

Groupe de travail Réseau
Request for Comments : 2622
 RFC rendue obsolète : 2280
 Catégorie : En cours de normalisation
 juin 1999

C. Alaettinoglu, USC/ISI
 C. Villamizar, Avici Systems
 E. Gerich, At Home Network
 D. Kessens, Qwest Communications
 D. Meyer, University of Oregon
 T. Bates, Cisco Systems
 D. Karrenberg, RIPE NCC
 M. Terpstra, Bay Networks

Traduction Claude Brière de L'Isle

Langage de spécification de politique d'acheminement (RPSL)

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.

Notice de copyright

Copyright (C) The Internet Society (1999). Tous droits réservés.

Résumé

RPSL donne à un opérateur de réseau la capacité de spécifier des politiques d'acheminement à divers niveaux dans la hiérarchie Internet ; par exemple, au niveau du système autonome. En même temps, des politiques peuvent être spécifiées dans RPSL avec un détail suffisant pour que les configurations de routeur de bas niveau puissent être générées à partir d'elles. RPSL est extensible ; de nouveaux protocoles d'acheminement et de nouvelles caractéristiques de protocoles peuvent être introduits à tout moment.

Table des Matières

1. Introduction.....	2
2. Noms RPSL, mots réservés, et représentation.....	3
3. Informations de contact.....	4
3.1 Classe mntner.....	5
3.2 Classe person.....	6
3.3 Classe role.....	7
4. Classe route.....	7
5. Classes d'ensembles.....	8
5.1 Classe as-set.....	8
5.2 Classe route-set.....	9
5.3 Objets d'ensembles prédéfinis.....	10
5.4 Filtres et classe filter-set.....	10
5.5 Classe rtr-set.....	13
5.6 Appariements et classe peering-set.....	14
6 Classe aut-num.....	15
6.1 Attribut import : spécification de politique d'importation.....	16
6.2 Attribut export : spécification de politique d'exportation.....	17
6.3 Autres protocoles d'acheminement, protocole d'acheminement multi-protocoles et injection de chemins entre les protocoles.....	17
6.4 Résolution d'ambiguïté.....	18
6.5 Attribut par défaut : spécification de politique par défaut.....	19
6.6 Spécification de politique structurée.....	20
7 Classe dictionary.....	22
7.1 Dictionnaire RPSL initial et exemple d'actions et de filtres de politique.....	24
8. Classe Advanced route.....	27
8.1 Spécification de chemins agrégés.....	27
8.2 Spécification de chemins statiques.....	31
9. Classe inet-rtr.....	31
10 Extension à RPSL.....	33
10.1 Extensions en changeant la classe dictionary.....	33

10.2 Extensions en ajoutant de nouveaux attribut à des classes existantes.....	33
10.3 Extensions par ajout de nouvelles classes.....	34
10.4 Extensions en changeant la syntaxe des attributs RPSL existants.....	34
11. Considérations pour la sécurité.....	34
12. Remerciements.....	34
Références.....	34
A Sites des registres d'acheminement.....	35
B Règles de grammaire.....	35
C. Changements depuis la RFC2280.....	41
D. Adresse des auteurs.....	42
Déclaration complète de droits de reproduction.....	42

1. Introduction

Le présent mémoire est le document de référence pour le langage de spécification de politique d'acheminement (RPSL, *Routing Policy Specification Language*). RPSL donne à un opérateur de réseau la capacité de spécifier les politiques d'acheminement à divers niveaux dans la hiérarchie de l'Internet ; par exemple au niveau du système autonome (AS, *Autonomous System*). En même temps, les politiques peuvent être spécifiées dans RPSL avec un détail suffisant pour que les configurations de routeur de bas niveau puissent être générées à partir d'elles. RPSL est extensible ; de nouveaux protocoles d'acheminement et de nouvelles caractéristiques de protocoles peuvent être introduites à tout moment.

RPSL remplace le langage actuel de spécification de politique de l'Internet connu sous le nom de [RIPE-181] ou [RFC1786]. [RIPE-81] a été le premier langage déployé dans l'Internet pour spécifier les politiques d'acheminement. Il a été ensuite remplacé par [RIPE-181]. L'utilisation pratique de RIPE-181 a rendu évident que certaines politiques ne pouvaient pas être spécifiées et qu'il était nécessaire d'avoir un langage amélioré et plus généralisé. RPSL traite les limitations de RIPE-181.

RPSL a été conçu de façon à ce qu'une politique d'acheminement globale puisse être contenue dans une seule base de données répartie entretenue de façon coopérative pour améliorer l'intégrité de l'acheminement de l'Internet. RPSL n'est pas conçu comme un langage de configuration de routeur. RPSL est conçu de façon à ce que les configurations de routeurs puissent être générées à partir de la description de la politique pour un système autonome (classe aut-num) combinée avec la description d'un routeur (classe inet-rtr) en fournissant principalement l'identifiant de routeur, le numéro de système autonome du routeur, les interfaces et les homologues du routeur, et combinée avec des transpositions globales de base de données des ensembles d'AS en AS (classe as-set) et des AS et ensembles de chemins d'origine en préfixes de chemins (classes route et route-set). La population précise de la base de données RPSL peut aider à contribuer à de tels objectifs par des configurations de routeurs qui protègent contre une répartition accidentelle (ou malveillante) d'informations d'acheminement inappropriées, par la vérification de l'acheminement de l'Internet, et l'agrégation des limites au delà d'un seul AS.

RPSL est tourné vers l'objet ; c'est-à-dire que les objets contiennent des éléments d'informations de politique et d'administration. Ces objets sont enregistrés dans le registre des acheminements de l'Internet (IRR, *Internet Routing Registry*) par les organisations autorisées. Le processus d'enregistrement sort du domaine d'application du présent document. Prière de se référer à [IRR], [RFC1034], [RFC2650] pour des précisions sur l'IRR.

Dans les sections suivantes, on présente les classes qui sont utilisées pour définir divers objets de politique et d'administration. La classe "mntner" définit des entités autorisée à ajouter, supprimer et modifier un ensemble d'objets. Les classes "person" et "role" décrivent le personnel de contact technique et administratif. Les systèmes autonomes (AS) sont spécifiés en utilisant la classe "aut-num". Les chemins sont spécifiés en utilisant la classe "route". Des ensembles d'objets peuvent être définis en utilisant les classes "as-set", "route-set", "filter-set", "peering-set", et "rtr-set". La classe "dictionary" assure les possibilités d'extension du langage. La classe "inet-rtr" est utilisée pour spécifier les routeurs. Beaucoup de ces classes étaient à l'origine définies dans des documents antérieurs [RIPE-104], [RIPE-120], [RIPE-122], [RIPE-157], [RIPE-181] et ont été améliorées.

Le présent document est auto-suffisant. Cependant, le lecteur est invité à lire RIPE-181 [RFC1786] et les documents associés [RIPE-104], [RIPE-120], [RIPE-122], [RIPE-157] car ils fournissent un arrière plan significatif sur les motivations et les principes sous-jacents à RIPE-181 et par conséquent, à RPSL. Pour un document de vulgarisation sur RPSL, le lecteur devrait se reporter au document d'application de RPSL, la [RFC2650].

2. Noms RPSL, mots réservés, et représentation

Chaque classe a un ensemble d'attributs qui mémorisent une partie des informations sur les objets de la classe. Les attributs peuvent être obligatoires ou facultatifs : un attribut obligatoire doit être défini pour tous les objets de la classe ; les attributs facultatifs peuvent être sautés. Les attributs peuvent aussi être à une seule valeur ou multi-valeurs. Chaque objet est identifié de façon univoque par un ensemble d'attributs, désignés par la "clé" de la classe.

La valeur d'un attribut a un type. Les types suivants sont les plus largement utilisés. Noter que RPSL est insensible à la casse et que seuls les caractères du jeu de caractères ASCII peuvent être utilisés.

<object-name>

Dans RPSL, de nombreux objets ont un nom. Un <object-name> (*nom d'objet*) est constitué de lettres, de chiffres, du caractère souligné "_", et du caractère trait d'union "-"; le premier caractère d'un nom doit être une lettre, et le dernier caractère d'un nom doit être une lettre ou un chiffre. Les mots suivants sont réservés par RPSL, et ils ne peuvent pas être utilisés comme noms : any, as-any, rs-any, peeras, and, or, not, atomic, from, to, at, action, accept, announce, except, refine, networks, into, inbound, outbound.

Les noms qui commencent par certains préfixes sont réservés pour certains types d'objet. Les noms qui commencent par "as-" sont réservés aux noms d'ensembles. Les noms qui commencent par "rs-" sont réservés aux noms d'ensembles de chemins. Les noms qui commencent par "rtrs-" sont réservés aux noms d'ensembles de routeurs. Les noms qui commencent par "fltr-" sont réservés aux noms d'ensembles de filtres. Les noms qui commencent par "prng-" sont réservés aux noms d'ensemble d'homologues.

<as-number>

Un AS numéro x est représenté par la chaîne "ASx". C'est-à-dire, l'AS 226 est représenté par AS226.

<ipv4-address>

Une adresse IPv4 est représentée par une séquence de quatre entiers dans la gamme de 0 à 255 séparés par le caractère point ".". Par exemple, 128.9.128.5 représente une adresse IPv4 valide. Dans le reste de ce document, on peut se référer à des adresses IPv4 comme à des adresses IP.

<address-prefix>

Un préfixe d'adresse est représenté par une adresse IPv4 suivie par le caractère barre oblique "/" suivi par un entier dans la gamme de 0 à 32. Voici des préfixes d'adresse valides : 128.9.128.5/32, 128.9.0.0/16, 0.0.0.0/0 ; et les préfixes d'adresse suivants sont invalides : 0/0, 128.9/16 car 0 ou 128.9 ne sont pas des chaînes qui contiennent quatre entiers.

