modèle d'information spécifique du domaine peut facilement étendre la classe PolicyImplicitVariable ou ses sous classes pour définir des variables spécifiques du domaine et du contexte. Par exemple, un modèle d'information de politique de QS spécifique d'un domaine peut introduire une classe de variables implicitement liées pour modéliser des applications en déduisant une classe qosApplicationVariable de la classe abstraite PolicyImplicitVariable.

2. L'introduction d'une seule classe structurelle pour les variables implicitement liées aurait à inclure une propriété énumératrice qui contienne toutes les variables implicitement liées individuelles possibles. Cela signifie qu'un modèle d'information spécifique d'un domaine qui souhaite introduire une variable implicitement liée doit étendre l'énumérateur même. Il en résulte de multiples définitions de la même classe, différant par les valeurs disponibles dans la classe d'énumérateur. Une définition, dans le présent document, inclurait les noms des variables implicitement liées communes, tandis qu'une seconde définition, dans le document du modèle d'information spécifique du domaine, pourrait inclure des valeurs supplémentaires ('qosApplicationVariable' dans l'exemple ci-dessus). Il ne serait même pas évident pour le développeur d'application qu'il existe plusieurs définitions de classe. Il serait encore plus difficile au développeur d'application de trouver réellement la classe correcte à utiliser.
3. De plus, une définition fondée sur un énumérateur exigerait que chaque valeur supplémentaire soit enregistrée auprès de l'IANA pour assurer la conformité aux normes. Cela rendrait le processus difficile à manier.
4. Un argument possible contre le mécanisme d'héritage serait le fait que cette approche résulte en une explosion de définitions de classes par rapport à une classe d'énumérateurs, qui n'introduit qu'une seule classe. Bien que, par lui-même, ce point ne condamne pas l'approche, on peut avancer que les modèles de données dérivés de ce modèle d'information peuvent être plus difficiles à optimiser pour les applications. On rejettera cet argument au motif que l'optimisation des applications est de moindre valeur pour un modèle d'information que la clarté et la facilité d'extension. De plus, il est difficile de prétendre que le modèle d'héritage fasse peser une charge insupportable à l'optimisation. Par exemple, un modèle de données peut quand même utiliser l'énumération pour noter les instances de variables pré définies et revendiquer la conformité à PCIMe, pour autant que le modèle de données puisse être transposé correctement en les définitions spécifiées dans le présent document.

5.8.10 Valeurs de politique

La classe abstraite PolicyValue est utilisée pour modéliser les valeurs et constantes utilisées dans les conditions de politique. Les différents types de valeurs sont déduits de cette classe, pour représenter les divers attributs requis. Les extensions de la classe abstraite PolicyValue, définies dans le présent document, fournissent une liste des valeurs pour les attributs de base des réseaux. Les valeurs peuvent être utilisées pour représenter des constantes comme valeurs désignées. Des valeurs désignées peuvent être conservées dans un conteneur de politiques réutilisables pour être réutilisées par plusieurs conditions. Des exemples de constantes incluent les accès bien connus, les protocoles bien connus, les adresses de serveurs, et autres concepts similaires.

Les sous classes de PolicyValue définissent trois types de valeurs de base : les scalaires, les gammes et les ensembles. Par exemple, un numéro d'accès bien connu pourrait être défini en utilisant la classe PolicyIntegerValue, définissant respectivement une seule valeur (80 pour HTTP), une gamme (80-88), ou un ensemble (80, 82, 8080) d'accès. Pour les détails, voir la définition de classe pour chaque type de valeur au paragraphe 6.14.

PCIMe définit les sous classes suivantes de la classe abstraite PolicyValue :

Classes d'utilisation générale :
- PolicyStringValue, *(valeur de chaîne de politique)*
- PolicyIntegerValue, *(valeur d'entier de politique)*
- PolicyBitStringValue *(valeur de chaîne binaire de politique)*
- PolicyBooleanValue. *(valeur booléenne de politique)*

Classes pour les valeurs de couche 3 du réseau :
- PolicyIPv4AddrValue, *(valeur d'adresse IPv4 de politique)*
- PolicyIPv6AddrValue. *(valeur d'adresse IPv6 de politique)*

Classes pour les valeurs de couche 2 du réseau :
- PolicyMACAddrValue. *(valeur d'adresse MAC de politique)*

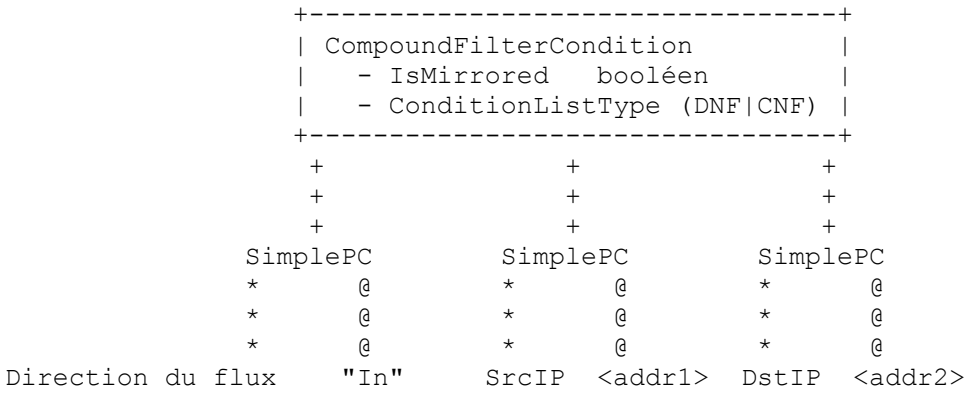
Pour les détails, voir le paragraphe de définition de classe de chaque classe au paragraphe 6.14.

5.9 Filtrage de paquets

PCIMe contient deux mécanismes pour représenter les filtres de paquet. Le plus général, appelé ici le modèle de niveau domaine, exprime les filtres de paquet en termes de variables de politique et valeurs de politique. L'autre mécanisme, appelé ici le modèle de niveau appareil, exprime les filtres de paquet d'une façon qui se transpose plus directement en champs du paquet auquel les filtres sont appliqués. Bien qu'il soit possible de transposer ces deux représentations de filtres de paquet, aucune transposition n'est fournie par PCIMe lui-même.

5.9.1 Filtres de paquet de niveau domaine

En plus du remplissage des trous de l'infrastructure de politique globale, PCIMe propose un seul mécanisme pour exprimer les filtres de paquet de niveau domaine dans les conditions de politique. Cela se fait en réponse au souci que bien que dans la "vague" initiale de sous modèles dérivés de PCIM, tous soient filtrants sur les paquets IP, chacun le faisait d'une façon légèrement différente. PCIMe propose une façon commune pour exprimer les filtres de paquet IP. La figure suivante illustre comment les conditions de filtrage de paquet sont exprimées dans PCIMe.



Légende des agrégations :
++++ PolicyConditionInPolicyCondition
**** PolicyVariableInSimplePolicyCondition
@@@@ PolicyValueInSimplePolicyCondition

Figure 9 : Filtrage de paquet dans les conditions de politique

Dans la Figure 9, chaque SimplePolicyCondition représente un seul champ à filtrer sur : Adresse de source IP, Adresse de destination IP, Accès de source, etc. Une SimplePolicyCondition supplémentaire indique la direction dans laquelle un paquet voyage sur une interface, entrant ou sortant. À cause de la condition Direction du flux, il faut faire attention en agrégeant un ensemble de SimplePolicyConditions en une CompoundFilterCondition. Autrement, la CompoundPolicyCondition résultante pourrait correspondre à tous les paquets entrants, ou tous les paquets sortants, alors que ce n'est probablement pas ce qu'on voulait.

Les SimplePolicyConditions individuelles peuvent être niées lorsque elles sont agrégées par une CompoundFilterCondition.

CompoundFilterCondition est une sous-classe de CompoundPolicyCondition. Elle introduit une propriété supplémentaire, la propriété booléenne IsMirrored. L'objet de cette propriété est de permettre qu'une seule CompoundFilterCondition corresponde aux paquets qui voyagent dans les deux directions sur une connexion de niveau supérieur telle qu'une session TCP. Lorsque cette propriété est VRAI, les paquets supplémentaires correspondent à un filtre, au delà de ceux qui y auraient correspondu à l'origine. Un exemple va illustrer comment fonctionne cette propriété.

Supposons qu'on ait une CompoundFilterCondition qui agrège les trois filtres suivants, qui sont ajoutés ensemble par l'opérateur logique ET :
- FlowDirection = "In"
- Source IP = 9.1.1.1
- Source Port = 80

Sans considérer si IsMirrored est VRAI ou FAUX, les paquets entrants vont satisfaire cette CompoundFilterCondition si leur adresse IP de source = 9.1.1.1 et leur accès de source = 80. Si IsMirrored est VRAI, un paquet sortant va cependant aussi satisfaire la CompoundFilterCondition si son adresse IP de destination = 9.1.1.1 et son accès de destination = 80.

IsMirrored "bascule" les champs d'en-tête de paquet de source/destination suivants :
- FlowDirection "In" / FlowDirection "Out"
- Source IP address / Destination IP address
- Source port / Destination port
- Source MAC address / Destination MAC address
- Source [layer-2] SAP / Destination [layer-2] SAP.

5.9.2 Filtres de paquet de niveau appareil

Au niveau appareil, les filtres d'en-tête de paquet sont représentés par deux sous classes de la classe abstraite FilterEntryBase : IpHeadersFilter et 8021Filter. Les sous modèles de PCIMe peuvent définir d'autres sous classes de FilterEntryBase en plus de ces deux là ; ICPM [RFC3585], par exemple, définit des sous classes pour les filtres spécifiques de IPsec.

Les instances des sous classes de FilterEntryBase ne sont pas utilisées directement comme filtres. Elles sont toujours agrégées dans une FilterList par l'agrégation EntriesInFilterList. Pour PCIMe et ses sous modèles la propriété EntrySequence dans cette agrégation prend toujours sa valeur par défaut de '0' qui indique que les entrées de filtre agrégées sont ajoutées ensemble par l'opération logique ET.

La classe FilterList inclut une propriété d'énumération Direction, représentant la direction du flux de trafic auquel la FilterList est à appliquer. La valeur Mirrored(4) pour Direction représente exactement la même chose que le booléen IsMirrored dans CompoundFilterCondition. Voir les détails au paragraphe 5.9.1.

5.10 Conformité à PCIM et PCIMe

Parce que PCIM et PCIMe fournissent les classes centrales de modélisation des politiques, ils ne sont en général pas suffisants par eux-mêmes pour représenter les règles réelles de politique. Des sous modèles, comme QPIM et ICPM, fournissent les moyens d'exprimer les règles de politique, en définissant des sous classes des classes définies dans PCIM et PCIMe, et/ou en indiquant comment les PolicyVariables et PolicyValues définies dans PCIMe peuvent être utilisées pour exprimer les conditions et actions applicables au sous modèle.

Un sous modèle particulier ne sera pas, en général, nécessaire pour utiliser chaque élément défini dans PCIM et PCIMe. Pour les éléments qu'il n'utilise pas, un sous modèle DEVRAIT rester silencieux sur le fait que ses mises en œuvre doivent prendre en charge l'élément, ne doivent pas prendre en charge l'élément, devrait prendre en charge l'élément, etc. Pour les éléments qu'il utilise, un sous modèle DEVRAIT indiquer quels éléments ses mises en œuvre doivent prendre en charge, quels éléments elles devraient prendre en charge, et quels éléments elles peuvent prendre en charge.

PCIM et PCIMe eux-mêmes définissent simplement les éléments qui peuvent être utiles aux sous modèles. Ces documents restent silencieux sur le fait que les mises en œuvre doivent prendre en charge un élément, devrait le prendre en charge, etc.

Ce modèle (et les sous modèles dérivés) définit les conditions et actions qui sont utilisées par les règles de politique. Bien que les conditions et actions définies ici soient directes et qu'on puisse supposer qu'elles auront un large soutien, lorsque les sous modèles seront développés, il est vraisemblable que des situations se feront jour où des conditions ou actions spécifiques ne seront pas acceptées par certaines parties du système d'exécution de politique. De même, il peut aussi se produire des situations où les règles contiendront des erreurs syntaxiques ou sémantiques.

On devrait comprendre que le comportement et l'effet de conditions ou actions indéfinies ou incorrectement définies ne sont pas prescrits par ce modèle d'information. Bien qu'il serait utile qu'ils soient prescrits, les variations de mise en œuvre restreignent la capacité de ce modèle d'information à contrôler les effets. Par exemple, si une mise en œuvre a seulement détecté qu'un PEP ne pourra pas appliquer une certaine action sur ce PEP, il sera très difficile de déclarer qu'une telle défaillance devrait affecter d'autres PEP, ou le processus du PDP. D'un autre côté, si le PDP détermine qu'il ne peut pas correctement évaluer une condition, cette défaillance peut bien affecter toutes les applications des règles contenantes.

6. Définitions de classes

Les définitions suivantes s'ajoutent à celles de PCIM. Les définitions de PCIM qui ne sont pas DÉCONSEILLÉES dans le présent document font toujours partie du modèle d'information de cœur de politique actuel.

6.1 Classe abstraite "PolicySet"

PolicySet est une classe abstraite qui peut grouper des politiques en un ensemble structuré de politiques.

NOM	PolicySet
DESCRIPTION	Classe abstraite qui représente un ensemble de politiques qui forment un ensemble cohérent. L'ensemble des politiques contenues a une stratégie de décision commune et un ensemble commun de rôles de politique. Les sous classes incluent PolicyGroup et PolicyRule.
DÉRIVÉ DE	Policy

ABSTRAITE VRAI
PROPRIÉTÉS PolicyDecisionStrategy ; PolicyRoles

La propriété PolicyDecisionStrategy spécifie la méthode d'évaluation pour les groupes et règles de politique contenus dans l'ensemble de politiques.

NOM PolicyDecisionStrategy
DESCRIPTION Méthode d'évaluation utilisée pour les politiques contenues dans le PolicySet. FirstMatching applique les actions de la première règle qui s'évalue à VRAI ; AllMatching applique les actions de toutes les règles qui s'évaluent à VRAI.
SYNTAXE uint16
VALEURS 1 [FirstMatching], 2 [AllMatching]
PAR DÉFAUT 1 [FirstMatching]

La définition de PolicyRoles est inchangée par rapport à PCIM. Il est, cependant, déplacé de la classe Policy à la superclasse PolicySet.

6.2 Mise à jour de la classe "PolicyGroup"de PCIM

La classe PolicyGroup est déplacée, de sorte qu'elle est maintenant dérivée de PolicySet.

NOM PolicyGroup
DESCRIPTION Conteneur pour un ensemble de PolicyRules et PolicyGroups en rapports.
DÉRIVÉ DE PolicySet
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.3 Mise à jour de la classe "PolicyRule" de PCIM

La classe PolicyRule est déplacée, de sorte qu'elle est maintenant dérivée de PolicySet. La propriété Priority est aussi déconseillée dans PolicyRule, et PolicyRoles est maintenant hérité de la classe parente PolicySet. Finalement, une nouvelle propriété ExecutionStrategy est introduite, en parallèle à la propriété du même nom dans la classe CompoundPolicyAction.

NOM PolicyRule
DESCRIPTION Classe centrale pour représenter la sémantique "Si condition alors action" associée à une règle de politique.
DÉRIVÉ DE PolicySet
ABSTRAITE FAUX
PROPRIÉTÉS Enabled ; ConditionListType ; RuleUsage ; Priority DÉCONSEILLÉ POUR PolicySetComponent.Priority ET POUR PolicySetInSystem.Priority ; Mandatory ; SequencedActions ; ExecutionStrategy

La propriété ExecutionStrategy définit la stratégie d'exécution comme étant utilisée sur les suites d'actions agrégées par cette PolicyRule. (Une propriété équivalente ExecutionStrategy est aussi définie pour la classe CompoundPolicyAction, pour fournir la même indication pour la suite d'actions agrégées par une CompoundPolicyAction.) Le présent document définit trois stratégies d'exécution :

Do Until Success *(faire jusqu'à réussite)* :exécute les actions selon un ordre prédéfini, jusqu'à la réussite de l'exécution d'une seule action.

Do All *(faire tout)* : exécute TOUTES les actions qui font partie de l'ensemble modélisé, selon leur ordre prédéfini, et continue de le faire, même si une ou plusieurs des actions échouent.

Do Until Failure *(faire jusqu'à un échec)* : exécute les actions selon un ordre prédéfini, jusqu'à un premier échec dans l'exécution d'une seule sous action.

La définition de la propriété est la suivante :

NOM ExecutionStrategy
DESCRIPTION Une énumération indiquant comment interpréter l'ordre des actions pour les actions agrégées par cette PolicyRule.
SYNTAXE uint16 (ENUM, {1=Do Until Success, 2=Do All, 3=Do Until Failure})
PAR DÉFAUT Do All (2)

6.4 Classe "SimplePolicyCondition"

Une simple condition de politique est composée d'un triplet ordonné:

<Variable> MATCH <Valeur>

On ne fournit pas de modélisation formelle de l'opérateur MATCH. La relation 'match' est implicite. De telles conditions simples sont évaluées en répondant à la question :

Est ce que <variable> correspond à <valeur> ?

La relation 'match' est à interpréter par l'analyse des instances de variable et de valeur associées à la condition simple.

Les conditions simples sont les blocs de construction de conditions booléennes plus complexes, modélisées par la classe CompoundPolicyCondition.

La classe SimplePolicyCondition est dérivée de la classe PolicyCondition définie dans PCIM.

Une variable et une valeur doivent être associées à une simple condition pour en faire une condition significative, en utilisant, respectivement, les agrégations PolicyVariableInSimplePolicyCondition et PolicyValueInSimplePolicyCondition.

La définition de la classe est la suivante :

NOM	SimplePolicyCondition
DÉRIVÉ DE	PolicyCondition
ABSTRAITE	Faux
PROPRIÉTÉS	(aucune)

6.5 Classe "CompoundPolicyCondition"

Cette classe représente une condition de politique composée, formée par l'agrégation de plus simples conditions de politique.

NOM	CompoundPolicyCondition
DESCRIPTION	Une sous classe de PolicyCondition qui introduit la propriété ConditionListType, utilisée pour allouer la sémantique DNF / CNF aux conditions de politique subordonnées.
DÉRIVÉ DE	PolicyCondition
ABSTRAITE	FAUX
PROPRIÉTÉS	ConditionListType

La propriété ConditionListType est utilisée pour spécifier si la liste des conditions de politique associées à cette condition de politique composée est en forme disjonctive normale (DNF, *disjunctive normal form*) ou en forme conjonctive normale (CNF, *conjunctive normal form*). Si cette propriété n'est pas présente, le type de liste est par défaut DNF. La définition de la propriété est la suivante :

NOM	ConditionListType
DESCRIPTION	Indique si la liste des conditions de politique associée à cette règle de politique est en forme disjonctive normale (DNF) ou conjonctive normale (CNF).
SYNTAXE	uint16
VALEURS	DNF(1), CNF(2)
PAR DÉFAUT	DNF(1)

6.6 Classe "CompoundFilterCondition"

Cette sous classe de CompoundPolicyCondition introduit une propriété supplémentaire, le booléen IsMirrored. Cette propriété active ou désactive la "bascule" des champs source et destination correspondants dans une spécification de filtre.

NOM	CompoundFilterCondition
DESCRIPTION	Une sous classe de CompoundPolicyCondition qui introduit la propriété IsMirrored.
DÉRIVÉ DE	CompoundPolicyCondition
ABSTRAITE	FAUX

PROPRIÉTÉS IsMirrored

La propriété IsMirrored indique si les paquets qui "reflètent" une condition de filtre composé devraient être traités comme satisfaisant au filtre. La définition de la propriété est la suivante :

NOM IsMirrored
DESCRIPTION Indique si les paquets qui reflètent le filtre spécifié sont à traiter comme satisfaisant au filtre.
SYNTAXE booléen
PAR DÉFAUT FAUX

6.7 Classe "SimplePolicyAction"

La classe SimplePolicyAction modélise l'opération SET élémentaire. "SET <variable> TO <valeur>" *(régler la variable à une certaine valeur)*. L'opérateur SET DOIT écraser toute ancienne valeur de la variable.

Deux agrégations sont utilisées pour créer la paire <variable> <valeur>. L'agrégation PolicyVariableInSimplePolicyAction met en rapport une SimplePolicyAction avec une seule instance de variable. De même, l'agrégation PolicyValueInSimplePolicyAction met en rapport une SimplePolicyAction avec une seule instance de valeur. Les deux agrégations sont définies dans le présent document.

NOM SimplePolicyAction
DESCRIPTION Une sous classe de PolicyAction qui introduit la notion de "SET variable TO valeur".
DÉRIVÉ DE PolicyAction
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.8 Classe "CompoundPolicyAction"

La classe CompoundPolicyAction est utiliséE pour représenter une expression consistant en une séquence ordonnée de termes d'action. Chaque terme d'action est représenté comme une sous classe de la classe PolicyAction, définie dans [PCIM]. Les actions composées sont construites en associant les termes d'action dépendantes en utilisant l'agrégation PolicyActionInPolicyAction.

La définition de la classe est la suivante :

NOM CompoundPolicyAction
DESCRIPTION Une classe pour représenter la séquence des termes d'action. Chaque terme d'action est défini comme étant une sous classe de la classe PolicyAction.
DÉRIVÉ DE PolicyAction
ABSTRAITE FAUX
PROPRIÉTÉS SequencedActions ; ExecutionStrategy

C'est une classe concrète, qui est donc directement instanciable.

La propriété SequencedActions est identique à la propriété SequencedActions définie dans PCIM pour la classe PolicyRule.

La propriété ExecutionStrategy définit la stratégie d'exécution à utiliser sur la séquence d'actions associée à cette action composée. (Une propriété ExecutionStrategy équivalente est aussi définie pour la classe PolicyRule, pour fournir la même indication pour la séquence d'actions associée à une PolicyRule.) Le présent document définit trois stratégies d'exécution :

Do Until Success *(faire jusqu'à réussite)* : exécute les actions selon un ordre prédéfini, jusqu'à la réussite de l'exécution d'une seule action.

Do All *(faire tout)* : exécute TOUTES les actions qui font partie de l'ensemble modélisé, selon leur ordre prédéfini, et continue de le faire, même si une ou plusieurs des actions échouent.

Do Until Failure *(faire jusqu'à un échec)* : exécute les actions selon un ordre prédéfini, jusqu'à un premier échec dans l'exécution d'une seule sous action.

Comme une CompoundPolicyAction peut elle-même être agrégée soit par une PolicyRule, soit par une autre CompoundPolicyAction, son succès ou son échec ne sera pas une entrée de la stratégie d'exécution de l'entité agrégeante. Par conséquent, on spécifie les règles suivantes pour déterminer si une CompoundPolicyAction réussit ou échoue :

Si la stratégie d'exécution de l'action composée de politique est Faire jusqu'à un succès, alors :
- si une action composante réussit, la CompoundPolicyAction réussit ;
- si toutes les actions composantes échouent, la CompoundPolicyAction échoue.

Si la stratégie d'exécution de l'action composée de politique est Faire tout, alors :
- si toutes les actions composantes réussissent, la CompoundPolicyAction réussit ;
- si au moins une action composante échoue, la CompoundPolicyAction échoue.

Si la stratégie d'exécution de l'action composée de politique est Faire jusqu'à un échec, alors :
- si toutes les actions composantes réussissent, la CompoundPolicyAction réussit ;
- si au moins une action composante échoue, la CompoundPolicyAction échoue.

La définition de la propriété ExecutionStrategy est la suivante :

NOM	ExecutionStrategy
DESCRIPTION	Une énumération indiquant comment interpréter l'ordre des actions pour les actions agrégées par cette CompoundPolicyAction.
SYNTAXE	uint16 (ENUM, {1=Do Until Success, 2=Do All, 3=Do Until Failure})
PAR DÉFAUT	Do All (2)

6.9 Classe abstraite "PolicyVariable"

Les variables sont utilisées pour construire des conditions individuelles. La variable spécifie la propriété d'un flux ou d'un événement qui devrait être satisfaite lors de l'évaluation de la condition. Cependant, toutes les combinaisons d'une variable et d'une valeur ne créent pas une condition significative. Par exemple, une variable d'adresse de source IP ne peut pas être confrontée à une valeur qui spécifie un numéro d'accès. Une certaine variable sélectionne l'ensemble des types de valeurs confrontables.

Une variable peut avoir des contraintes qui limitent l'ensemble des valeurs au sein d'un type de valeur particulier qui peut lui être confronté dans une condition. Par exemple, une variable d'accès de source limite l'ensemble des valeurs pour représenter des entiers à la gamme de 0 à 65 535. Les entiers en dehors de cette gamme ne peuvent pas être confrontés à la variable d'accès de source, même si ils sont du type de données correct. Les contraintes pour une certaine variable sont indiquées par l'association ExpectedPolicyValuesForVariable.

PolicyVariable est une classe abstraite. Les classes de variables de contexte implicites et explicites sont définies comme des sous classes de la classe PolicyVariable. Un ensemble de variables implicites est défini aussi dans le présent document.

La définition de la classe est la suivante :

NOM	PolicyVariable
DÉRIVÉ DE	Policy
ABSTRAITE	VRAI
PROPRIÉTÉS	(aucune)

6.10 Classe "PolicyExplicitVariable"

Les variables de politique explicitement définies sont évaluées dans le contexte du schéma CIM et de ses modèles de construction. La classe PolicyExplicitVariable indique la propriété exacte du modèle à évaluer ou manipuler. Voir au paragraphe 5.8.6 une discussion complète de ce qui arrive lorsque les valeurs des propriétés ModelClass et ModelProperty dans une instance de cette classe ne correspondent pas aux caractéristiques de la construction du modèle évalué ou mis à jour.