<address-prefix-range>

Une gamme de préfixes d'adresse est un préfixe d'adresse suivi par un opérateur de gamme facultatif. Les opérateurs de gamme sont :

\wedge - est l'opérateur exclusif le plus spécifique ; il représente le plus spécifique du préfixe d'adresse à l'exclusion du préfixe d'adresse lui-même. Par exemple, 128.9.0.0/16 \wedge - contient le plus spécifique de 128.9.0.0/16 sauf 128.9.0.0/16.

\wedge + est l'opérateur inclusif le plus spécifique ; il représente le plus spécifique du préfixe d'adresse incluant le préfixe d'adresse lui-même. Par exemple, 5.0.0.0/8 \wedge + contient tout le plus spécifique de 5.0.0.0/8 y compris 5.0.0.0/8.

\wedge n où n est un entier, représente tous les éléments spécifiques de la longueur n du préfixe d'adresse. Par exemple, 30.0.0.0/8 \wedge 16 contient tous les éléments plus spécifiques de 30.0.0.0/8 qui sont de longueur 16 tels que 30.9.0.0/16.

\wedge n-m où n et m sont des entiers, représente tous les éléments spécifiques de la longueur n à la longueur m du préfixe d'adresse. Par exemple, 30.0.0.0/8 \wedge 24-32 contient tous les éléments plus spécifiques de 30.0.0.0/8 qui sont de longueur 24 à 32 tels que 30.9.9.96/28.

Les opérateurs de gamme peuvent aussi être appliqués aux ensembles de préfixes d'adresse. Dans ce cas, ils se répartissent sur les membres de l'ensemble. Par exemple, pour un route-set (défini plus loin) rs-foo, rs-foo \wedge + contient tous les éléments inclusifs plus spécifiques de tous les préfixes contenus dans rs-foo.

C'est une erreur de faire suivre un opérateur de gamme par un autre (par exemple, 30.0.0.0/8 \wedge 24-28 \wedge + est une erreur). Cependant, un opérateur de gamme peut être appliqué à un ensemble de préfixes d'adresse qui contient en lui des gammes de préfixes d'adresse (par exemple, {30.0.0.0/8 \wedge 24-28} \wedge 27-30 n'est pas une erreur). Dans ce cas, l'opérateur extérieur \wedge n-m se répartit sur l'opérateur interne \wedge k-l et devient l'opérateur \wedge max(n,k)-m si m est supérieur ou égal à max(n,k), ou autrement, le préfixe est supprimé de l'ensemble. Noter que l'opérateur \wedge n est équivalent à \wedge n-n ; préfix/l \wedge + est équivalent à

prefix/l^l-32 ; prefix/l⁻ est équivalent à prefix/l^(l+1)-32 ; {prefix/lⁿ-m}⁺ est équivalent à {prefix/lⁿ-32} ; et {prefix/lⁿ-m}⁻ est équivalent à {prefix/l⁽ⁿ⁺¹⁾-32}. Par exemple,

```
{128.9.0.0/16+}- == {128.9.0.0/16-}
{128.9.0.0/16-}+ == {128.9.0.0/16-}
{128.9.0.0/1617}24 == {128.9.0.0/1624}
{128.9.0.0/1620-24}26-28 == {128.9.0.0/1626-28}
{128.9.0.0/1620-24}22-28 == {128.9.0.0/1622-28}
{128.9.0.0/1620-24}18-28 == {128.9.0.0/1620-28}
{128.9.0.0/1620-24}18-22 == {128.9.0.0/1620-22}
{128.9.0.0/1620-24}18-19 == {}
```

<date> Une date est représentée par un entier de huit chiffres de la forme AAAAMMJJ où AAAA représente l'année, MM représente le mois de l'année (de 01 à 12) et JJ représente le jour du mois (de 01 à 31). Toutes les dates sont en UTC sauf mention contraire. Par exemple, le 24 juin 1996 est représenté par 19960624.

<email-address> est comme décrit dans la [RFC0822].

<dns-name> est comme décrit dans la [RFC1034].

<nic-handle> est un mot alloué comme identifiant univoque utilisé par l'acheminement, l'allocation d'adresse, et d'autres registres, pour se référer de façon non ambiguë aux informations de contact. Les classes person et role transposent les brides NIC en noms de personnes réelles, et en informations de contact.

<free-form> est une séquence de caractères ASCII.

<X-name> est un nom d'objet de type X. C'est-à-dire que <mntner-name> est le nom d'un objet mntner.

<registry-name> est le nom d'un registre d'IRR. Les registres d'acheminement sont énumérés dans l'Appendice A.

La valeur d'un attribut peut aussi être une liste d'un de ces types. Une liste est représentée en séparant les membres de la liste par des virgules ",". Par exemple, "AS1, AS2, AS3, AS4" est une liste de numéros d'AS. Noter qu'être une liste avec des valeurs et être avec multi-valeurs est orthogonal. Un attribut multi valeurs a plus d'une valeur, dont chacune peut être ou non une liste. D'un autre côté, un attribut qui a une seule valeur peut avoir une valeur de liste.

Un objet RPSL est représenté textuellement comme une liste de paires attribut-valeur. Chaque paire attribut-valeur est écrite sur une ligne séparée. Le nom de l'attribut commence à la colonne 0, suivi par le caractère ":" et suivi par la valeur de l'attribut. L'attribut qui a le même nom que la classe de l'objet devrait être spécifié en premier. La représentation de l'objet se termine lorsque se rencontre une ligne blanche. La valeur d'un attribut peut être développée sur plusieurs lignes, en ayant une espace, une tabulation ou un caractère plus ('+') comme premier caractère des lignes de continuation. Le caractère "+" pour la ligne de continuation permet que les valeurs d'attribut contiennent des lignes blanches. Plus d'espaces peuvent être utilisées facultativement après le caractère de continuation pour faciliter la lecture. L'ordre des paires attribut-valeur est significatif.

Une description d'objet peut contenir des commentaires. Un commentaire peut être n'importe où dans une définition d'objet ; il commence au premier caractère "#" sur une ligne et se termine au premier caractère de fin de ligne. Les caractères espace peuvent être utilisés pour faciliter la lecture.

Un entier peut être spécifié en utilisant (1) la notation du langage de programmation C (par exemple, 1, 12345) ; (2) une séquence de quatre entiers de un octet (dans la gamme de 0 à 255) séparés par le caractère point "." (par exemple, 1.1.1.1, 255.255.0.0) dans ce cas, un entier de 4 octets est formé en enchaînant ces entiers de un octet dans l'ordre du plus fort poids au moindre poids ; (3) une séquence d'entiers de deux octets (dans la gamme de 0 à 65 535) séparés par le caractère deux points ":" (par exemple, 3561:70, 3582:10) dans ce cas, un entier de quatre octets est formé en enchaînant ces entiers de deux octets dans l'ordre du plus fort poids au moindre poids.

3. Informations de contact

Les classes mntner, person et role, les attributs admin-c, tech-c, mnt-by, changed, et source de toutes les classes décrivent les informations de contact. La classe mntner spécifie aussi les informations d'authentification exigées pour créer, supprimer et mettre à jour d'autres objets. Ces classes ne spécifient pas de politiques d'acheminement et chaque registre peut avoir à leur égard des exigences différentes ou supplémentaires. On présente ici le dénominateur commun de ce qui est

la mise en œuvre de la base de données de RIPE [RIPE-157]. Prière de consulter dans votre registre les plus récentes spécifications de ces classes et attributs. Le document "Sécurité du système de politique d'acheminement" [RFC2725] décrit plus en détails le modèle d'authentification et d'autorisation.

3.1 Classe mntner

La classe mntner spécifie les informations d'authentification nécessaires pour créer, supprimer et mettre à jour les objets RPSL. Un fournisseur, avant de pouvoir créer des objets RPSL, doit d'abord créer un objet mntner. Les attributs de la classe mntner sont indiqués à la Figure 1. La classe mntner a été décrite d'abord dans [RIPE-120].

L'attribut mntner est obligatoire et est la clé de classe. Sa valeur est un nom RPSL. L'attribut auth spécifie le schéma qui sera utilisé pour identifier et authentifier les demandes de mise à jour à ce mainteneur. Il a la syntaxe suivante :

auth: <scheme-id> <auth-info>

Par exemple, auth: NONE

Attribut	Valeur	Type
mntner	<object-name>	obligatoire, une seule valeur, clé de classe
descr	<free-form>	obligatoire, une seule valeur
auth	voir la description dans le texte	obligatoire, multi-valeurs
upd-to	<email-address>	obligatoire, multi-valeurs
mnt-nfy	<email-address>	facultatif, multi-valeurs
tech-c	<nic-handle>	obligatoire, multi-valeurs
admin-c	<nic-handle>	facultatif, multi-valeurs
remarks	<free-form>	facultatif, multi-valeurs
notify	<email-address>	facultatif, multi-valeurs
mnt-by	liste de <mntner-name>	obligatoire, multi-valeurs
changed	<email-address> <date>	obligatoire, multi-valeurs
source	<registry-name>	obligatoire, une seule valeur

Figure 1 : Attributs de la classe mntner

auth: CRYPT-PW dhjsdfhruewf
auth: MAIL-FROM .*@ripe\.net

Les <scheme-id> actuellement définis sont : NONE, MAIL-FROM, PGP-KEY et CRYPT-PW. Les <auth-info> sont des informations supplémentaires requises par un schéma particulier : dans le cas de MAIL-FROM, c'est une expression régulière qui correspond à des adresses de messagerie électronique valides ; dans le cas de CRYPT-PW, c'est un mot de passe en format chiffré UNIX; et dans le cas de PGP-KEY, c'est un pointeur sur un objet key-certif [RFC2726] contenant la clé publique PGP de l'utilisateur. Si plusieurs attributs auth sont spécifiés, une demande de mise à jour satisfaisant l'un d'eux est authentifiée comme provenant du mainteneur.

L'attribut upd-to est une adresse de messagerie électronique. Sur une tentative non autorisée de mise à jour d'un objet entretenu par ce mainteneur, un message électronique sera envoyé à cette adresse. L'attribut mnt-nfy est une adresse de messagerie électronique. Un message de notification sera transmis à cette adresse chaque fois qu'un objet entretenu par ce mainteneur est ajouté, changé ou supprimé.

L'attribut descr est une brève description textuelle de forme libre de l'objet. L'attribut tech-c est une bride NIC de contact technique. C'est quelqu'un à contacter pour des problèmes techniques tels qu'une mauvaise configuration. L'attribut admin-c est une bride NIC de contact administratif. L'attribut remarks est un texte libre d'explication ou de précision. L'attribut notify est une adresse de messagerie électronique à laquelle devraient être envoyées les notifications de changements de cet objet. L'attribut mnt-by est une liste de noms d'objets mntner. L'autorisation des changements à cet objet est gouvernée par tout objet du mainteneur référencé. L'attribut changed indique qui est le dernier à avoir changé cet objet, et quand ce changement a été fait. Sa syntaxe a la forme suivante :

changed: <adresse-mél> <AAAAMMJJ>

Par exemple, changed: johndoe@terabit-labs.nn 19900401

Le <adresse-mél> identifie la personne qui a fait le dernier changement. <AAAAMMJJ> est la date du changement. L'attribut source spécifie le registre où l'objet est enregistré. La Figure 2 donne un exemple d'objet mntner. Dans

l'exemple, l'authentification par mot de passe en format UNIX crypt est utilisée.

```

mntner: RIPE-NCC-MNT
descr: RIPE-NCC Maintainer
admin-c: DK58
tech-c: OPS4-RIPE
upd-to: ops@ripe.net
mnt-nfy: ops-fyi@ripe.net
auth: CRYPT-PW lz1A7/JnfkTtI
mnt-by: RIPE-NCC-MNT
changed: ripe-dbm@ripe.net 19970820
source: RIPE

```

Figure 2 : Exemple d'objet mntner

Les attributs descr, tech-c, admin-c, remarks, notify, mnt-by, changed et source sont des attributs de toutes les classes RPSL. Leur syntaxe, sémantique, et statuts : obligatoire, facultatif, multi-valeurs, ou une seule valeur, sont les mêmes pour toutes les classes RPSL. La seule exception est l'attribut admin-c qui est obligatoire pour la classe aut-num. On n'en discutera plus dans les autres sections.

3.2 Classe person

Une classe person est utilisée pour décrire les informations sur les gens. Bien qu'elle ne décrive pas la politique d'acheminement, on la décrit quand même brièvement ici car de nombreux objets de politique font référence à des objets person. La classe person a d'abord été décrite dans [RIPE-119].

Les attributs de la classe person sont montrés à la Figure 3. L'attribut person est le nom complet de la personne. Les attributs phone et fax-no ont la syntaxe suivante :

phone: +<code de pays> <ville> <abonné> [ext. <extension>]

Par exemple : phone: +31 20 12334676

Attribut	Valeur	Type
person	<free-form>	obligatoire, une seule valeur
nic-hdl	<nic-handle>	obligatoire, une seule valeur, clé de classe
address	<free-form>	obligatoire, multi-valeurs
phone	voir la description dans le texte	obligatoire, multi-valeurs
fax-no	comme pour phone	facultatif, multi-valeurs
e-mail	<adresse-mél>	obligatoire, multi-valeurs

Figure 3 : Attributs de la classe person

phone: +44 123 987654 ext. 4711

La Figure 4 montre un exemple d'objet person.

```

person: Daniel Karrenberg
address: RIPE Network Coordination Centre (NCC)
address: Singel 258
address: NL-1016 AB Amsterdam
address: Netherlands
phone: +31 20 535 4444
fax-no: +31 20 535 4445
e-mail: Daniel.Karrenberg@ripe.net
nic-hdl: DK58
changed: Daniel.Karrenberg@ripe.net 19970616
source: RIPE

```

Figure 4 : Exemple d'objet person

3.3 Classe role

La classe role est similaire à l'objet person. Cependant, au lieu de décrire un être humain, elle décrit un rôle tenu par un ou plusieurs êtres humains. Des exemples sont les bureaux d'assistance, les centres de surveillance du réseau, les administrateurs systèmes, etc. L'objet role est particulièrement utile car souvent une personne qui tient un rôle peut changer, mais le rôle lui-même subsiste.

Les attributs de la classe role sont montrés à la Figure 5. Les attributs nic-hdl des classes person et role partagent le même espace de noms. L'attribut trouble de l'objet role peut contenir des informations de contact supplémentaires à utiliser lorsque survient un problème dans tout objet qui fait référence à cet objet role. La Figure 6 donne un exemple d'objet role.

Attribut	Valeur	Type
role	<free-form>	obligatoire, une seule valeur
nic-hdl	<nic-handle>	obligatoire, une seule valeur, clé de classe
trouble	<free-form>	facultatif, multi-valeurs
address	<free-form>	obligatoire, multi-valeurs
phone	voir la description dans le texte	obligatoire, multi-valeurs
fax-no	comme pour phone	facultatif, multi-valeurs
e-mail	<adresse-mél>	obligatoire, multi-valeurs

Figure 5 : Attributs de la classe role

```

role:                RIPE NCC Operations
trouble:
address:             Singel 258
address:             1016 AB Amsterdam
address:             The Netherlands
phone:               +31 20 535 4444
fax-no:              +31 20 545 4445
e-mail:              ops@ripe.net
admin-c:             CO19-RIPE
tech-c:              RW488-RIPE
tech-c:              JLSD1-RIPE
nic-hdl:             OPS4-RIPE
notify:              ops@ripe.net
changed:             roderik@ripe.net 19970926
source:              RIPE

```

Figure 6 : Exemple d'objet role

4. Classe route

Chaque chemin interAS (aussi appelé un chemin inter domaine) généré par un AS est spécifié en utilisant un objet route. Les attributs de la classe route sont montrés à la Figure 7. L'attribut route est le préfixe d'adresse du chemin et l'attribut origin est le numéro d'AS de l'AS qui génère le chemin dans le système d'acheminement interAS. La paire d'attributs route et origin est la clé de classe. La Figure 8 montre des exemples de quatre objets route (on n'inclut pas les attributs de contact tels que admin-c, tech-c par souci de concision). Noter que les deux derniers objets route ont le même préfixe d'adresse, à savoir 128.8.0.0/16. Cependant, ce sont des objets route différents car ils sont générés par des AS différents (c'est-à-dire, ils ont des clés différentes).

Attribut	Valeur	Type
route	<address-prefix>	obligatoire, une seule valeur, clé de classe
origin	<as-number>	obligatoire, une seule valeur, clé de classe
member-of	liste de <route-set-names>	facultatif, multi-valeurs
inject	voir Section 8	facultatif, multi-valeurs
components	voir Section 8	facultatif, une seule valeur
aggr-bndry	voir Section 8	facultatif, une seule valeur
aggr-mtd	voir Section 8	facultatif, une seule valeur
export-comps	voir Section 8	facultatif, une seule valeur
holes	voir Section 8	facultatif, multi-valeurs

Figure 7 : Attributs de la classe route

```
route: 128.9.0.0/16
origin: AS226
```

```
route: 128.99.0.0/16
origin: AS226
```

```
route: 128.8.0.0/16
origin: AS1
```

```
route: 128.8.0.0/16
origin: AS2
```

Figure 8 : Objets Route

5. Classes d'ensembles

Pour spécifier des politiques, il est souvent utile de définir des ensembles d'objets. À cette fin, on définit les classes as-set, route-set, rtr-set, filter-set, et peering-set. Ces classes définissent un ensemble désigné. Les membres de ces ensembles peuvent être spécifiés soit directement en les énumérant dans la définition des ensembles, soit indirectement en faisant que les objets membres se réfèrent aux noms des ensembles, ou à une combinaison des deux méthodes.

Un nom d'ensemble est un mot rpsl avec les restrictions suivantes : tous les noms de as-set commencent par le préfixe "as-". Tous les noms de route-set commencent par le préfixe "rs-". Tous les noms de rtr-set commencent par le préfixe "rtrs-". Tous les noms de filter-set commencent par le préfixe "fltr-". Tous les noms de peering-set commencent par le préfixe "prng-". Par exemple, as-foo est un nom valide de as-set.

Les noms d'ensembles peuvent aussi être hiérarchiques. Un nom d'ensemble hiérarchique est une séquence de noms d'ensembles et de numéros d'AS séparés par deux points ":". Au moins un composant d'un tel nom doit être un nom d'ensemble réel (c'est-à-dire, commencer par un des préfixes ci-dessus). Tous les composants de nom d'ensemble d'un nom hiérarchique doivent être du même type. Par exemple, les noms suivants sont valides : AS1:AS-CUSTOMERS, AS1:RS-EXPORT:AS2, RS-EXCEPTIONS:RS-BOGUS.