La définition de la classe est la suivante :

NOM	PolicyExplicitVariable
DÉRIVÉ DE	PolicyVariable
ABSTRAITE	Faux
PROPRIÉTÉS	ModelClass, ModelProperty

6.10.1 Propriété à une seule valeur "ModelClass"

Cette propriété est une chaîne qui spécifie le nom de la classe dont la propriété est évaluée ou réglée comme PolicyVariable.

La propriété est définie comme suit :

NOM ModelClass
SYNTAXE Chaîne

6.10.2 Propriété à une seule valeur ModelProperty

Cette propriété est une chaîne qui spécifie le nom de la propriété, au sein de la ModelClass, qui est évaluée ou réglée comme une PolicyVariable. La propriété est définie comme suit :

NOM ModelProperty
SYNTAXE Chaîne

6.11 Classe abstraite "PolicyImplicitVariable"

Les variables de politique implicitement définies sont évaluées en dehors du contexte du schéma CIM et des constructions de son modèle. Des sous classes spécifient le type des données et la sémantiques des PolicyVariables.

L'interprétation et l'évaluation d'une PolicyImplicitVariable peut varier, selon le contexte particulier dans lequel elle est utilisée. Par exemple, une adresse "SourceIP" peut noter le champ Adresse de source d'un en-tête de paquet IP, ou l'adresse d'envoi délivrée par un message RSVP PATH.

La définition de la classe est la suivante :

NOM PolicyImplicitVariable
DÉRIVÉ DE PolicyVariable
ABSTRAITE Vrai
PROPRIÉTÉS ValueTypes[ ]

6.11.1 Propriété multi valeurs "ValueTypes"

Cette propriété est un ensemble de chaînes qui spécifient une liste non ordonnée de valeurs/types de données possibles qui peut être utilisée dans des conditions et actions simples, avec cette variable. Les types de valeurs sont spécifiés par leur nom de classe (sous classes de PolicyValue telles que PolicyStringValue). La liste des noms de classe permet à une application de chercher sur un nom spécifique, ainsi que de s'assurer que le type de données de la variable est le type correct.

La liste des types de valeur par défaut pour chaque sous classe de PolicyImplicitVariable est spécifiée dans la définition de cette variable.

La propriété est définie comme suit :

NOM ValueTypes
SYNTAXE Chaîne

6.12 Sous classes de "PolicyImplicitVariable" spécifiées dans PCIMe

Les sous classes suivantes de PolicyImplicitVariable sont définies dans PCIMe.

6.12.1 Classe "PolicySourceIPv4Variable"

NOM PolicySourceIPv4Variable
DESCRIPTION L'adresse IPv4 de source de l'en-tête de paquet IP le plus externe. "Le plus externe" se réfère ici au paquet IP lorsque il s'écoule sur le réseau, avant que tout en-tête en ait été effacé.
TYPES DE VALEURS ADMIS : PolicyIPv4AddrValue
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX

PROPRIÉTÉS (aucune)

6.12.2 Classe "PolicySourceIPv6Variable"

NOM PolicySourceIPv6Variable
DESCRIPTION Adresse IPv6 de source de l'en-tête de paquet IP le plus externe. "Le plus externe" se réfère ici au paquet IP lorsque il s'écoule sur le réseau, avant que tout en-tête en ait été effacé.
TYPES DE VALEURS ADMIS : - PolicyIPv6AddrValue
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.3 Classe "PolicyDestinationIPv4Variable"

NOM PolicyDestinationIPv4Variable
DESCRIPTION Adresse IPv4 de destination de l'en-tête de paquet IP le plus externe. "Le plus externe" se réfère ici au paquet IP lorsque il s'écoule sur le réseau, avant que tout en-tête en ait été effacé.
TYPES DE VALEURS ADMIS : - PolicyIPv4AddrValue
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.4 Classe "PolicyDestinationIPv6Variable"

NOM PolicyDestinationIPv6Variable
DESCRIPTION Adresse IPv6 de destination de l'en-tête de paquet IP le plus externe. "Le plus externe" se réfère ici au paquet IP lorsque il s'écoule sur le réseau, avant que tout en-tête en ait été effacé.
TYPES DE VALEURS ADMIS : - PolicyIPv6AddrValue
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.5 Classe "PolicySourcePortVariable"

NOM PolicySourcePortVariable
DESCRIPTION Les accès sont définis comme l'abstraction que les protocoles de transport utilisent pour distinguer entre plusieurs destinations au sein d'un certain ordinateur hôte. Pour les flux TCP et UDP, la PolicySourcePortVariable est logiquement liée au champ Accès de source de l'en-tête de paquet UDP ou TCP le plus externe. "Le plus externe" se réfère ici au paquet IP lorsque il s'écoule sur le réseau, avant que tout en-tête en ait été effacé.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..65535)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.6 Classe "PolicyDestinationPortVariable"

NOM PolicyDestinationPortVariable
DESCRIPTION Les accès sont définis comme l'abstraction que les protocoles de transport utilisent pour distinguer entre plusieurs destinations au sein d'un certain ordinateur hôte. Pour les flux TCP et UDP, la PolicyDestinationPortVariable est logiquement liée au champ Accès de destination de l'en-tête de paquet UDP ou TCP le plus externe. "Le plus externe" se réfère ici au paquet IP lorsque il s'écoule sur le réseau, avant que tout en-tête en ait été effacé.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..65535)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.7 Classe "PolicyIPProtocolVariable"

NOM PolicyIPProtocolVariable

DESCRIPTION C’est le numéro du protocole IP.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..255)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.8 Classe "PolicyIPVersionVariable"

NOM PolicyIPVersionVariable
DESCRIPTION C’est le numéro de version IP. Les valeurs bien connues sont 4 et 6.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..15)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.9 Classe "PolicyIPToSVariable"

NOM PolicyIPToSVariable
DESCRIPTION C’est l’octet IP TOS *(type de service)*.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..255) - PolicyBitStringValue (8 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.10 Classe "PolicyDSCPVariable"

NOM PolicyDSCPVariable
DESCRIPTION C’est le codet de 6 bits de service différenciés.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..63) - PolicyBitStringValue (6 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.11 Classe "PolicyFlowIdVariable"

NOM PolicyFlowIdVariable
DESCRIPTION C’est l’identifiant de flux de l’en-tête de paquet IPv6 le plus externe. "Le plus externe" se réfère ici au paquet IP lorsque il s’écoule sur le réseau, avant que tout en-tête en ait été effacé.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..1048575 - PolicyBitStringValue (20 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.12 Classe "PolicySourceMACVariable"

NOM PolicySourceMACVariable
DESCRIPTION C’est l’adresse MAC de source.
TYPES DE VALEURS ADMIS : - PolicyMACAddrValue
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.13 Classe "PolicyDestinationMACVariable"

NOM PolicyDestinationMACVariable
DESCRIPTION C’est l’adresse MAC de destination.
TYPES DE VALEURS ADMIS : - PolicyMACAddrValue
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.14 Classe "PolicyVLANVariable"

NOM PolicyVLANVariable
DESCRIPTION C'est l'identifiant de réseau virtuel ponté de zone locale, un champ de 12 bits comme défini dans la norme IEEE 802.1q.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..4095) - PolicyBitStringValue (12 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.15 Classe "PolicyCoSVariable"

NOM PolicyCoSVariable
DESCRIPTION C'est la classe de service, un champ de 3 bits utilisé dans l'en-tête de couche 2 pour choisir le traitement de transmission. Lié au champ de priorité d'utilisateur de IEEE 802.1q.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..7) - PolicyBitStringValue (3 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.16 Classe "PolicyEthertypeVariable"

NOM PolicyEthertypeVariable
DESCRIPTION C'est le numéro de protocole Ethertype des trames Ethernet.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..65535) - PolicyBitStringValue (16 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.17 Classe "PolicySourceSAPVariable"

NOM PolicySourceSAPVariable
DESCRIPTION Numéro de point d'accès de service de source de l'en-tête de LLC de IEEE 802.2.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..255) - PolicyBitStringValue (8 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.18 Classe "PolicyDestinationSAPVariable"

NOM PolicyDestinationSAPVariable
DESCRIPTION Numéro de point d'accès de service de destination de l'en-tête de LLC de IEEE 802.2.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..255) - PolicyBitStringValue (8 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.19 Classe "PolicySNAPOUIVariable"

NOM PolicySNAPOUIVariable
DESCRIPTION Valeur des trois premiers octets du champ Identifiant du protocole d'accès de sous réseau pour l'encapsulation 802.2, contenant un identifiant d'organisation unique (OUI). La valeur 00-00-00 indique l'encapsulation de trames Ethernet (RFC1042). La valeur de OUI de 00-00-F8 indique l'encapsulation spéciale de trames Ethernet par certains types de ponts (IEEE 802.1H). D'autres valeurs sont acceptées, mais ne sont pas définies ici. Ces valeurs de OUI sont à interpréter selon les conventions de notation de IEEE 802. Pour l'une et l'autre des encapsulations Ethernet, le reste du champ Identifiant de protocole est représenté par la PolicySNAPTypeVariable.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..16777215) - PolicyBitStringValue (24 bits)
DÉRIVÉE DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.20 Classe "PolicySNAPTypeVariable"

NOM PolicySNAPTypeVariable
DESCRIPTION La valeur des 4ème et 5ème octets du champ Identifiant du protocole d'accès de sous réseau (SNAP) pour l'encapsulation IEEE 802 lorsque la PolicySNAPOUIVariable indique un des deux formats de trame Ethernet encapsulée. Cette valeur est indéfinie pour les autres valeurs de PolicySNAPOUIVariable.
TYPES DE VALEURS ADMIS : - PolicyIntegerValue (0..65535) - PolicyBitStringValue (16 bits)
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

6.12.21 Classe "PolicyFlowDirectionVariable"

NOM PolicyFlowDirectionVariable
DESCRIPTION Direction d'un flux par rapport à un élément de réseau. La direction peut être "IN" et/ou "OUT".
TYPES DE VALEURS ADMIS : - PolicyStringValue ('IN", "OUT")
DÉRIVÉ DE PolicyImplicitVariable
ABSTRAITE FAUX
PROPRIÉTÉS (aucune)

Pour correspondre aux deux flux entrant et sortant, l'objet associé PolicyStringValue a deux entrées dans sa propriété StringList : "IN" et "OUT".

6.13 Classe abstraite "PolicyValue"

C'est une classe abstraite qui sert de classe de base pour toutes les sous classes qui sont utilisées pour définir des objets de valeur dans PCIMe. Elle est utilisée pour définir les valeurs et constantes utilisées dans les conditions de politique. La définition de la classe est la suivante :

NOM PolicyValue
DÉRIVÉ DE Policy
ABSTRAITE Vrai
PROPRIÉTÉS (aucune)

6.14 Sous classes de "PolicyValue" spécifiées dans PCIMe

Les paragraphes qui suivent contiennent les sous classes de PolicyValue définies dans PCIMe. Des sous classes supplémentaires peuvent être définies dans les modèles dérivés de PCIMe.

6.14.1 Classe "PolicyIPv4AddrValue"

Cette classe est utilisée pour fournir une liste des valeurs d'adresses IPv4, des noms d'hôtes et des gammes d'adresses à confronter à une condition de politique. La définition de la classe est la suivante :

NOM PolicyIPv4AddrValue
DÉRIVÉ DE PolicyValue
ABSTRAITE Faux
PROPRIÉTÉS IPv4AddrList[ ]

La propriété IPv4AddrList donne une liste non ordonnée de chaînes, dont chacune spécifie une seule adresse IPv4, un nom d'hôte, ou une gamme d'adresses IPv4, selon la définition ABNF [RFC2234] d'une adresse IPv4, comme spécifié ci-dessous :
IPv4address = 1*3CHIFFRE "." 1*3CHIFFRE "." 1*3CHIFFRE "." 1*3CHIFFRE
IPv4prefix = IPv4address "/" 1*2CHIFFRE
IPv4range = IPv4address"-"IPv4address
IPv4maskedaddress = IPv4address","IPv4address
Hostname (comme défini dans la [RFC1035])

Dans la définition ci-dessus, chaque entrée de chaîne est soit :
1. Une seule adresse IPv4 en notation séparée par des points, comme défini ci-dessus. Exemple : 121.1.1.2
2. Une gamme de préfixes d'adresses IPv4, comme défini ci-dessus, spécifiée par une adresse et une longueur de préfixe,

séparés par "/". Exemple : 2.3.128.0/15
3. Une gamme d'adresses IPv4range définie ci-dessus, spécifiée par une adresse de début en notation séparée par des points et une adresse de fin en notation séparée par des points, séparées par "-". La gamme inclut toutes les adresses entre les adresses de début et de fin de la gamme, incluant ces deux adresses. Exemple : 1.1.22.1-1.1.22.5
4. Une gamme d'adresses IPv4maskedaddress, comme défini ci-dessus, spécifiée par une adresse et un gabarit. L'adresse et le gabarit sont représentés en notation séparée par des points, et séparés par une virgule ",". L'adresse objet du gabarit apparaît avant la virgule, et le gabarit apparaît après la virgule. Exemple: 2.3.128.0,255.255.248.0.
5. Un seul nom d'hôte Hostname. Le format Hostname suit les lignes directrices et les restrictions spécifiées dans la [RFC1035]. Exemple : www.bigcompany.com.