L'objet d'un nom d'ensemble hiérarchique est de partager l'espace du nom d'ensemble de telle sorte que les mainteneurs de l'ensemble X1 contrôlent la totalité de l'espace de nom d'ensemble sous-jacent, c'est-à-dire, X1:...:Xn-1. Donc, un objet set avec le nom X1:...:Xn-1:Xn ne peut être créé que par le mainteneur de l'objet qui a le nom X1:...:Xn-1. C'est-à-dire que seul le mainteneur de AS1 peut créer un ensemble qui a le nom AS1:AS-FOO ; et seul le mainteneur de AS1:AS-FOO peut créer un ensemble du nom de AS1:AS-FOO:AS-BAR. Prière de voir les détails dans le document sur la sécurité de RPS [RFC2725].

5.1 Classe as-set

Les attributs de la classe as-set sont montrés à la Figure 9. L'attribut as-set définit le nom de l'ensemble. Il a un nom RPSL qui commence par "as-". Les attributs members font la liste des membres de l'ensemble. L'attribut members est une liste des numéros d'AS, ou d'autres noms d'as-set.

Attribut	Valeur	Type
as-set	<object-name>	obligatoire, une seule valeur, clé de classe
members	liste de <as-numbers> ou <as-set-names>	facultatif, multi-valeurs
mbrs-by-ref	liste de <mntner-names>	facultatif, multi-valeurs

Figure 9 : Attributs de la classe as-set

La Figure 10 présente deux objets as-set. L'ensemble as-foo contient deux AS, à savoir AS1 et AS2. L'ensemble as-bar contient les membres de l'ensemble as-foo et AS3, c'est-à-dire qu'il contient AS1, AS2, AS3. L'ensemble as-empty ne contient pas de membre.

```
as-set: as-foo          as-set: as-bar          as-set: as-empty
members: AS1, AS2      members: AS3, as-foo
```

Figure 10 : Objets as-set

L'attribut `mbrs-by-ref` est une liste des noms de mainteneurs ou le mot clé ANY. Si cet attribut est utilisé, l'ensemble d'AS inclut aussi les AS dont les objets `aut-num` sont enregistrés par un de ces mainteneurs et dont l'attribut `member-of` se réfère au nom de cet ensemble d'AS. Si la valeur d'un attribut `mbrs-by-ref` est ANY, tout objet AS qui se réfère à l'ensemble d'AS est un membre de l'ensemble. Si l'attribut `mbrs-by-ref` manque, seuls les AS figurant sur la liste de l'attribut `members` sont des membres de l'ensemble.

```

as-set: as-foo
members: AS1, AS2
mbrs-by-ref: MNTR-ME

aut-num: AS3          aut-num: AS4
member-of: as-foo    member-of: as-foo
mnt-by: MNTR-ME      mnt-by: MNTR-OTHER

```

Figure 11 : Objets as-set

La Figure 11 présente un exemple d'objet `as-set` qui utilise l'attribut `mbrs-by-ref`. L'ensemble `as-foo` contient AS1, AS2 et AS3. AS4 n'est pas un membre de l'ensemble `as-foo` même si l'objet `aut-num` fait référence à `as-foo`. C'est parce que `MNTR-OTHER` ne figure pas dans la liste de l'attribut `mbrs-by-ref` de `as-foo`.

5.2 Classe route-set

Les attributs de la classe `route-set` sont montrés à la Figure 12. L'attribut `route-set` définit le nom de l'ensemble. C'est un nom RPSL qui commence par "rs-". L'attribut `members` fait la liste des membres de l'ensemble. L'attribut `members` est une liste de préfixes d'adresses ou d'autres noms de `route-set`. Noter que la classe `route-set` est un ensemble de préfixes de chemins, et non d'objets RPSL `route`.

Attribut	Valeur	Type
<code>route-set</code>	<object-name>	obligatoire, une seule valeur, clé de classe
<code>members</code>	liste de <address-prefix-range> ou <route-set-name> ou <route-set-name><range-operator>	facultatif, multi-valeurs
<code>mbrs-by-ref</code>	liste de <mntner-names>	facultatif, multi-valeurs

Figure 12 : Attributs de la classe route-set

La Figure 13 présente des exemples d'objet `route-set`. L'ensemble `rs-foo` contient deux préfixes d'adresse, à savoir `128.9.0.0/16` et `128.9.0.0/24`. L'ensemble `rs-bar` contient les membres de l'ensemble `rs-foo` et le préfixe d'adresse `128.7.0.0/16`.

Un préfixe d'adresse ou un nom `route-set` dans un attribut `members` peut facultativement être suivi par un opérateur `range`. Par exemple, l'ensemble suivant :

```

route-set: rs-foo
members: 128.9.0.0/16, 128.9.0.0/24

route-set: rs-bar
members: 128.7.0.0/16, rs-foo

```

Figure 13 : Objets route-set

```

route-set: rs-bar
members: 5.0.0.0/8^+, 30.0.0.0/8^24-32, rs-foo^+

```

contient tout le plus spécifique de `5.0.0.0/8` incluant `5.0.0.0/8`, tout le plus spécifique de `30.0.0.0/8` qui sont de longueur de 24 à 32 comme `30.9.9.96/28`, et tout le plus spécifique des préfixes d'adresse dans l'ensemble de chemins `rs-foo`.

L'attribut `mbrs-by-ref` est une liste des noms de mainteneurs ou le mot clé ANY. Si cet attribut est utilisé, l'ensemble de chemins inclut aussi les préfixes d'adresses dont l'objet `route` est enregistré par un de ces mainteneurs et dont l'attribut `member-of` se réfère au nom de cet ensemble de chemins. Si la valeur d'un attribut `mbrs-by-ref` est ANY, tout objet `route` qui se réfère au nom de l'ensemble de chemins est un membre. Si l'attribut `mbrs-by-ref` manque, seuls les préfixes d'adresse énumérés dans l'attribut `members` sont membres de l'ensemble.

```
route-set: rs-foo
mbrs-by-ref: MNTR-ME, MNTR-YOU
```

```
route-set: rs-bar
members: 128.7.0.0/16
mbrs-by-ref: MNTR-YOU
```

```
route: 128.9.0.0/16
origin: AS1
member-of: rs-foo
mnt-by: MNTR-ME
```

```
route: 128.8.0.0/16
origin: AS2
member-of: rs-foo, rs-bar
mnt-by: MNTR-YOU
```

Figure 14 : Objets route-set

La Figure 14 présente un exemple d'objets route-set qui utilise l'attribut mbrs-by-ref. L'ensemble rs-foo contient deux préfixes d'adresse, à savoir 128.8.0.0/16 et 128.9.0.0/16 car les objets route pour 128.8.0.0/16 et 128.9.0.0/16 se réfèrent au nom de l'ensemble rs-foo dans l'attribut member-of. L'ensemble rs-bar contient les préfixes d'adresse 128.7.0.0/16 et 128.8.0.0/16. Le chemin 128.7.0.0/16 est mentionné explicitement dans l'attribut members de rs-bar, et l'objet route pour 128.8.0.0/16 se réfère au nom de l'ensemble rs-bar dans son attribut member-of.

Noter que, si un préfixe d'adresse est cité dans un attribut members d'un ensemble de chemins, il est un membre de cet ensemble de chemins. L'objet route qui correspond à ce préfixe d'adresse n'a pas besoin de contenir un attribut member-of qui se réfère à ce nom d'ensemble. L'attribut member-of de la classe route est un mécanisme supplémentaire pour spécifier indirectement les membres.

5.3 Objets d'ensembles prédéfinis

Dans un contexte qui attend un ensemble de chemins (par exemple, un attribut members de la classe route-set) un numéro d'AS ASx définit l'ensemble des chemins qui sont générés par ASx ; et un AS-X as-set définit l'ensemble des chemins qui sont générés par les AS dans AS-X. Un chemin p est dit être généré par l'ASx si il y a un objet route pour p avec ASx comme valeur de l'attribut origin. Par exemple, dans la Figure 15, l'ensemble de chemins rs-special contient 128.9.0.0/16, des chemins de AS1 et AS2, et des chemins des AS dans l'ensemble d'AS AS-FOO.

```
route-set: rs-special
members: 128.9.0.0/16, AS1, AS2, AS-FOO
```

Figure 15 : Utilisation des numéros d'AS et des ensembles d'AS dans les ensembles de chemins

L'ensemble rs-any contient tous les chemins enregistrés dans l'IRR. L'ensemble as-any contient tous les AS enregistrés dans l'IRR.

5.4 Filtres et classe filter-set

Les attributs de la classe filter-set sont montrés à la Figure 16. Un objet filter-set définit un ensemble de chemins qui correspondent à ce filtre. L'attribut filter-set définit le nom du filtre. C'est un nom RPSL qui commence par "fltr-".

Attribut	Valeur	Type
filter-set	<object-name>	obligatoire, une seule valeur, clé de classe
filter	<filter>	obligatoire, une seule valeur

Figure 16 : Attributs de la classe filter

```

filter-set: fltr-foo
filter: { 5.0.0.0/8, 6.0.0.0/8 }

filter-set: fltr-bar
filter: (AS1 ou fltr-foo) et <AS2>

```

Figure 17 :Objets filter-set

L'attribut filter définit le filtre de politique de l'ensemble. Un filtre de politique est une expression logique qui lorsque elle est appliquée à un ensemble de chemins retourne un sous ensemble de ces chemins. On dit que le filtre de politique correspond au sous ensemble retourné. Le filtre de politique peut correspondre à des chemins en utilisant tout attribut de chemin BGP, comme le préfixe d'adresse de destination (ou NLRI), AS-path, ou des attributs de communauté.

Les filtres de politique peuvent être composites en utilisant les opérateurs AND, OR, et NOT. Les filtres de politique suivants peuvent être utilisés pour choisir un sous-ensemble de chemins :

ANY : le mot-clé ANY correspond à tous les chemins.

Address-Prefix Set : c'est une liste explicite de préfixes d'adresses incluse entre des accolades '{' et '}'. Le filtre de politique correspond à l'ensemble des chemins dont le préfixe d'adresse de destination est dans l'ensemble. Par exemple :

```

{ 0.0.0.0/0 }
{ 128.9.0.0/16, 128.8.0.0/16, 128.7.128.0/17, 5.0.0.0/8 }
{ }

```

Un préfixe d'adresse peut facultativement être suivi par un opérateur range (c'est-à-dire que

```

{ 5.0.0.0/8^+, 128.9.0.0/16^-, 30.0.0.0/8^16, 30.0.0.0/8^24-32 }

```

contient tout le plus spécifique de 5.0.0.0/8 incluant 5.0.0.0/8, tout le plus spécifique de 128.9.0.0/16 à l'exclusion de 128.9.0.0/16, tout le plus spécifique de 30.0.0.0/8 qui est de longueur 16 tel que 30.9.0.0/16, et tout le plus spécifique de 30.0.0.0/8 qui est de longueur de 24 à 32 tel que 30.9.96/28.

Route Set Name : un nom d'ensemble de chemins correspond à l'ensemble des chemins qui sont membres de l'ensemble. Un nom d'ensemble de chemins peut être un nom d'un objet route-set, un numéro d'AS, ou un nom d'un objet as-set (les numéros d'AS et les noms de as-set définissent implicitement des ensembles de chemins ; voir au paragraphe 5.3). Par exemple :

```

aut-num: AS1
import: de AS2 accept AS2
import: de AS2 accept AS-FOO
import: de AS2 accept RS-FOO

```

Le mot clé PeerAS peut être utilisé à la place du numéro d'AS de l'AS homologue. PeerAS est particulièrement utile lorsque l'appariement est spécifié en utilisant une expression d'AS. Par exemple :

```

as-set: AS-FOO
members: AS2, AS3

```

```

aut-num: AS1
import: de AS-FOO accept PeerAS

```

est la même chose que :

```

aut-num: AS1
import: de AS2 accept AS2
import: de AS3 accept AS3

```

Un route set name peut aussi être suivi par un des opérateurs '^-', '^+', exemple, { 5.0.0.0/8, 6.0.0.0/8 }^+ est égal à { 5.0.0.0/8^+, 6.0.0.0/8^+ }, et AS1^- égale tous les exclusivement plus spécifiques des chemins générés par AS1.

AS Path Regular Expression : une expression régulière AS-path peut être utilisée comme filtre de politique en incluant l'expression dans '<' et '>'. Un filtre de politique AS-path correspond à l'ensemble de chemins qui traversent une

séquence d'AS qui correspondent à l'expression régulière AS-path. Un routeur peut vérifier cela en utilisant l'attribut AS_PATH dans le protocole de passerelle frontière (BGP, *Border Gateway Protocol*) [RFC1771], ou l'attribut RD_PATH dans le protocole d'acheminement inter domaine (IRDP, *Inter-Domain Routing Protocol*) [IDRP].

Les expressions régulières AS-path sont des expressions régulières conformes à POSIX sur l'alphabet des numéros d'AS. La construction d'expression régulière est la suivante :

ASN : ASN est un numéro d'AS. ASN correspond au AS-path qui est de longueur 1 et contient le numéro d'AS correspondant (par exemple, l'expression régulière d'AS-path AS1 correspond au AS-path "1").

Le mot clé PeerAS peut être utilisé à la place du numéro d'AS de l'AS homologue.

AS-set : AS-set est un nom d'ensemble d'AS. AS-set correspond aux AS-path qui satisfont à une des AS dans l'AS-set.

. : correspond aux AS-path qui sont satisfaits par tout numéro d'AS.

[...] : c'est un ensemble de numéros d'AS. Il correspond aux AS-path qui sont satisfaits par les numéros d'AS cités entre les crochets. Les numéros d'AS dans l'ensemble sont séparés par des caractères espace. Si un '-' est utilisé entre deux numéros d'AS dans cet ensemble, tous les numéros d'AS entre les deux numéros d'AS sont inclus dans l'ensemble. Si un nom d'as-set figure dans la liste, tous les numéros d'AS dans l'as-set sont inclus.

[^...] : est un numéro d'AS complété. Il correspond à tout AS-path qui n'est pas satisfait par les numéros d'AS dans l'ensemble.

^ : correspond à la chaîne vide au début d'un AS-path.

\$: correspond à la chaîne vide à la fin d'un AS-path.

On donne maintenant la liste des opérateurs d'expressions régulières en ordre décroissant d'évaluation. Ces opérateurs sont associatifs à gauche, c'est à dire effectués de gauche à droite.

Opérateurs postfix unaires * + ? {m} {m,n} {m,}

Pour une expression régulière A, A* correspond à zéro, une ou plusieurs occurrences de A ; A+ correspond à zéro, une ou plusieurs occurrences de A ; A? correspond à zéro, une ou plusieurs occurrences de A ; A{m} correspond à m occurrences de A ; A{m,n} correspond à de m à n occurrences de A ; A{m,} correspond à m ou plus occurrences de A. Par exemple, [AS1 AS2]{2} correspond à AS1 AS1, AS1 AS2, AS2 AS1, et AS2 AS2.

Opérateurs postfix unaires ~* ~+ ~{m} ~{m,n} ~{m,}

Ces opérateurs ont des fonctions similaires à celles des opérateurs correspondants cités ci-dessus, mais toutes les occurrences de l'expression régulière doivent correspondre au même schéma. Par exemple, [AS1 AS2]~{2} correspond à AS1 AS1 et AS2 AS2, mais ne correspond pas à AS1 AS2 et AS2 AS1.

Opérateur d'enchaînement binaire

C'est un opérateur implicite et il existe entre deux expressions régulières A et B lorsque aucun autre opérateur explicite n'est spécifié. L'expression A B résultante correspond à un AS-path si A correspond à un préfixe de l'AS-path et B correspond au reste de l'AS-path.

Opérateur binaire alternatif (ou) |

Pour les expressions régulières A et B, A | B correspond à tout AS-path qui est satisfait par A ou B.

Des parenthèses peuvent être utilisées pour outrepasser l'ordre par défaut d'évaluation. Des espaces peuvent être utilisées pour améliorer la lisibilité.

Voici des exemples de filtres AS-path :

```
<AS3>
<^AS1>
<AS2$>
<^AS1 AS2 AS3$>
<^AS1 .* AS2$>
```

Le premier exemple correspond à tout chemin dont le AS-path contient AS3, le second correspond aux chemins dont le AS-path commence par AS1, le troisième correspond aux chemins dont le AS-path se termine par AS2, le quatrième correspond aux chemins dont le AS-path est exactement "1 2 3", et le cinquième correspond aux chemins dont le AS-path commence par AS1 et se termine en AS2 avec n'importe quel nombre de numéros d'AS entre eux.

Filtres de politique composites : les opérateurs suivants (en ordre d'évaluation décroissant) peuvent être utilisés pour former des filtres de politique composites :

NOT : étant donné un filtre de politique x, NOT x correspond à l'ensemble de chemins qui ne sont pas satisfaits par x. C'est la négation du filtre de politique x.

AND : étant donnés deux filtres de politique x et y, x AND y correspond à l'intersection des chemins qui sont satisfaits par x et qui sont satisfaits par y.

OR : étant donnés deux filtres de politique x et y, x OR y correspond à l'union des chemins qui sont satisfaits par x et qui sont satisfaits par y. Noter qu'un opérateur OR peut être implicite, c'est-à-dire que 'x y' est équivalent à 'x OR y'.

Par exemple,

```
NOT {128.9.0.0/16, 128.8.0.0/16}
AS226 AS227 OR AS228
AS226 AND NOT {128.9.0.0/16}
AS226 AND {0.0.0.0/0^0-18}
```

Le premier exemple correspond à tout chemin sauf 128.9.0.0/16 et 128.8.0.0/16. Le seconde exemple correspond aux chemins de AS226, AS227 et AS228. Le troisième exemple correspond aux chemins de AS226 sauf 128.9.0.0/16. Le quatrième exemple correspond aux chemins de AS226 dont la longueur n'est pas supérieure à 18.

Les filtres de politique d'attributs de politique d'acheminement peuvent aussi utiliser les valeurs d'autres attributs pour des comparaisons. Les attributs dont les valeurs peuvent être utilisées dans des filtres de politique sont spécifiés dans le dictionnaire RPSL. Prière de se référer à la Section 7 pour les détails. Un exemple qui utilise l'attribut de communauté BGP est donné ci-dessous :

```
aut-num: AS1
export: to AS2 announce AS1 AND NOT community(NO_EXPORT)
```

Les filtres qui utilisent les attributs de politique d'acheminement définis dans le dictionnaire sont évalués avant d'évaluer les opérateurs AND, OR et NOT.

Nom d'ensemble de filtres

Un nom d'ensemble de filtres correspond à l'ensemble des chemins qui sont satisfaits par son attribut de filtre. Noter que l'attribut de filtre d'un ensemble de filtres peut se référer de façon récurrente à d'autres noms d'ensemble de filtres. Par exemple dans la Figure 17, fltr-foo satisfait à { 5.0.0.0/8, 6.0.0.0/8 }, et fltr-bar satisfait aux chemins de AS1 ou { 5.0.0.0/8, 6.0.0.0/8 } si leur AS-path contenait AS2.