Les conditions de confrontation des valeurs de IPv4AddrValues s'évaluent à vrai selon les règles génériques correspondantes. De plus, un nom d'hôte est confronté à une autre représentation de IPv4address valide par la résolution préalable du nom d'hôte en adresse IPv4, puis en comparant ensuite les adresses. Les confrontations de noms d'hôtes sont faites en utilisant une comparaison de chaîne des deux noms.

La définition de la propriété est la suivante :

NOM	IPv4AddrList
SYNTAXE	Chaîne
FORMAT	IPv4address | IPv4prefix | IPv4range | IPv4maskedaddress | hostname

6.14.2 Classe "PolicyIPv6AddrValue

Cette classe est utilisée pour définir une liste d'adresses IPv6, de noms d'hôtes, et de valeurs de gammes d'adresses. La définition de la classe est la suivante :

NOM	PolicyIPv6AddrValue
DÉRIVÉ DE	PolicyValue
ABSTRAITE	Faux
PROPRIÉTÉS	IPv6AddrList[ ]

La propriété IPv6AddrList fournit une liste non ordonnée de chaînes, dont chacune spécifie une adresse IPv6, un nom d'hôte, ou une gamme d'adresses IPv6. La définition du format d'adresse IPv6 utilise le format d'adresse standard défini dans la [RFC2373]. La définition ABNF [RFC2234] spécifiée dans la [RFC2373] est :
IPv6address = hexpart [":" IPv4address]
IPv4address = 1*3CHIFFRE "." 1*3CHIFFRE "." 1*3CHIFFRE "." 1*3CHIFFRE
IPv6prefix = hexpart "/" 1*2CHIFFRE
hexpart = hexseq | hexseq "::" [hexseq] | "::" [hexseq]
hexseq = hex4 *(":" hex4)
hex4 = 1*4HEXDIG
IPv6range = IPv6address"-"IPv6address
IPv6maskedaddress = IPv6address","IPv6address
Hostname (comme défini dans [RFC1035])

Chaque entrée de chaîne est soit :
1. Une seule adresse IPv6 comme défini ci-dessus.
2. Un seul nom d'hôte. Le format de Hostname suit les lignes directrices et les restrictions spécifiées dans la [RFC1035].
3. Une gamme d'adresses IPv6range, spécifiée par une adresse de début en notation séparée par des points et une adresse de fin en notation séparée par des points, séparées par "-". La gamme inclut toutes les adresses entre les adresses de début et de fin de la gamme, incluant ces deux adresses.
4. Une gamme d'adresses IPv4maskedaddress définie ci-dessus, spécifiée par une adresse et un gabarit. L'adresse et le gabarit sont représentés en notation séparée par des points et séparés par une virgule ",".
5. un seul IPv6prefix comme défini ci-dessus.

Les conditions de confrontation des valeurs de IPv6AddrValues s'évaluent à vrai selon les règles génériques correspondantes. De plus, un nom d'hôte est confronté à une autre représentation de IPv6address valide par la résolution préalable du nom d'hôte en adresse IPv6, puis en comparant ensuite les adresses. Les confrontations de noms d'hôtes sont faites en utilisant une comparaison de chaîne des deux noms.

6.14.3 Classe "PolicyMACAddrValue"

Cette classe est utilisée pour définir une liste d'adresses MAC et des valeurs de gamme d'adresse MAC. La définition de la

classe est la suivante :

NOM	PolicyMACAddrValue
DÉRIVÉ DE	PolicyValue
ABSTRAITE	Faux
PROPRIÉTÉS	MACAddrList[ ]

La propriété MACAddrList fournit une liste non ordonnée de chaînes, dont chacune spécifie une adresse MAC ou une gamme d'adresses MAC. Le format canonique d'adresse MAC 802 est utilisé. La définition en ABNF [RFC2234] est :

MACaddress = 1*4HEXDIG ":" 1*4HEXDIG ":" 1*4HEXDIG
MACmaskedaddress = MACaddress","MACaddress

Chaque entrée de chaîne est soit :
1. Une seule adresse MAC. Exemple: 0000:00A5:0000
2. Une gamme d'adresses MACmaskedaddress définie et spécifiée par une adresse et un gabarit. Le gabarit spécifie les bits pertinents dans l'adresse. Exemple : 0000:00A5:0000,FFFF:FFFF:0000 définit une gamme d'adresses MAC dans laquelle les quatre premiers octets sont égaux à 0000:00A5.

La définition de la propriété est la suivante :

NOM	MACAddrList
SYNTAXE	Chaîne
FORMAT	MACaddress | MACmaskedaddress

6.14.4 Classe "PolicyStringValue"

Cette classe est utilisée pour représenter une seule valeur de chaîne, ou un ensemble de valeurs de chaîne. Chaque valeur peut avoir des caractères génériques. La définition de la classe est la suivante :

NOM	PolicyStringValue
DÉRIVÉ DE	PolicyValue
ABSTRAITE	Faux
PROPRIÉTÉS	StringList[ ]

La propriété StringList fournit une liste non ordonnée de chaînes, représentant chacune une seule chaîne avec des caractères génériques. Le caractère astérisque "*" est utilisé comme caractère générique, et représente un remplacement de chaîne arbitraire. Par exemple, la valeur "abc*def" correspond à la chaîne "abcxyzdef", et la valeur "abc*def*" correspond à la chaîne "abcxxxdefyyyzzz". La définition de la syntaxe est identique à la syntaxe d'assertion de sous chaîne définie dans la [RFC2252]. Si le caractère astérisque est exigé au titre de la valeur de la chaîne elle-même, il DOIT être mis entre guillemets comme décrit au paragraphe 4.3 de la [RFC2252].

La définition de la propriété est la suivante :

NOM	StringList
SYNTAXE	Chaîne

6.14.5 Classe "PolicyBitStringValue"

Cette classe est utilisée pour représenter une seule valeur de chaîne binaire, ou un ensemble de valeurs de chaîne binaire. La définition de la classe est la suivante :

NOM	PolicyBitStringValue
DÉRIVÉ DE	PolicyValue
ABSTRAITE	Faux
PROPRIÉTÉS	BitStringList[ ]

La propriété BitStringList fournit une liste non ordonnée de chaînes, représentant chacune une seule chaîne binaire ou un ensemble de chaînes binaires. Le nombre de bits spécifié DEVRAIT être égal au nombre de bits de la variable attendue. Par exemple, pour une variable d'un octet, 8 bits devraient être spécifiés. Si la variable n'a pas une longueur fixée, la chaîne de bits devrait être confrontée à la chaîne binaire de plus fort poids de la variable. La définition formelle d'une chaîne binaire est :
binary-digit = "0" / "1"

bitString = 1*binary-digit
maskedBitString = bitString","bitString

Chaque entrée de chaîne est soit :
1. Une seule chaîne de bits. Exemple : 00111010
2. Une gamme de chaînes de bits spécifiée en utilisant une chaîne de bits et un gabarit de bits. Les champs de la chaîne de bits et du gabarit ont le nombre de bits spécifié. Le gabarit de chaîne binaire spécifie les bits significatifs dans la valeur de la chaîne de bits. Par exemple, 110110, 100110 et 110111 correspondrait à la chaîne maskedBitString 100110,101110 mais pas 100100.

La définition de la propriété est la suivante :

NOM	BitStringList
SYNTAXE	Chaîne
FORMAT	bitString | maskedBitString

6.14.6 Classe "PolicyIntegerValue"

Cette classe fournit une liste d'entiers et de valeurs de gamme d'entiers. Des entiers de taille arbitraire peuvent être représentés. La définition de la classe est la suivante :

NOM	PolicyIntegerValue
DÉRIVÉ DE	PolicyValue
ABSTRAITE	Faux
PROPRIÉTÉS	IntegerList[ ]

La propriété IntegerList fournit une liste non ordonnée d'entiers et de valeurs de gamme d'entier, représentées comme des chaînes. Le format de cette propriété prend une des formes suivantes :
1. Une valeur d'entier.
2. Une gamme d'entiers. La gamme est spécifiée par un entier de début et un entier de fin, séparés par '..'. L'entier de début DOIT être inférieur ou égal à l'entier de fin. La gamme inclut tous les entiers entre les entiers de début et de fin, incluant les deux entiers.

Pour représenter une gamme d'entiers, c'est-à-dire non bornée, les mots réservés -INFINITY et/ou INFINITY peuvent être utilisés à la place des entiers de début et de fin. En plus des confrontations ordinaires d'entiers, INFINITY correspond à INFINITY et -INFINITY correspond à -INFINITY.

La définition ABNF de la [RFC2234] est :

entier = [-]1*CHIFFRE | "INFINITY" | "-INFINITY"
integerrange = entier".."entier

Lorsque on utilise des gammes, les opérateurs 'supérieur à',' supérieur ou égal à', 'inférieur à', et 'inférieur ou égal à' peuvent être exprimés. Par exemple, "X est supérieur à 5" (où X est un entier) peut être traduit en "X correspond à 6-INFINITY". Cela permet de garder simple la sémantique de la condition de correspondance de l'opérateur pour la classe SimplePolicyCondition (c'est-à-dire, juste la valeur "match").

La définition de la propriété est la suivante :

NOM	IntegerList
SYNTAXE	Chaîne
FORMAT	entier | gamme-d'entiers

6.14.7 Classe "PolicyBooleanValue"

Cette classe est utilisée pour représenter une valeur booléenne (VRAI/FAUX). La définition de la classe est la suivante :

NOM	PolicyBooleanValue
DÉRIVÉ DE	PolicyValue
ABSTRAITE	Faux
PROPRIÉTÉS	BooleanValue

La définition de la propriété est la suivante :

NOM	BooleanValue
SYNTAXE	booléen

6.15 Classe "PolicyRoleCollection"

Cette classe représente une collection d'éléments gérés qui partagent un rôle commun. PolicyRoleCollection existe toujours dans le contexte d'un système, spécifié en utilisant l'association PolicyRoleCollectionInSystem. La valeur de la propriété PolicyRole dans cette classe spécifie le rôle, et peut être confrontée aux valeurs de la matrice PolicyRoles dans PolicyRules et PolicyGroups. Les éléments gérés qui partagent le rôle défini dans cette collection sont agrégés dans la collection via l'association ElementInPolicyRoleCollection.

NOM	PolicyRoleCollection
DESCRIPTION	Une sous classe de la classe CIM Collection utilisée pour grouper des éléments gérés qui partagent un rôle.
DÉRIVÉ DE	Collection
ABSTRAITE	FAUX
PROPRIÉTÉS	PolicyRole

6.15.1 Propriété à une seule valeur "PolicyRole"

Cette propriété représente le rôle associé à une PolicyRoleCollection. La définition de la propriété est la suivante :

NOM	PolicyRole
DESCRIPTION	Chaîne qui représente le rôle associé à une PolicyRoleCollection.
SYNTAXE	Chaîne

6.16 Classe "ReusablePolicyContainer"

La nouvelle classe ReusablePolicyContainer est définie comme suit :

NOM	ReusablePolicyContainer
DESCRIPTION	Classe qui représente un conteneur défini administrativement pour les informations réutilisables en rapport avec la politique. Cette classe n'introduit aucune propriété supplémentaire au delà de celles de sa super classe AdminDomain. Elle participe cependant à un certain nombre d'associations uniques.
DÉRIVÉ DE	AdminDomain
ABSTRAITE	FAUX
PROPRIÉTÉS	(aucune)

6.17 Classe "PolicyRepository" déconseillée de PCIM

La définition de la classe PolicyRepository (de PCIM) est mise à jour comme suit avec une indication que la classe a été déconseillée. Noter que lorsque un élément du modèle est déconseillé, son élément de remplacement est identifié explicitement.