5.5 Classe rtr-set

Les attributs de la classe rtr-set sont montrés à la Figure 18. L'attribut rtr-set définit le nom de l'ensemble. C'est un nom RPSL qui commence par "rtrs-". L'attribut members fait la liste des membres de l'ensemble. L'attribut members est une liste de noms inet-rtr, d'adresses IPv4 ou d'autres noms rtr-set.

Attribut	Valeur	Type
rtr-set	<nom d'objet>	obligatoire, une seule valeur, clé de classe
members	liste de <noms inet-rtr> ou <noms rtr-set> ou <adresses IPv4>	facultatif, multi-valeurs
mbrs-by-ref	liste de <noms mntner>	facultatif, multi-valeurs

Figure 18 : Attributs de classe rtr-set

La Figure 19 présente deux objets rtr-set. L'ensemble rtrs-foo contient deux routeurs, à savoir rtr1.isp.net et rtr2.isp.net. L'ensemble rtrs-bar contient les membres de l'ensemble rtrs-foo et rtr3.isp.net, c'est-à-dire qu'il contient rtr1.isp.net, rtr2.isp.net, rtr3.isp.net.

```
rtr-set: rtrs-foo          rtr-set: rtrs-bar
members: rtr1.isp.net, rtr2.isp.net  members: rtr3.isp.net, rtrs-foo
```

Figure 19 : Objets rtr-set

L'attribut `mbrs-by-ref` est une liste de noms de mainteneurs ou le mot clé ANY. Si cet attribut est utilisé, l'ensemble de routeurs comporte aussi des routeurs dont les objets `inet-rtr` sont enregistrés par un de ces mainteneurs et dont l'attribut `member-of` se réfère au nom de cet ensemble de routeurs. Si la valeur d'un attribut `mbrs-by-ref` est ANY, tout objet `inet-rtr` se référant à l'ensemble de routeurs est un membre de l'ensemble. Si l'attribut `mbrs-by-ref` manque, seuls les routeurs figurant sur la liste de l'attribut `members` sont membres de l'ensemble.

```
rtr-set: rtrs-foo
members: rtr1.isp.net, rtr2.isp.net
mbrs-by-ref: MNTR-ME

inet-rtr: rtr3.isp.net
local-as: as1
ifaddr: 1.1.1.1 masklen 30
member-of: rtrs-foo
mnt-by: MNTR-ME
```

Figure 20 : Objets rtr-set

La Figure 20 présente un exemple d'objet `rtr-set` qui utilise l'attribut `mbrs-by-ref`. L'ensemble `rtrs-foo` contient `rtr1.isp.net`, `rtr2.isp.net` et `rtr3.isp.net`.

5.6 Appariements et classe `peering-set`

Les attributs de la classe `peering-set` sont montrés à la Figure 21. Un objet `peering-set` définit un ensemble d'appariements qui sont énumérés dans ses attributs `peering`. L'attribut `peering-set` définit le nom de l'ensemble. C'est un nom RPSL qui commence par "prng-".

Attribut	Valeur	Type
<code>peering-set</code>	<object-name>	obligatoire, une seule valeur, clé de classe
<code>peering</code>	<peering>	obligatoire, multi-valeurs

Figure 21 : Attributs de classe `filter`

L'attribut `peering` définit un appariement qui peut être utilisé pour importer ou exporter des chemins.

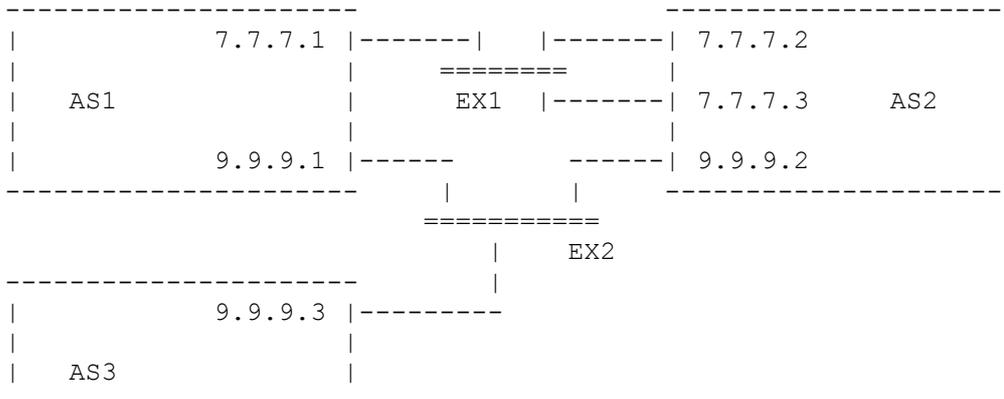


Figure 22 : Exemple de topologie consistant en trois AS, AS1, AS2, et AS3 ; deux points d'échange, EX1 et EX2 ; et six routeurs

Pour décrire les appariements, on va utiliser la topologie de la Figure 22. Dans cette topologie, il y a trois AS : AS1, AS2, et AS3 ; deux points d'échange, EX1 et EX2 ; et six routeurs. Les routeurs connectés au même point d'échange s'apparient avec chaque autre et échangent des informations d'acheminement. C'est à dire que 7.7.7.1, 7.7.7.2 et 7.7.7.3 s'apparient chacun avec chaque autre ; 9.9.9.1, 9.9.9.2 et 9.9.9.3 s'apparient chacun avec chaque autre.

La syntaxe d'une spécification d'appariement est :

```
<as-expression> [<router-expression-1>] [at <router-expression-2>] | <peering-set-name>
```

où <as-expression> est une expression sur les numéros d'AS et les ensembles d'AS qui utilise les opérateurs AND, OR, et

EXCEPT, et <router-expression-1> et <router-expression-2> sont des expressions sur les adresses IP de routeur, les noms inet-rtr, et les noms rtr-set qui utilisent les opérateurs AND, OR, et EXCEPT. L'opérateur binaire "EXCEPT" est l'opérateur de soustraction d'ensemble et a la même priorité que l'opérateur AND (il est sémantiquement équivalent à la combinaison "AND NOT"). C'est à dire que "(AS1 OR AS2) EXCEPT AS2" est égal à "AS1".

Cette forme identifie tous les appariements entre tout routeur local dans <router-expression-2> et un de ses routeurs homologues dans <router-expression-1> dans les AS dans <as-expression>. Si <router-expression-2> n'est pas spécifié, il prend comme valeur par défaut tous les routeurs de l'AS locale qui s'apparient avec les AS dans <as-expression>. Si <router-expression-1> n'est pas spécifié, il prend comme valeur par défaut tous les routeurs des AS homologues dans <as-expression> qui s'apparient avec l'AS locale.

Si un <peering-set-name> est utilisé, les appariements sont énumérés dans l'objet peering-set correspondant. Noter que les objets peering-set peuvent être récurrents.

De nombreuses formes particulières de cette spécification d'appariement générale sont possibles. Les exemples suivants illustrent les cas les plus courants, en utilisant l'attribut import de la classe aut-num. Dans l'exemple suivant, 7.7.7.1 importe 128.9.0.0/16 de 7.7.7.2.

```
(1) aut-num : AS1
    import: from AS2 7.7.7.2 at 7.7.7.1 accept { 128.9.0.0/16 }
```

Dans l'exemple suivant, 7.7.7.1 importe 128.9.0.0/16 de 7.7.7.2 et de 7.7.7.3.

```
(2) aut-num : AS1
    import: from AS2 at 7.7.7.1 accept { 128.9.0.0/16 }
```

Dans l'exemple suivant, 7.7.7.1 importe 128.9.0.0/16 de 7.7.7.2 et de 7.7.7.3, et 9.9.9.1 importe 128.9.0.0/16 de 9.9.9.2.

```
(3) aut-num: AS1
    import: from AS2 accept { 128.9.0.0/16 }
```

Dans l'exemple suivant, 9.9.9.1 importe 128.9.0.0/16 de 9.9.9.2 et de 9.9.9.3.

```
(4) as-set: AS-FOO
    members: AS2, AS3

    aut-num: AS1
    import: from AS-FOO at 9.9.9.1 accept { 128.9.0.0/16 }
```

Dans l'exemple suivant, 9.9.9.1 importe 128.9.0.0/16 de 9.9.9.2 et de 9.9.9.3, et 7.7.7.1 importe 128.9.0.0/16 de 7.7.7.2 et de 7.7.7.3.

```
(5) aut-num: AS1
    import: from AS-FOO accept { 128.9.0.0/16 }
```

Dans l'exemple suivant, AS1 importe 128.9.0.0/16 de AS3 au routeur 9.9.9.1

```
(6) aut-num: AS1
    import: from AS-FOO and not AS2 at not 7.7.7.1 accept { 128.9.0.0/16 }
```

Cela parce que "AS-FOO and not AS2" égale AS3 et "not 7.7.7.1" égale 9.9.9.1.

Dans l'exemple suivant, 9.9.9.1 importe 128.9.0.0/16 de 9.9.9.2 et de 9.9.9.3.

```
(7) peering-set: prng-bar
    peering: AS1 at 9.9.9.1
    peering-set: prng-foo
    peering: prng-bar
    peering: AS2 at 9.9.9.1
    aut-num: AS1
    import: from prng-foo accept { 128.9.0.0/16 }
```

6 Classe aut-num

Les politiques d'acheminement sont spécifiées à l'aide de la classe aut-num. Les attributs de la classe aut-num sont montrés à la Figure 23. La valeur de l'attribut aut-num est le numéro d'AS de l'AS décrite par cet objet. L'attribut as-name est un nom symbolique (dans la syntaxe des noms RPSL) de l'AS. Les politiques d'importation, d'exportation et par défaut de l'AS sont spécifiées en utilisant respectivement les attributs import, export et default.