NOM	PolicyRepository
DÉCONSEILLÉ POUR	ReusablePolicyContainer
DESCRIPTION	Classe qui représente un conteneur défini administrativement pour les informations réutilisables en rapport avec la politique. Cette classe n'introduit aucune propriété supplémentaire au delà de celles de sa super classe AdminDomain. Elle participe cependant à un certain nombre d'associations uniques.
DÉRIVÉ DE	AdminDomain
ABSTRAITE	FAUX
PROPRIÉTÉS	(aucune)

6.18 Classe abstraite "FilterEntryBase"

FilterEntryBase est la classe de base abstraite de laquelle sont dérivées toutes les classes d'entrée de filtre. Elle sert de point

terminal pour l'agrégation EntriesInFilterList, qui groupe les entrées de filtre en listes de filtres. Ses propriétés incluent des attributs de désignation CIM et une propriété booléenne IsNegated (pour transformer facilement en "NON" les informations de correspondance spécifiées dans une instance d'une de ses sous classes).

La définition de la classe est la suivante :

NOM	FilterEntryBase
DESCRIPTION	Classe abstraite qui représente un seul filtre, c'est-à-dire qui est agrégée dans une FilterList via l'agrégation EntriesInFilterList.
DÉRIVÉ DE	LogicalElement
TYPE	Abstrait
PROPRIÉTÉS	IsNegated

6.19 Classe "IpHeadersFilter"

Cette classe concrète contient les propriétés les plus couramment exigées pour effectuer le filtrage sur les en-têtes IP, TCP ou UDP. Les propriétés qui ne sont pas présentes dans une instance de IPHeadersFilter sont traitées comme 'toutes valeurs'. Une propriété HdrIpVersion identifie si les adresses IP dans une instance sont IPv4 ou IPv6. Comme les adresses de source et destination IP viennent du même en-tête de paquet, elles vont toujours être du même type.

La définition de la classe est la suivante :

NOM	IpHeadersFilter
DESCRIPTION	Classe qui représente un filtre entier d'en-tête IP, ou tout sous ensemble de celui-ci.
DÉRIVÉ DE	FilterEntryBase
TYPE	Concret
PROPRIÉTÉS	HdrIpVersion, HdrSrcAddress, HdrSrcAddressEndOfRange, HdrSrcMask, HdrDestAddress, HdrDestAddressEndOfRange, HdrDestMask, HdrProtocolID, HdrSrcPortStart, HdrSrcPortEnd, HdrDestPortStart, HdrDestPortEnd, HdrDSCP[ ], HdrFlowLabel

6.19.1 Propriété HdrIpVersion

Cette propriété est un entier non signé de 8 bits, qui identifie la version des adresses IP à filtrer. Les versions IP sont identifiées comme elles sont dans le champ Version de l'en-tête du paquet IP - IPv4 = 4, IPv6 = 6. Ces deux valeurs sont les seules définies pour cette propriété.

La valeur de cette propriété détermine la taille des chaînes d'octet dans les six propriétés HdrSrcAddress, HdrSrcAddressEndOfRange, HdrSrcMask, HdrDestAddress, HdrDestAddressEndOfRange, et HdrDestMask, comme suit :

- IPv4 : OctetString(SIZE (4))
- IPv6 : OctetString(SIZE (16|20)), selon la présence d'un identifiant de portée

Si on ne fournit pas de valeur pour cette propriété, le filtre ne prend pas en considération la version IP pour le choix des paquets satisfaisants, c'est-à-dire que version IP satisfait pour toutes les valeurs. Dans ce cas, HdrSrcAddress, HdrSrcAddressEndOfRange, HdrSrcMask, HdrDestAddress, HdrDestAddressEndOfRange, et HdrDestMask doivent aussi ne pas être présents.

6.19.2 Propriété HdrSrcAddress

Cette propriété est une chaîne d'octets, d'une taille déterminée par la valeur de la propriété HdrIpVersion, représentant une adresse IP de source. Lorsque il n'y a pas de valeur HdrSrcAddressEndOfRange, cette valeur est comparée à l'adresse de source dans l'en-tête IP, selon le gabarit représenté dans la propriété HdrSrcMask. (Noter que le gabarit est traité par l'opérateur logique ET avec l'adresse.) Lorsque il y a une valeur de HdrSrcAddressEndOfRange, celle-ci est le début de la gamme spécifiée (c'est-à-dire que HdrSrcAddress est inférieur à HdrSrcAddressEndOfRange) c'est-à-dire comparée à l'adresse de source dans l'en-tête IP et satisfait toute valeur de la gamme.

Si une valeur de cette propriété n'est pas fournie, le filtre ne prend alors pas en considération HdrSrcAddress pour le choix des paquets correspondants, c'est-à-dire, HdrSrcAddress correspond pour toutes les valeurs.

6.19.3 Propriété HdrSrcAddressEndOfRange

Cette propriété est une chaîne d'octets, d'une taille déterminée par la valeur de la propriété HdrIpVersion, représentant la fin d'une gamme d'adresses de source IP (inclusive), où le début de la gamme est la valeur de la propriété HdrSrcAddress.

Si on n'a pas fourni de valeur pour HdrSrcAddress, cette propriété NE DOIT PAS être fournie non plus. Si une valeur est fournie pour cette propriété, alors HdrSrcMask NE DOIT PAS être fourni.

6.19.4 Propriété HdrSrcMask

Cette propriété est une chaîne d'octets, d'une taille déterminée par la valeur de la propriété HdrIpVersion, représentant un gabarit à utiliser pour comparer l'adresse de source dans l'en-tête IP avec la valeur représentée dans la propriété HdrSrcAddress.

Si on n'a pas fourni de valeur pour cette propriété, le filtre ne prend alors pas en considération HdrSrcMask pour sélectionner les paquets qui correspondent, c'est-à-dire que la valeur de HdrSrcAddress ola gamme d'adresse de source doit correspondre exactement à l'adresse de source dans le paquet. Si une valeur est fournie pour cette propriété, HdrSrcAddressEndOfRange NE DOIT PAS être fourni.

6.19.5 Propriété HdrDestAddress

Cette propriété est une chaîne d'octets, d'une taille déterminée par la valeur de la propriété HdrIpVersion, représentant une adresse de destination IP. Lorsque il n'y a pas de valeur de HdrDestAddressEndOfRange, cette valeur est comparée à l'adresse de destination dans l'en-tête IP, selon le gabarit représenté dans la propriété HdrDestMask. (Noter que le gabarit et ajouté à l'adresse par l'opérateur logique ET.) Lorsque il y a une valeur de HdrDestAddressEndOfRange, cette valeur est le début de la gamme spécifiée (c'est-à-dire, la HdrDestAddress est inférieure à la HdrDestAddressEndOfRange) c'est-à-dire comparée à l'adresse de destination dans l'en-tête IP et correspond à toute valeur dans la gamme.

Si on n'a pas fourni de valeur pour cette propriété, le filtre ne prend alors pas en considération HdrDestAddress pour la sélection des paquets qui correspondent, c'est-à-dire, HdrDestAddress correspond pour toutes les valeurs.

6.19.6 Propriété HdrDestAddressEndOfRange

Cette propriété est une chaîne d'octets, d'une taille déterminée par la valeur de la propriété HdrIpVersion, représentant la fin (incluse) d'une gamme d'adresses IP de destination, où le début de la gamme est la valeur de la propriété HdrDestAddress.

Si une valeur n'est pas fournie pour HdrDestAddress, alors cette propriété NE DOIT PAS non plus être fournie. Si une valeur est fournie pour cette propriété, alors HdrDestMask NE DOIT PAS être fourni.

6.19.7 Propriété HdrDestMask

Cette propriété est une chaîne d'octets, d'une taille déterminée par la valeur de la propriété HdrIpVersion, représentant un gabarit à utiliser pour comparer l'adresse de destination dans l'en-tête IP avec la valeur représentée dans la propriété HdrDestAddress.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération HdrDestMask pour la sélection des paquets qui correspondent, c'est-à-dire, la valeur de HdrDestAddress ou de la gamme d'adresses de destination doit correspondre exactement à l'adresse de destination dans le paquet. Si une valeur est fournie pour cette propriété, alors HdrDestAddressEndOfRange NE DOIT PAS être fourni.

6.19.8 Propriété HdrProtocolID

Cette propriété est un entier non signé de 8 bits, représentant un type de protocole IP. Cette valeur est comparée au champ Protocole dans l'en-tête IP.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération HdrProtocolID pour le choix des paquets qui correspondent, c'est-à-dire, HdrProtocolID correspond pour toutes les valeurs.

6.19.9 Propriété HdrSrcPortStart

Cette propriété est un entier non signé de 16 bits, représentant l'extrémité inférieure d'une gamme d'accès UDP ou TCP de source. L'extrémité supérieure de la gamme est représentée par la propriété HdrSrcPortEnd. La valeur de HdrSrcPortStart NE

DOIT PAS être supérieure à la valeur de HdrSrcPortEnd. Un seul accès est indiqué par des valeurs égales pour HdrSrcPortStart et HdrSrcPortEnd.

Un filtre d'accès de source est évalué en vérifiant si l'accès de source identifié dans l'en-tête IP tombe dans la gamme de valeurs entre HdrSrcPortStart et HdrSrcPortEnd, incluant ces deux points d'extrémité.

Si on n'a pas fourni de valeur pour cette propriété, le filtre ne prend alors pas en considération HdrSrcPortStart pour sélectionner les paquets qui correspondent, c'est-à-dire, il n'y a pas de limite inférieure dans les valeurs d'accès de source correspondantes.

6.19.10 Propriété HdrSrcPortEnd

Cette propriété est un entier non signé de 16 bits, représentant la limite supérieure d'une gamme d'accès de source UDP ou TCP. L'extrémité inférieure de la gamme est représentée par la propriété HdrSrcPortStart. La valeur de HdrSrcPortEnd NE DOIT PAS être inférieure à la valeur de HdrSrcPortStart. Un seul accès est indiqué par des valeurs égales pour HdrSrcPortStart et HdrSrcPortEnd.

Un filtre d'accès de source est évalué en vérifiant si l'accès de source identifié dans l'en-tête IP tombe dans la gamme de valeurs entre HdrSrcPortStart et HdrSrcPortEnd, incluant ces deux points d'extrémité.

Si on n'a pas fourni de valeur pour cette propriété, le filtre ne prend alors pas en considération HdrSrcPortEnd pour sélectionner les paquets qui correspondent, c'est-à-dire, il n'y a pas de limite supérieure dans les valeurs d'accès de source.

6.19.11 Propriété HdrDestPortStart

Cette propriété est un entier non signé de 16 bits, représentant l'extrémité inférieure d'une gamme d'accès de destination UDP ou TCP. La limite supérieure de la gamme est représentée par la propriété HdrDestPortEnd. La valeur de HdrDestPortStart NE DOIT PAS être supérieure à la valeur de HdrDestPortEnd. Un seul accès est indiqué par des valeurs égales pour HdrDestPortStart et HdrDestPortEnd.

Un filtre d'accès de destination est évalué en vérifiant si l'accès de destination identifié dans l'en-tête IP tombe dans la gamme des valeurs entre HdrDestPortStart et HdrDestPortEnd, incluant ces deux points d'extrémité.

Si on n'a pas fourni de valeur pour cette propriété, le filtre ne prend alors pas en considération HdrDestPortStart pour sélectionner les paquets qui correspondent, c'est-à-dire, il n'y a pas de limite inférieure dans les valeurs d'accès de destination.

6.19.12 Propriété HdrDestPortEnd

Cette propriété est un entier non signé de 16 bits, représentant l'extrémité supérieure d'une gamme d'accès de destination UDP ou TCP. L'extrémité inférieure de la gamme est représentée par la propriété HdrDestPortStart. La valeur de HdrDestPortEnd NE DOIT PAS être inférieure à la valeur de HdrDestPortStart. Un seul accès est indiqué par des valeurs égales pour HdrDestPortStart et HdrDestPortEnd.

Un filtre d'accès de destination est évalué en vérifiant si l'accès de destination identifié dans l'en-tête IP tombe dans la gamme des valeurs entre HdrDestPortStart et HdrDestPortEnd, incluant ces deux points d'extrémité.

Si on n'a pas fourni de valeur pour cette propriété, le filtre ne prend alors pas en considération HdrDestPortEnd pour sélectionner les paquets qui correspondent, c'est-à-dire, il n'y a pas de limite supérieure dans la confrontation des valeurs des accès de destination.

6.19.13 Propriété HdrDSCP

La propriété HdrDSCP est définie comme une matrice de uint8, restreinte à la gamme 0..63. Comme les DSCP sont définis comme des codets discrets, sans structure inhérente, il n'y a pas de relation sémantiquement signifiante entre les différents DSCP. Par conséquent, il n'y a aucune disposition pour spécifier une gamme de DSCP dans cette propriété. Cependant, une liste de DSCP individuels, qui sont traités avec l'opérateur logique OU pour former un filtre, est prise en charge par la syntaxe de la matrice.

Si on n'a pas fourni de valeur pour cette propriété, le filtre ne prend alors pas en considération HdrDSCP pour la sélection des paquets qui correspondent, c'est-à-dire, HdrDSCP correspond pour toutes les valeurs.