Attribut	Valeur	Type
aut-num	<as-number>	obligatoire, une seule valeur, clé de classe
as-name	<object-name>	obligatoire, une seule valeur
member-of	liste de <as-set-names>	facultatif, multi-valeurs
import	voir paragraphe 6.1	facultatif, multi valued
export	voir paragraphe 6.2	facultatif, multi valued
default	voir paragraphe 6.5	facultatif, multi valued

Figure 23 : Attributs de la classe aut-num

6.1 Attribut import : spécification de politique d'importation

Dans RPSL, une politique d'importation est divisée en expressions de politique d'importation. Chaque expression de politique d'importation est spécifiée en utilisant un attribut import. L'attribut import a la syntaxe suivante (on développera cette syntaxe aux paragraphes 6.3 et 6.6) :

```
import: from <peering-1> [action <action-1>]
      .
      .
      .
      from <peering-N> [action <action-N>]
      accept <filter>
```

La spécification de l'action est facultative. La sémantique d'un attribut import est la suivante : l'ensemble des chemins qui satisfont à <filtre> est importé de tous les homologues dans <peerings> ; lors de l'importation de chemins à <peering-M>, l'<action-M> est exécutée.

Par exemple,

```
aut-num: AS1
import: from AS2 action pref = 1; accept { 128.9.0.0/16 }
```

Cet exemple déclare que le chemin 128.9.0.0/16 est accepté de AS2 avec la préférence 1. On a déjà présenté comment sont spécifiés les appariements (paragraphe 5.6) et les filtres (paragraphe 5.4). On présente maintenant comment spécifier les actions.

6.1.1 Spécification d'action

Dans RPSL, les actions de politique établissent ou modifient des attributs de chemins, en allouant une préférence à un chemin, en ajoutant une communauté BGP à l'attribut de chemin de communauté BGP, ou en établissant l'attribut MULTI-EXIT-DISCRIMINATOR. Les actions de politique peuvent aussi donner pour instruction aux routeurs d'effectuer des opérations particulières, telles que l'amortissement des fluctuations de chemin.

Les attributs de politique d'acheminement dont les valeurs peuvent être modifiées dans les actions de politique sont spécifiés dans le dictionnaire RPSL. Prière de se référer à la Section 7 pour une liste de ces attributs. Chaque action dans RPSL est terminée par le caractère deux-points (';'). Il est possible de former des actions de politique composites en les énumérant les unes à la suite des autres. Dans une action de politique composite, les actions sont exécutées de gauche à droite. Par exemple,

```
aut-num: AS1
import: from AS2
      action pref = 10; med = 0; community.append(10250, 3561:10);
      accept { 128.9.0.0/16 }
```

établit la préférence à 10, med à 0, et ensuite ajoute 10250 et 3561:10 à l'attribut de chemin de communauté BGP. L'attribut pref est l'inverse de l'attribut local-pref (c'est-à-dire que local-pref == 65535 - pref). Un chemin avec un attribut local-pref est toujours préféré à un chemin qui n'en a pas.

```
aut-num: AS1
import: from AS2 action pref = 1;
      from AS3 action pref = 2;
      accept AS4
```

L'exemple ci-dessus déclare que les chemins de AS4 sont acceptés de AS2 avec la préférence 1, et de AS3 avec la

préférence 2 (les chemins avec des valeurs de préférence inférieures sont préférés aux chemins qui ont des valeurs de préférence plus élevées).

```
aut-num: AS1
import: from AS2 7.7.7.2 at 7.7.7.1 action pref = 1;
       from AS2          action pref = 2;
       accept AS4
```

L'exemple ci-dessus déclare que les chemins de AS4 sont acceptés de AS2 sur l'appariement 7.7.7.1-7.7.7.2 avec la préférence 1, et sur tout autre appariement avec AS2 avec la préférence 2.

6.2 Attribut export : spécification de politique d'exportation

De même, une expression de politique d'exportation est spécifiée en utilisant un attribut export. L'attribut export a la syntaxe suivante :

```
export: to <peering-1> [action <action-1>]
       . . .
       to <peering-N> [action <action-N>] announce <filter>
```

La spécification de l'action est facultative. La sémantique d'un attribut export est la suivante : l'ensemble des chemins qui est satisfait par <filter> est exporté à tous les homologues spécifiés dans <peerings> ; lors de l'exportation de chemins à <peering-M>, <action-M> est exécuté.

Par exemple,

```
aut-num: AS1
export: to AS2 action med = 5; community . = { 70 }; announce AS4
```

Dans cet exemple, les chemins d'AS4 sont annoncés à AS2 avec la valeur de l'attribut med réglée à 5 et community 70 est ajouté à la liste de communautés.

Exemple :

```
aut-num: AS1
export: to AS-FOO announce ANY
```

Dans cet exemple, AS1 annonce tous ses chemins aux AS dans l'ensemble AS-FOO.

6.3 Autres protocoles d'acheminement, protocole d'acheminement multi-protocoles et injection de chemins entre les protocoles

La syntaxe complète des attributs import et export est la suivante :

```
import: [protocol <protocol-1>] [into <protocol-2>]
       from <peering-1> [action <action-1>]
       . . .
       from <peering-N> [action <action-N>]
       accept <filter>
export: [protocol <protocol-1>] [into <protocol-2>]
       to <peering-1> [action <action-1>]
       . . .
       to <peering-N> [action <action-N>]
       announce <filter>
```

Lorsque les spécifications facultatives de protocole peuvent être utilisées pour spécifier des politiques pour d'autres protocoles d'acheminement, ou pour injecter des chemins d'un protocole dans un autre protocole, ou pour des politiques d'acheminement multi-protocoles. Les noms de protocoles valides sont définis dans le dictionnaire. Le <protocol-1> est le nom du protocole dont les chemins sont échangés. Le <protocol-2> est le nom du protocole qui reçoit ces chemins. <protocol-1> et <protocol-2> prennent tous deux la valeur par défaut du protocole de routeur Internet extérieur, actuellement BGP.

Dans l'exemple suivant, tous les chemins interAS sont injectés dans RIP.

```

aut-num: AS1
import: from AS2 accept AS2
export: protocol BGP4 into RIP to AS1 announce ANY

```

Dans l'exemple suivant, AS1 accepte les chemins de AS2 y compris tous chemins plus spécifique de AS2, mais n'injecte pas ces chemins plus spécifiques supplémentaires dans OSPF.

```

aut-num: AS1
import: from AS2 accept AS2^+
export: protocol BGP4 into OSPF to AS1 announce AS2

```

Dans l'exemple qui suit, AS1 injecte ses chemins statiques (chemins qui sont membres de l'ensemble AS1:RS-STATIC-ROUTES) dans le protocole d'acheminement interAS et ajoute deux fois AS1 à leurs chemins d'AS.

```

aut-num: AS1
import: protocol STATIC into BGP4
      from AS1 action aspath.prepend(AS1, AS1); accept AS1:RS-STATIC-ROUTES

```

Dans l'exemple suivant, AS1 importe différents ensembles de chemins d'envoi individuel pour un chemin inverse de transmission en diffusion groupée provenant de AS2:

```

aut-num: AS1
import: from AS2 accept AS2
import: protocol IDMR from AS2 accept AS2:RS-RPF-ROUTES

```

6.4 Résolution d'ambiguïté

Il est possible que le même appariement puisse être couvert par plus d'une spécification d'appariement dans une expression de politique. Par exemple :

```

aut-num: AS1
import: from AS2 7.7.7.2 at 7.7.7.1 action pref = 2;
      from AS2 7.7.7.2 at 7.7.7.1 action pref = 1; accept AS4

```

Ce n'est pas une erreur, bien que très déconseillé. Pour dissiper l'ambiguïté, on utilise l'action correspondant à la première spécification d'appariement. C'est-à-dire que les chemins sont acceptés avec la préférence 2. On appelle cette règle specification-order.

Considérons l'exemple :

```

aut-num: AS1
import: from AS2          action pref = 2;
      from AS2 7.7.7.2 at 7.7.7.1 action pref = 1; dpa = 5; accept AS4

```

où les deux spécifications d'appariement couvrent l'appariement 7.7.7.1-7.7.7.2, bien que la seconde le couvre plus spécifiquement. La règle d'ordre de spécification s'applique quand même, et seule l'action "pref = 2" est exécutée. En fait, la seconde paire peering-action n'est pas utile car la première paire peering-action la couvre toujours. Si la politique désirée était d'accepter ces chemins avec la préférence 1 sur cet appariement particulier et avec la préférence 2 dans tous les autres appariements, l'utilisateur aurait dû spécifier :

```

aut-num: AS1
import: from AS2 7.7.7.2 at 7.7.7.1 action pref = 1; dpa = 5;
      from AS2          action pref = 2; accept AS4

```

Il est aussi possible que plus d'une expression de politique puisse couvrir le même ensemble de chemins pour le même appariement. Par exemple :

```

aut-num: AS1
import: from AS2 action pref = 2; accept AS4
import: from AS2 action pref = 1; accept AS4

```

Dans ce cas, la règle d'ordre de spécification est encore utilisée. C'est à dire que les chemins d'AS4 sont acceptés d'AS2 avec la préférence 2. Si les filtres se recouvraient sans être exactement les mêmes :

```
aut-num: AS1
import: from AS2 action pref = 2; accept AS4
import: from AS2 action pref = 1; accept AS4 OR AS5
```

Les chemins d'AS4 sont acceptés de AS2 avec la préférence 2 et cependant les chemins d'AS5 sont aussi acceptés, mais avec la préférence 1.