6.19.14 Propriété HdrFlowLabel

Le champ Étiquette de flux de 20 bits dans l'en-tête IPv6 peut être utilisé par une source pour étiqueter des séquences de paquets pour lesquelles il demande un traitement particulier de la part des appareils IPv6, comme une qualité de service qui ne soit pas celle par défaut, ou un service en 'temps réel'. Cette propriété est une chaîne d'octets de taille 3 (c'est-à-dire, 24 bits), dans laquelle l'étiquette de flux de 20 bits apparaît dans les 20 bits de droite, bourrée à gauche avec b'0000'.

Si on n'a pas fourni de valeur pour cette propriété, le filtre ne prend alors pas en considération HdrFlowLabel pour la sélection des paquets qui correspondent, c'est-à-dire, HdrFlowLabel correspond pour toutes les valeur.

6.20 Classe "8021Filter"

Cette classe concrète permet d'exprimer dans un seul objet les adresses MAC 802.1 de source et de destination, ainsi que l'identifiant de protocole 802.1, la priorité, et les champs d'identifiant de VLAN.

La définition de la classe est la suivante :

NOM	8021Filter
DESCRIPTION	Classe qui permet d'exprimer dans un seul objet l'adresse MAC 802.1 de source et de destination, l'identifiant de protocole 802.1, la priorité, et les champs d'identifiant de VLAN.
DÉRIVÉ DE	FilterEntryBase
TYPE	Concret
PROPRIÉTÉS	8021HdrSrcMACAddr, 8021HdrSrcMACMask, 8021HdrDestMACAddr, 8021HdrDestMACMask, 8021HdrProtocolID, 8021HdrPriorityValue, 8021HDRVLANID

6.20.1 Propriété 8021HdrSrcMACAddr

Cette propriété est une chaîne d'octets de taille 6, représentant une adresse MAC de source de 48 bits en format canonique. Cette valeur est comparée au champ Adresse de source dans l'en-tête MAC, selon le gabarit représenté dans la propriété 8021HdrSrcMACMask.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération 8021HdrSrcMACAddr pour sélectionner les paquets qui correspondent, c'est-à-dire que 8021HdrSrcMACAddr correspond pour toutes les valeurs.

6.20.2 Propriété 8021HdrSrcMACMask

Cette propriété est une chaîne d'octets de taille 6, représentant un gabarit de 48 bits à utiliser pour comparer le champ Adresse de source dans l'en-tête MAC à la valeur représentée dans la propriété 8021HdrSrcMACAddr.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération 8021HdrSrcMACMask pour sélectionner les paquets qui correspondent, c'est-à-dire que la valeur de 8021HdrSrcMACAddr doit correspondre exactement à l'adresse MAC de source dans le paquet.

6.20.3 Propriété 8021HdrDestMACAddr

Cette propriété est une chaîne d'octets de taille 6, représentant une adresse MAC de destination de 48 bits en format canonique. Cette valeur est comparée au champ Adresse de destination dans l'en-tête MAC, selon le gabarit représenté dans la propriété 8021HdrDestMACMask.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération 8021HdrDestMACAddr pour sélectionner les paquets qui correspondent, c'est-à-dire, 8021HdrDestMACAddr correspond pour toutes les valeurs.

6.20.4 Propriété 8021HdrDestMACMask

Cette propriété est une chaîne d'octets de taille 6, représentant un gabarit de 48 bits à utiliser pour comparer le champ Adresse de destination dans l'en-tête MAC avec la valeur représentée dans la propriété 8021HdrDestMACAddr.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération 8021HdrDestMACMask pour sélectionner les paquets qui correspondent, c'est-à-dire que la valeur de 8021HdrDestMACAddr doit correspondre exactement à l'adresse MAC de destination dans le paquet.

6.20.5 Propriété 8021HdrProtocolID

Cette propriété est un entier non signé de 16 bits, représentant un type de protocole Ethernet. Cette valeur est comparée au champ Type Ethernet dans l'en-tête MAC 802.3.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération 8021HdrProtocolID pour sélectionner les paquets qui correspondent, c'est-à-dire, 8021HdrProtocolID correspond pour toutes les valeurs.

6.20.6 Propriété 8021HdrPriorityValue

Cette propriété est un entier non signé de 8 bits, représentant une priorité 802.1Q. Cette valeur est comparée au champ Priority dans l'en-tête 802.1Q. Comme le champ Priority 802.1Q consiste en 3 bits, les valeurs pour cette propriété sont limitées à la gamme 0..7.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération 8021HdrPriorityValue pour sélectionner les paquets qui correspondent, c'est-à-dire, 8021HdrPriorityValue correspond pour toutes les valeurs.

6.20.7 Propriété 8021HdrVLANID

Cette propriété est un entier non signé de 32 bits, représentant un identifiant de VLAN 802.1Q. Cette valeur est comparée au champ VLAN ID dans l'en-tête 802.1Q. Comme le champ 802.1Q VLAN ID consiste en 12 bits, les valeurs pour cette propriété sont limitées à la gamme 0..4095.

Si on n'a pas fourni de valeur pour cette propriété, alors le filtre ne prend pas en considération 8021HdrVLANID pour sélectionner les paquets qui correspondent, c'est-à-dire, 8021HdrVLANID correspond pour toutes les valeurs.

6.21 Classe FilterList

C'est une classe concrète qui agrège des instances de (sous classes de) FilterEntryBase via l'agrégation EntriesInFilterList. Il est possible d'agréger différents types de filtres dans une seule FilterList - par exemple, des filtres d'en-tête de paquet (représentés par la classe IpHeadersFilter) et des filtres de sécurité (représentés par des sous classes de FilterEntryBase définies par IPsec).

La propriété d'agrégation EntriesInFilterList.EntrySequence est toujours réglée à 0, pour indiquer que les entrées de filtre agrégées sont ajoutées ensemble par l'opérateur logique ET pour former un sélecteur pour une classe de trafic.

La définition de la classe est la suivante :

NOM	FilterList
DESCRIPTION	Classe concrète qui représente l'agrégation de plusieurs filtres.
DÉRIVÉ DE	LogicalElement
TYPE	Concret
PROPRIÉTÉS	Direction

6.21.1 Propriété Direction

Cette propriété est une énumération d'entiers non signés de 16 bits, représentant la direction du flux de trafic auquel la FilterList est à appliquer. Les valeurs d'énumération définies sont :

- NonApplicable(0)
- Entrée(1)
- Sortie(2)
- LesDeux(3) - cette valeur est utilisée pour indiquer que la direction est immatérielle, par exemple, pour filtrer sur un sous réseau de source sans considérer si le flux est entrant ou sortant.
- Miroir(4) - Cette valeur est aussi applicable aux deux traitements de flux entrant et sortant, mais elle indique que le critère de filtre est appliqué de façon asymétrique au trafic dans les deux directions et donc, spécifie l'inverse des critères de source et de destination (par opposition à l'égalité de ces critères telle qu'indiqué par "LesDeux"). Les conditions de la confrontation dans les instances de sous classe agrégée FilterEntryBase sont définies dans la perspective des flux sortants et appliquées aux flux entrants ainsi qu'en inversant les critères de source et de destination. Ainsi, par exemple, si on considère une FilterList avec 3 entrées de filtre qui indiquent l'accès de destination = 80, et respectivement les adresses de source et de destination de a et b. Alors, pour la direction sortante, les entrées de filtre correspondent comme spécifié et le 'miroir' (pour la direction entrante) correspond à l'accès de source = 80 et aux adresses de source et de destination

respectivement de a et b.

7. Définitions d'association et d'agrégation

Les définitions qui suivent s'ajoutent à celles de PCIM lui-même. les définitions de PCIM qui ne sont pas DÉCONSEILLÉES font toujours partie du modèle global d'informations de cœur de politique.

7.1 Agrégation "PolicySetComponent"

PolicySetComponent est une nouvelle classe d'agrégation qui collecte les instances des sous classes de PolicySet (PolicyGroups et PolicyRules) dans des ensembles cohérents de politiques.

NOM	PolicySetComponent
DESCRIPTION	Classe concrète qui représente les composants d'un ensemble de politiques qui ont la même stratégie de décision, et sont rangés par priorité au sein de l'ensemble.
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref PolicySet[0..n]] ; PartComponent[ref PolicySet[0..n]] ; Priority

La définition de la propriété Priority est inchangée par rapport à sa précédente définition dans [PCIM].

NOM	Priority
DESCRIPTION	Un entier non négatif pour donner une priorité à ce composant de PolicySet par rapport aux autres composants du même PolicySet. Une plus grande valeur indique une plus forte priorité.
SYNTAXE	uint16
PAR DÉFAUT	0

7.2 Agrégation "PolicyGroupInPolicyGroup" déconseillée de PCIM

La nouvelle agrégation PolicySetComponent est utilisée directement pour représenter l'agrégation des groupes de politique par un groupe de politiques de niveau supérieur. Donc, l'agrégation PolicyGroupInPolicyGroup n'est plus nécessaire, et peut être déconseillée.

NOM	PolicyGroupInPolicyGroup
DÉCONSEILLÉ POUR	PolicySetComponent
DESCRIPTION	Une classe qui représente l'agrégation de PolicyGroups par un PolicyGroup de niveau supérieur.
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref PolicyGroup[0..n]] ; PartComponent[ref PolicyGroup[0..n]]

7.3 Agrégation "PolicyRuleInPolicyGroup" déconseillée de PCIM

La nouvelle agrégation PolicySetComponent est utilisée directement pour représenter l'agrégation de règles de politique par un groupe de politiques. Donc, l'agrégation PolicyRuleInPolicyGroup n'est plus nécessaire, et peut être déconseillée.

NOM	PolicyRuleInPolicyGroup
DÉCONSEILLÉ POUR	PolicySetComponent
DESCRIPTION	Une classe qui représente l'agrégation de PolicyRules par un PolicyGroup.
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref PolicyGroup[0..n]] ; PartComponent[ref PolicyRule[0..n]]

7.4 Association abstraite "PolicySetInSystem"

PolicySetInSystem est une nouvelle association qui définit une relation entre un système et un ensemble de politiques utilisé dans la portée administrative de ce système (par exemple, AdminDomain, ComputerSystem). La propriété Priority est utilisée

pour allouer une priorité relative à un PolicySet dans la portée administrative dans les contextes où ce n'est pas un composant d'un autre PolicySet.

NOM	PolicySetInSystem
DESCRIPTION	Une classe abstraite qui représente la relation entre un System et un PolicySet, c'est-à-dire utilisé dans la portée administrative du System.
DÉRIVÉ DE	PolicyInSystem
ABSTRAITE	VRAI
PROPRIÉTÉS	Antecedent[ref System[0..1]] ; Dependent [ref PolicySet[0..n]] ; Priority

La propriété Priority est utilisée pour spécifier la priorité relative du PolicySet référencé lorsque il y a plus d'une instance de PolicySet qui est appliquée à une ressource gérée qui n'est pas un PolicySetComponent et, donc n'a pas d'autre priorité relative définie.

NOM	Priority
DESCRIPTION	Entier non négatif pour allouer une priorité au PolicySet référencé parmi les autres instances de PolicySet qui ne sont pas des composants d'un PolicySet commun. Une plus grande valeur indique une plus forte priorité.
SYNTAXE	uint16
PAR DÉFAUT	0

7.5 Mise à jour de l'association faible "PolicyGroupInSystem" de PCIM

Sans considérer si il est un composant d'un autre PolicySet, un PolicyGroup est lui-même défini au sein de la portée d'un système. Cette association relie un PolicyGroup au système dans la portée duquel le PolicyGroup est défini. C'est une sous classe de l'association abstraite PolicySetInSystem. La définition de classe de l'association est la suivante :

NOM	PolicyGroupInSystem
DESCRIPTION	Une classe qui représente le fait qu'un PolicyGroup est défini dans la portée d'un System.
DÉRIVÉ DE	PolicySetInSystem
ABSTRAITE	FAUX
PROPRIÉTÉS	Antecedent[ref System[1..1]] ; Dependent [ref PolicyGroup[weak]]

La référence "Antecedent" est héritée de PolicySetInSystem, et écrasée pour restreindre sa cardinalité à [1..1]. Elle sert de référence d'objet à un système qui fournit une portée pour un ou plusieurs PolicyGroups. Comme c'est une association faible, la cardinalité pour cette référence d'objet est toujours 1, c'est-à-dire qu'un PolicyGroup est toujours défini au sein de la portée de exactement un System.

La référence "Dependent" est héritée de PolicySetInSystem, et écrasée pour devenir une référence d'objet à un PolicyGroup défini dans la portée d'un System. Noter que pour une seule instance de la classe d'association PolicyGroupInSystem, cette propriété (comme toutes les propriétés de référence) est à une seule valeur. La cardinalité [0..n] indique qu'un certain système peut avoir 0, 1, ou plus d'un PolicyGroups définis dans sa portée.

7.6 Mise à jour de l'association faible "PolicyRuleInSystem" de PCIM

Sans considérer si il est un composant d'un autre PolicySet, une PolicyRule est elle-même définie dans la portée d'un système. Cette association relie une PolicyRule au système dans la portée duquel la PolicyRule est définie. C'est une sous classe de l'association abstraite PolicySetInSystem. La définition de classe pour l'association est la suivante :

NOM	PolicyRuleInSystem
DESCRIPTION	Classe qui représente le fait qu'une PolicyRule est définie dans la portée d'un System.
DÉRIVÉ DE	PolicySetInSystem
ABSTRAITE	FAUX
PROPRIÉTÉS	Antecedent[ref System[1..1]] ; Dependent[ref PolicyRule[weak]]

La référence "Antecedent" est héritée de PolicySetInSystem, et écrasée pour restreindre sa cardinalité à [1..1]. Elle sert de référence d'objet à un système qui fournit une portée pour une ou plusieurs PolicyRules. Comme c'est une association faible, la cardinalité pour cette référence d'objet est toujours 1, c'est-à-dire qu'une PolicyRule est toujours définie au sein de la portée de exactement un System.

La référence "Dependent" est héritée de PolicySetInSystem, et écrasée pour devenir une référence d'objet à une PolicyRule

définie dans la portée d'un System. Noter que pour une seule instance de la classe d'association PolicyRuleInSystem, cette propriété (comme toutes les propriétés de référence) est à une seule valeur. La cardinalité [0..n] indique qu'un certain système peut avoir 0, 1, ou plus d'une PolicyRules définies dans sa portée.

7.7 Agrégation abstraite "PolicyConditionStructure"

NOM	PolicyConditionStructure
DESCRIPTION	Une classe qui représente l'agrégation de PolicyConditions par une instance agrégeante.
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	VRAI
PROPRIÉTÉS	PartComponent[ref PolicyCondition[0..n]] ; GroupNumber ; ConditionNegated

7.8 Mise à jour de l'agrégation "PolicyConditionInPolicyRule" de PCIM

L'agrégation PCIM "PolicyConditionInPolicyRule" est mise à jour pour en faire une sous classe de la nouvelle agrégation abstraite PolicyConditionStructure. Les propriétés GroupNumber et ConditionNegated sont maintenant héritées, plutôt que spécifiées explicitement comme elles l'étaient dans PCIM.

NOM	PolicyConditionInPolicyRule
DESCRIPTION	Une classe qui représente l'agrégation de PolicyConditions par une PolicyRule.
DÉRIVÉ DE	PolicyConditionStructure
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref PolicyRule[0..n]]

7.9 Agrégation "PolicyConditionInPolicyCondition"

Une seconde sous classe de PolicyConditionStructure est définie, qui représente l'arrangement de conditions de politique en une condition de politique de niveau supérieur.

NOM	PolicyConditionInPolicyCondition
DESCRIPTION	Une classe qui représente l'agrégation de PolicyConditions par une autre PolicyCondition.
DÉRIVÉ DE	PolicyConditionStructure
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref CompoundPolicyCondition[0..n]]

7.10 Agrégation abstraite "PolicyActionStructure"

NOM	PolicyActionStructure
DESCRIPTION	Une classe qui représente l'agrégation de PolicyActions par une instance agrégeante.
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	VRAI
PROPRIÉTÉS	PartComponent[ref PolicyAction[0..n]] ; ActionOrder

La définition de la propriété ActionOrder apparaît au paragraphe 7.8.3 de PCIM [RFC3060].

7.11 Mise à jour de l'agrégation "PolicyActionInPolicyRule" de PCIM

L'agrégation PCIM "PolicyActionInPolicyRule" est mise à jour pour en faire une sous classe de la nouvelle agrégation abstraite PolicyActionStructure. La propriété ActionOrder est maintenant héritée, plutôt que spécifiée explicitement comme elle l'était dans PCIM.

NOM	PolicyActionInPolicyRule
DESCRIPTION	Une classe qui représente l'agrégation de PolicyActions par une PolicyRule.
DÉRIVÉ DE	PolicyActionStructure
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref PolicyRule[0..n]]

7.12 Agrégation "PolicyActionInPolicyAction"

Une seconde sous classe de PolicyActionStructure est définie, représentant l'arrangement d'actions de politique en une action de politique de niveau supérieur.

NOM	PolicyActionInPolicyAction
DESCRIPTION	Une classe qui représente l'agrégation de PolicyActions par une autre PolicyAction.
DÉRIVÉ DE	PolicyActionStructure
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref CompoundPolicyAction[0..n]]

7.13 Agrégation "PolicyVariableInSimplePolicyCondition"

Une condition de politique simple est représentée comme un triplet ordonné {variable, opérateur, valeur}. Cette agrégation fournit le lien entre une instance de SimplePolicyCondition et une seule PolicyVariable. L'agrégation PolicyValueInSimplePolicyCondition relie la SimplePolicyCondition à une seule PolicyValue. La propriété Operator de SimplePolicyCondition représente le troisième élément du triplet, l'opérateur.

La définition de la classe pour cette agrégation est la suivante :

NOM	PolicyVariableInSimplePolicyCondition
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	Faux
PROPRIÉTÉS	GroupComponent[ref SimplePolicyCondition[0..n]] ; PartComponent[ref PolicyVariable[1..1]]

La propriété de référence "GroupComponent" est héritée de PolicyComponent, et écrasée pour devenir une référence d'objet à une SimplePolicyCondition qui contient exactement une PolicyVariable. Noter que pour toute instance seule de la classe d'agrégation PolicyVariableInSimplePolicyCondition, cette propriété est à une seule valeur. La cardinalité [0..n] indique qu'il peut y avoir 0, 1, ou plusieurs objets SimplePolicyCondition qui contiennent un certain objet de variable de politique.

La propriété de référence "PartComponent" est héritée de PolicyComponent, et écrasée pour devenir une référence d'objet à une PolicyVariable, c'est-à-dire définie dans la portée d'une SimplePolicyCondition. Noter que pour toute instance seule de la classe d'association PolicyVariableInSimplePolicyCondition, cette propriété (comme toutes les propriétés de référence) est à une seule valeur. La cardinalité [1..1] indique qu'une SimplePolicyCondition doit avoir exactement une variable de politique définie dans sa portée afin d'avoir un sens.

7.14 Agrégation "PolicyValueInSimplePolicyCondition"

Une condition de politique simple est représentée comme un triplet ordonné {variable, opérateur, valeur}. Cette agrégation fournit le lien entre une instance de SimplePolicyCondition et une seule PolicyValue. L'agrégation PolicyVariableInSimplePolicyCondition relie la SimplePolicyCondition à une seule PolicyVariable. La propriété Operator de SimplePolicyCondition représente le troisième élément du triplet, l'opérateur.

La définition de classe pour cette agrégation est la suivante :

NOM	PolicyValueInSimplePolicyCondition
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	Faux
PROPRIÉTÉS	GroupComponent[ref SimplePolicyCondition[0..n]] ; PartComponent[ref PolicyValue[1..1]]

La propriété de référence "GroupComponent" est héritée de PolicyComponent, et écrasée pour devenir une référence d'objet à une SimplePolicyCondition qui contient exactement une PolicyValue. Noter que pour toute instance seule de la classe d'agrégation PolicyValueInSimplePolicyCondition, cette propriété est à une seule valeur. La cardinalité [0..n] indique qu'il peut y avoir 0, 1, ou plusieurs objets SimplePolicyCondition qui contiennent un certain objet de valeur de politique.

La propriété de référence "PartComponent" est héritée de PolicyComponent, et écrasée pour devenir une référence d'objet à une PolicyValue, c'est-à-dire définie dans la portée d'une SimplePolicyCondition. Noter que pour toute instance seule de la classe d'association PolicyValueInSimplePolicyCondition, cette propriété (comme toutes les propriétés de référence) est à une seule valeur. La cardinalité [1..1] indique d'une SimplePolicyCondition doit avoir exactement une valeur de politique définie dans sa portée pour avoir un sens.

7.15 Agrégation "PolicyVariableInSimplePolicyAction"

Une action de politique simple est représentée comme une paire {variable, valeur}. Cette agrégation fournit le lien entre une instance de SimplePolicyAction et une seule PolicyVariable. L'agrégation PolicyValueInSimplePolicyAction relie la SimplePolicyAction à une seule PolicyValue.

La définition de classe pour cette agrégation est la suivante :

NOM	PolicyVariableInSimplePolicyAction
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	Faux
PROPRIÉTÉS	GroupComponent[ref SimplePolicyAction[0..n]] ; PartComponent[ref PolicyVariable[1..1]]

La propriété de référence "GroupComponent" est héritée de PolicyComponent, et écrasée pour devenir une référence d'objet à une SimplePolicyAction qui contient exactement une PolicyVariable. Noter que pour toute instance seule de la classe d'agrégation PolicyVariableInSimplePolicyAction, cette propriété est à une seule valeur. La cardinalité [0..n] indique qu'il peut y avoir 0, 1, ou plusieurs objets SimplePolicyAction qui contiennent n'importe quel objet de variable de politique.

La propriété de référence "PartComponent" est héritée de PolicyComponent, et écrasée pour devenir une référence d'objet à une PolicyVariable, c'est-à-dire définie dans la portée d'une SimplePolicyAction. Noter que pour toute instance seule de la classe d'association PolicyVariableInSimplePolicyAction, cette propriété (comme toutes les propriétés de référence) est à une seule valeur. La cardinalité [1..1] indique qu'une SimplePolicyAction doit avoir exactement une variable de politique définie dans sa portée afin d'avoir un sens.

7.16 Agrégation "PolicyValueInSimplePolicyAction"

Une action de politique simple est représentée comme une paire {variable, valeur}. Cette agrégation fournit le lien entre une instance de SimplePolicyAction et une seule PolicyValue. L'agrégation PolicyVariableInSimplePolicyAction relie la SimplePolicyAction à une seule PolicyVariable.

La définition de classe pour cette agrégation est la suivante :

NOM	PolicyValueInSimplePolicyAction
DÉRIVÉ DE	PolicyComponent
ABSTRAITE	Faux
PROPRIÉTÉS	GroupComponent[ref SimplePolicyAction[0..n]] ; PartComponent[ref PolicyValue[1..1]]

La propriété de référence "GroupComponent" est héritée de PolicyComponent, et écrasée pour devenir une référence d'objet à une SimplePolicyAction qui contient exactement une PolicyValue. Noter que pour toute instance seule de la classe d'agrégation PolicyValueInSimplePolicyAction, cette propriété est à une seule valeur. La cardinalité [0..n] indique qu'il peut y avoir 0, 1, ou plusieurs objets SimplePolicyAction qui contiennent n'importe quel objet de valeur de politique.

La propriété de référence "PartComponent" est héritée de PolicyComponent, et écrasée pour devenir une référence d'objet à une PolicyValue, c'est-à-dire définie dans la portée d'une SimplePolicyAction. Noter que pour toute instance seule de la classe d'association PolicyValueInSimplePolicyAction, cette propriété (comme toutes les propriétés de référence) est à une seule valeur. La cardinalité [1..1] indique qu'une SimplePolicyAction doit avoir exactement une valeur de politique définie dans la portée afin d'avoir un sens.

7.17 Association "ReusablePolicy"

L'association ReusablePolicy rend possible d'inclure toute sous classe de la classe abstraite "Policy" dans un ReusablePolicyContainer.

NOM	ReusablePolicy
DESCRIPTION	Classe qui représente l'inclusion d'un élément de politique réutilisable dans un ReusablePolicyContainer. Les éléments réutilisables peuvent être PolicyGroups, PolicyRules, PolicyConditions, PolicyActions, PolicyVariables, PolicyValues, ou des instances de toute autre sous classe de la classe abstraite Policy.
DÉRIVÉ DE	PolicyInSystem

ABSTRAITE	FAUX
PROPRIÉTÉS	Antecedent[ref ReusablePolicyContainer[0..1]]

7.18 "PolicyConditionInPolicyRepository" de PCIM déconseillé

NOM	PolicyConditionInPolicyRepository
DÉCONSEILLÉ POUR	ReusablePolicy
DESCRIPTION	Classe qui représente l'inclusion d'une PolicyCondition réutilisable dans un PolicyRepository.
DÉRIVÉ DE	PolicyInSystem
ABSTRAITE	FAUX
PROPRIÉTÉS	Antecedent[ref PolicyRepository[0..1]] ; Dependent[ref PolicyCondition[0..n]]

7.19 "PolicyActionInPolicyRepository" de PCIM déconseillé

NOM	PolicyActionInPolicyRepository
DÉCONSEILLÉ POUR	ReusablePolicy
DESCRIPTION	Classe qui représente l'inclusion d'une PolicyAction réutilisable dans un PolicyRepository.
DÉRIVÉ DE	PolicyInSystem
ABSTRAITE	FAUX
PROPRIÉTÉS	Antecedent[ref PolicyRepository[0..1]] ; Dependent[ref PolicyAction[0..n]]

7.20 Association ExpectedPolicyValuesForVariable

Cette association relie un objet PolicyValue à un objet PolicyVariable, modélisant l'ensemble de valeurs attendues pour cette PolicyVariable. En utilisant cette association, une variable (instance) peut être contrainte de se lier/de se voir allouer seulement un ensemble de valeurs admises. Par exemple, modéliser une énumération de variable d'accès de source, dont une crée une instance de la classe PolicySourcePortVariable et y associe l'ensemble des valeurs (entiers) qui représentent l'énumération admise, en utilisant le nombre d'instances approprié de l'association ExpectedPolicyValuesForVariable.

Noter qu'une seule instance de variable peut être contrainte par un nombre quelconque de valeurs, et qu'une seule valeur peut être utilisée pour contraindre un nombre quelconque de variables. Ces relations sont manifestées par la cardinalité n-à-m de l'association.

L'objet de cette association est de prendre en charge la validation de conditions simples de politique et d'actions simples de politique, avant leur déploiement sur un point de mise en application. Cette association, et l'objet PolicyValue auquel on se réfère, ne joue aucun rôle lorsque un PDP ou un PEP évalue une condition de politique simple, ou exécute une action simple de politique. Voir au paragraphe 5.8.3 les détails sur ce point.

La définition de classe pour l'association est la suivante :

NOM	ExpectedPolicyValuesForVariable
DESCRIPTION	Classe qui représente l'association d'un ensemble de valeurs attendues à un objet de variable.
DÉRIVÉ DE	Dependency
ABSTRAITE	FAUX
PROPRIÉTÉS	Antecedent [ref PolicyVariable[0..n]] ; Dependent [ref PolicyValue [0..n]]

La propriété de référence Antecedent est héritée de Dependency. Son type et sa cardinalité sont écrasés pour fournir la sémantique d'une variable qui a facultativement une contrainte de valeur. La cardinalité [0..n] indique que tout nombre de variables peut être contraint par une valeur donnée.

La propriété de référence "Dependent" est héritée de Dependency, et écrasée pour devenir une référence d'objet à une PolicyValue qui représente les valeurs que peut avoir une PolicyVariable particulière. La cardinalité [0..n] indique qu'une certaine variable de politique peut avoir 0, 1 ou plusieurs PolicyValues définies pour modéliser le ou les ensembles de valeurs que peut prendre la variable de politique.

7.21 Agrégation "ContainedDomain"

L'agrégation ContainedDomain donne un moyen pour incorporer un ReusablePolicyContainer à l'intérieur d'un autre. L'agrégation est définie au niveau de la super classe de ReusablePolicyContainer, AdminDomain, pour lui permettre de s'appliquer à des domaines autres que le cœur de politique.

NOM	ContainedDomain
DESCRIPTION	Une classe qui représente l'agrégation de domaines administratifs de niveau inférieur par un AdminDomain. de niveau supérieur
DÉRIVÉ DE	SystemComponent
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref AdminDomain [0..n]] ; PartComponent[ref AdminDomain [0..n]]

7.22 "PolicyRepositoryInPolicyRepository" déconseillé de PCIM

NOM	PolicyRepositoryInPolicyRepository
DÉCONSEILLÉ POUR	ContainedDomain
DESCRIPTION	Une classe qui représente l'agrégation de PolicyRepositories par un PolicyRepository de niveau supérieur.
DÉRIVÉ DE	SystemComponent
ABSTRAITE	FAUX
PROPRIÉTÉS	GroupComponent[ref PolicyRepository[0..n]] ; PartComponent[ref PolicyRepository[0..n]]

7.23 Agrégation "EntriesInFilterList"

Cette agrégation est une spécialisation de l'agrégation Component ; elle est utilisée pour définir un ensemble d'entrées de filtres (sous classes de FilterEntryBase) qui sont agrégées par une FilterList.
La cardinalité de l'agrégation elle-même est 0..1 sur l'extrémité de FilterList, et 0..n sur l'extrémité FilterEntryBase. Donc, en général, une entrée de filtre peut exister sans être agrégée dans une FilterList. Cependant, la seule façon dont une entrée de filtre peut figurer dans le modèle PCIMe est d'être agrégée dans une FilterList par cette agrégation.

La définition de classe pour cette agrégation est la suivante :

NOM	EntriesInFilterList
DESCRIPTION	Agrégation utilisée pour définir un ensemble d'entrées de filtre (sous classes de FilterEntryBase) qui sont agrégées par une FilterList particulière.
DÉRIVÉ DE	Component
ABSTRAITE	Faux
PROPRIÉTÉS	GroupComponent[ref FilterList[0..1]], PartComponent[ref FilterEntryBase[0..n], EntrySequence

7.23.1 Référence GroupComponent

Cette propriété est écrasée dans cette agrégation pour représenter une référence d'objet à un objet FilterList (au lieu de l'objet ManagedSystemElement plus générique défini dans sa super classe). Elle restreint aussi la cardinalité de l'agrégat à 0..1 (au lieu du plus générique 0-ou-plus) représentant le fait qu'une entrée de filtre existe toujours dans le contexte d'au plus une FilterList.

7.23.2 Référence PartComponent

Cette propriété est écrasée dans cette agrégation pour représenter une référence d'objet à un objet FilterEntryBase (au lieu de l'objet plus générique ManagedSystemElement défini dans sa super classe). Cet objet représente une seule entrée de filtre, qui peut être agrégée avec d'autres entrées de filtre pour former la FilterList.

7.23.3 Propriété EntrySequence

C'est un entier non signé de 16 bits qui indique l'ordre de l'entrée de filtre par rapport à tous les autres dans la FilterList. La valeur par défaut de '0' indique que l'ordre n'est pas significatif, parce que les entrées dans cette FilterList sont ajoutées les unes aux autres par l'opérateur logique ET.

7.24 Agrégation "ElementInPolicyRoleCollection"

L'agrégation suivante est utilisée pour associer des ManagedElements à un objet PolicyRoleCollection qui représente un rôle tenu par ces ManagedElements.

NOM	ElementInPolicyRoleCollection
DESCRIPTION	Classe qui représente l'inclusion d'un ManagedElement dans une collection, spécifiée comme ayant un certain rôle. Tous les éléments gérés dans la collection partagent le même rôle.
DÉRIVÉ DE	MemberOfCollection
ABSTRAITE	FAUX
PROPRIÉTÉS	Collection[ref PolicyRoleCollection [0..n]] ; Member[ref ManagedElement [0..n]]

7.25 Association faible "PolicyRoleCollectionInSystem"

Une PolicyRoleCollection est définie dans la portée d'un System. Cette association relie une PolicyRoleCollection au système dans la portée duquel elle est définie.
Lorsque on associe une PolicyRoleCollection à un System, cela devrait être fait en cohérence avec le système qui donne sa portée aux règles/groupes de politique qui sont appliqués aux ressources dans cette collection. Une PolicyRoleCollection est associée au même système que les PolicyRules et/ou PolicyGroups applicables, ou à un système supérieur dans l'arborescence formée par l'association SystemComponent.

La définition de classe pour cette association est la suivante :

NOM	PolicyRoleCollectionInSystem
DESCRIPTION	Classe qui représente le fait qu'une PolicyRoleCollection est définie dans la portée d'un System.
DÉRIVÉ DE	Dependency
ABSTRAITE	FAUX
PROPRIÉTÉS	Antecedent[ref System[1..1]] ; Dependent[ref PolicyRoleCollection[weak]]

La propriété de référence Antecedent est héritée de Dependency, et écrasée pour devenir une référence d'objet à un System, et pour restreindre sa cardinalité à [1..1]. Elle sert de référence d'objet à un System qui donne une portée pour une ou plusieurs PolicyRoleCollections. Comme c'est une association faible, la cardinalité pour cette référence d'objet est toujours 1, c'est-à-dire, une PolicyRoleCollection est toujours définie dans la portée d'exactement un System.

La propriété de référence Dependent est héritée de Dependency, et écrasée pour devenir une référence d'objet à une PolicyRoleCollection définie dans la portée d'un System. Noter que pour toute instance seule de la classe d'association PolicyRoleCollectionInSystem, cette propriété (comme toutes les propriétés Reference) est à une seule valeur. La cardinalité [0..n] indique qu'un System donné peut avoir 0, 1, ou plusieurs PolicyRoleCollections définies dans sa portée.

8. Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

9. Remerciements

Le point de départ du présent document était PCIM lui-même [RFC3060], et les trois premiers sous modèles qui en sont

dérivés, [RFC3644], [RFC3585], [MPLS]. Les auteurs de ces documents ont créé les extensions à PCIM, et posé les questions sur PCIM qui sont reflétées dans PCIMe.

10. Contributeurs

Le présent document inclut des textes écrits par un certain nombre d'auteurs (incluant l'éditeur) qui ont été fusionnés par l'éditeur. Les personnes suivantes ont contribué par des textes au présent document :

Lee Rafalow
IBM Corporation, BRQA/501
mél : rafalow@us.ibm.com

Yoram Ramberg
Cisco Systems
mél : yramberg@cisco.com

Yoram Snir
Cisco Systems
mél : ysnir@cisco.com

Andrea Westerinen
Cisco Systems
mél : andreaw@cisco.com

Ritu Chadha
Telcordia Technologies
mél :
chadha@research.telcordia.com

Marcus Brunner
NEC Europe Ltd.
mél :
brunner@ccrle.nec.de

Ron Cohen
Ntear LLC
mél : ronc@ntear.com

John Strassner
INTELLIDEN, Inc.
mél : john.strassner@intelliden.com

11. Considérations sur la sécurité

Le modèle d'informations de cœur de politique (PCIM, *Policy Core Information Model*) [RFC3060] décrit les considérations générales sur la sécurité qui se rapportent au modèle général de cœur de politique. Les extensions définies dans le présent document n'introduisent aucune considération supplémentaire relative à la sécurité.

12. Références normatives

[CIM] Distributed Management Task Force, Inc., "DMTF Technologies: CIM Standards CIM Schema: Version 2.5", disponible à http://www.dmtf.org/standards/cim_schema_v25.php .

[CIMv2.2] Distributed Management Task Force, Inc., "Common Information Model (CIM) Specification: Version 2.2", 14 juin 1999, disponible à http://www.dmtf.org/standards/documents/CIM/DSP0004.pdf .

[RFC1035] P. Mockapetris, "Noms de domaines – Mise en œuvre et spécification", STD 13, novembre 1987. (*MàJ par la* RFC6604)

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2234] D. Crocker et P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", novembre 1997. (*Obsolète, voir* RFC5234)

[RFC2252] M. Wahl, A. Coulbeck, T. Howes, S. Kille, "Protocole léger d'accès à un répertoire (v3) : Définitions de syntaxe d'attribut", décembre 1997. (*Obsolète, voir* RFC4510, RFC4517, RFC4523, RFC4512) (*MàJ par* RFC3377) *(P.S.)*

[RFC2373] R. Hinden, S. Deering, "Architecture d'adressage IP version 6", juillet 1998. (*Obsolète, voir* RFC4291) *(PS)*

[RFC3060] B. Moore et autres, "Spécification du modèle d'information de cœur de politique -- version 1", février 2001. *(P.S.)*

13. Références pour information

[MPLS] Chadha, R., et M. Brunner, M. Yoshida, J. Quittek, G. Mykoniatis, A. Poylisher, R. Vaidyanathan, A. Kind, F. Reichmeyer, "Policy Framework MPLS Information Model for QoS et TE", *(Non publiée)*

[RFC2028] R. Hovey, S. Bradner, "Les organisations impliquées dans le processus de normalisation de l'IETF", octobre 1996. (*MàJ par* RFC3668, RFC3979) (BCP0011)

[RFC3198] A. Westerinen et autres, "Terminologie pour la gestion fondée sur la politique", novembre 2001. *(Information)*

[RFC3585] J. Jason, L. Rafalow, E. Vyncke, "Modèle d'informations de politique de configuration IPsec", août 2003. *(P.S.)*

[RFC3644] Y. Snir et autres, "Modèle d'informations de politique de qualité de service (QS)", novembre 2003. *(P.S.)*

[RFC3670] B. Moore et autres, "Modèle d'information pour décrire les mécanismes de qualité de service d'appareils réseau sur le chemin de transmission", janvier 2004. *(P.S.)*

[RFC4011] S. Waldbusser et autres, "MIB de gestion fondée sur la politique", mars 2005. *(P.S.)*

Adresse de l'auteur

Bob Moore
IBM Corporation, BRQA/501
4205 S. Miami Blvd.
Research Triangle Park, NC 27709
USA
téléphone : +1 919-254-4436
Fax : +1 919-254-6243
mél : remoore@us.ibm.com

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par l'Internet Society.