On donne ensuite la règle générale d'ordre de spécification pour les mises en œuvre de RPSL. Considérons deux expressions de politique :

```
aut-num: AS1
import: from peerings-1 action action-1 accept filter-1
import: from peerings-2 action action-2 accept filter-2
```

Ces expressions de politique sont équivalentes aux trois expressions suivantes dans lesquelles il n'y a pas d'ambiguïté :

```
aut-num: AS1
import: from peerings-1 action action-1 accept filter-1
import: from peerings-3 action action-2 accept filter-2 AND NOT filter-1
import: from peerings-4 action action-2 accept filter-2
```

et peerings-3 sont ceux qui sont couverts par les deux peerings-1 et peerings-2, et les peerings-4 sont ceux qui sont couverts par peerings-2 mais pas par peerings-1 ("filter-2 AND NOT filter-1" correspond aux chemins qui sont satisfaits par le filter-2 mais pas par le filter-1).

Exemple :

```
aut-num: AS1
import: from AS2 7.7.7.2 at 7.7.7.1
      action pref = 2;
      accept {128.9.0.0/16}
import: from AS2
      action pref = 1;
      accept {128.9.0.0/16, 75.0.0.0/8}
```

Considérons deux appariements avec AS2, 7.7.7.1-7.7.7.2 et 9.9.9.1-9.9.9.2. Les deux expressions de politique couvrent 7.7.7.1-7.7.7.2. Sur cet appariement, le chemin 128.9.0.0/16 est accepté avec la préférence 2, et le chemin 75.0.0.0/8 est accepté avec la préférence 1. L'appariement 9.9.9.1-9.9.9.2 est seulement couvert par la seconde expression de politique. Donc, les deux chemins 128.9.0.0/16 et 75.0.0.0/8 sont acceptés avec la préférence 1 sur l'appariement 9.9.9.1-9.9.9.2.

Noter que la même règle de résolution d'ambiguïté s'applique aussi aux expressions de politique d'exportation et par défaut.

6.5 Attribut par défaut : spécification de politique par défaut

Les politiques d'acheminement pas défaut sont spécifiées en utilisant l'attribut default. L'attribut default a la syntaxe suivante :

```
default: to <peering> [action <action>] [networks <filter>]
```

Les spécifications <action> et <filter> sont facultatives. Leur sémantique est la suivante : la spécification <peering> indique que l'AS (et le routeur si il est présent) sont pris par défaut ; la spécification <action>, si elle est présente, indique divers attributs par défaut, par exemple une préférence relative si plusieurs valeurs par défaut sont spécifiées; et la spécification <filter>, si elle est présente, est un filtre de politique. Un routeur n'utilise de politique par défaut que si il a reçu les chemins satisfaits par <filter> provenant de cet homologue.

Dans l'exemple suivant, AS1 se replie par défaut sur AS2 pour l'acheminement.

```
aut-num: AS1
default: to AS2
```

Dans l'exemple suivant, le routeur 7.7.7.1 dans AS1 revient par défaut au routeur 7.7.7.2 dans AS2.

```
aut-num: AS1
default: to AS2 7.7.7.2 at 7.7.7.1
```

Dans l'exemple suivant, AS1 revient par défaut à AS2 et AS3, mais préfère AS2 à AS3.

```
aut-num: AS1
default: to AS2 action pref = 1;
default: to AS3 action pref = 2;
```

Dans l'exemple suivant, AS1 revient par défaut à AS2 et utilise 128.9.0.0/16 comme réseau par défaut.

```
aut-num: AS1
default: to AS2 networks { 128.9.0.0/16 }
```

6.6 Spécification de politique structurée

Les politiques d'importation et d'exportation peuvent être structurées. On ne recommande les politiques structurées qu'aux utilisateurs RPSL avertis. Sinon n'hésitez pas à sauter ce paragraphe.

La syntaxe pour une spécification de politique structurée est la suivante :

```
<import-factor> ::= from <peering-1> [action <action-1>]
...
from <peering-N> [action <action-N>]
accept <filter>;
```

```
<import-term> ::= <import-factor> |
LEFT-BRACE
<import-factor>
...
<import-factor>
RIGHT-BRACE
```

```
<import-expression> ::= <import-term> | <import-term> EXCEPT <import-expression> | <import-term> REFINE
<import-expression>
```

```
import: [protocol <protocol1>] [into <protocol2>] <import-expression>
```

Bien noter le point virgule à la fin d'un <import-factor>. Si la spécification de politique n'est pas structurée (comme dans tous les exemples des autres paragraphes) ce point virgule est facultatif. La syntaxe et la sémantique d'un <import-factor> est déjà définie au paragraphe 6.1.

Un <import-term> est soit une séquence de <import-factor> inclus entre des accolades (c'est-à-dire '{' et '}') soit juste un seul <import-factor>. La sémantique d'un <import-term> est l'union des <import-factor> en utilisant la règle d'ordre de spécification. Un <import-expression> est soit un seul <import-term> soit un <import-term> suivi par un des mots clés "except" et "refine", suivi par un autre <import-expression>. Noter que notre définition permet des expressions incorporées. Donc, il peut y avoir des exceptions aux exceptions, des raffinements aux raffinements, ou même des raffinements aux exceptions, et ainsi de suite.

La sémantique de l'opérateur except est la suivante : le résultat d'une opération except est un autre <import-term>. L'ensemble de politique résultant contient les politiques du côté droit mais leurs filtres sont modifiés de façon à n'inclure que les chemins aussi satisfaits par le côté gauche. Les politiques du côté gauche sont incluses ensuite et leurs filtres sont modifiés pour exclure les chemins satisfaits par le côté droit. Prière de noter que les filtres sont modifiés durant ce processus mais les actions sont copiées verbatim. Lorsque il y a plusieurs niveaux d'incorporation, les opérations (except et refine) sont effectuées de droite à gauche.

Considérons l'exemple suivant :

```
import: from AS1 action pref = 1; accept as-foo;
except {
  from AS2 action pref = 2; accept AS226;
  except {
```

```

    from AS3 action pref = 3; accept {128.9.0.0/16};
  }
}

```

où le chemin 128.9.0.0/16 est généré par AS226, et AS226 est un membre de l'ensemble d'AS as-foo. Dans cet exemple, le chemin 128.9.0.0/16 est accepté de AS3, tout autre chemin (pas 128.9.0.0/16) généré par AS226 est accepté de AS2, et tout autre chemin des AS dans as-foo est accepté de AS1.

On peut arriver à la même conclusion en utilisant l'algèbre définie ci-dessus. Considérons la spécification d'exception interne :

```

from AS2 action pref = 2; accept AS226;
except {
  from AS3 action pref = 3; accept {128.9.0.0/16};
}

```

elle est équivalente à

```

from AS3 action pref = 3; accept AS226 AND {128.9.0.0/16};
from AS2 action pref = 2; accept AS226 AND NOT {128.9.0.0/16};

```

Donc, l'expression d'origine est équivalente à :

```

import: from AS1 action pref = 1; accept as-foo;
except {
  from AS3 action pref = 3; accept AS226 AND {128.9.0.0/16};
  from AS2 action pref = 2; accept AS226 AND NOT {128.9.0.0/16};
}

```

qui est équivalente à

```

import: {
  from AS3 action pref = 3;
  accept as-foo AND AS226 AND {128.9.0.0/16};
  from AS2 action pref = 2;
  accept as-foo AND AS226 AND NOT {128.9.0.0/16};
  from AS1 action pref = 1;
  accept as-foo AND NOT
  (AS226 AND NOT {128.9.0.0/16} OR AS226 AND {128.9.0.0/16});
}

```

Comme AS226 est dans as-foo et que 128.9.0.0/16 est dans AS226, cela se simplifie en :

```

import: {
  from AS3 action pref = 3; accept {128.9.0.0/16};
  from AS2 action pref = 2; accept AS226 AND NOT {128.9.0.0/16};
  from AS1 action pref = 1; accept as-foo AND NOT AS226;
}

```

Dans le cas de l'opérateur refine, l'ensemble résultant est construit en prenant le produit cartésien des deux côtés comme suit : pour chaque politique l du côté gauche et pour chaque politique r du côté droit, les appariements de politique résultants sont les appariements communs à r et l ; le filtre de la politique résultante est l'intersection du filtre de l et du filtre de r ; et l'action de la politique résultante est l'action de l suivie de l'action de r. Si il n'y a pas d'appariements communs, ou si l'intersection des filtres est vide, il n'est pas généré de politique résultante.

Considérons l'exemple suivant :

```

import: { from AS-ANY action pref = 1; accept community(3560:10);
  from AS-ANY action pref = 2; accept community(3560:20);
} refine {
  from AS1 accept AS1;
  from AS2 accept AS2;
  from AS3 accept AS3;
}

```

```
}

```

Ici, tout chemin avec une communauté de 3560:10 reçoit une préférence de 1 et tout chemin avec une communauté de 3560:20 reçoit une préférence de 2 sans considérer de qui ils sont importés. Cependant, seuls les chemins de AS1 sont importés de S1, et seuls les chemins de AS2 sont importés de AS2, et seuls les chemins de AS3 sont importés de AS3, et aucun chemin n'est importé d'aucun autre AS. On peut arriver à la même conclusion en utilisant l'algèbre ci-dessus. C'est-à-dire que notre exemple est équivalent à :

```
import: {
  from AS1 action pref = 1; accept community(3560:10) AND AS1;
  from AS1 action pref = 2; accept community(3560:20) AND AS1;
  from AS2 action pref = 1; accept community(3560:10) AND AS2;
  from AS2 action pref = 2; accept community(3560:20) AND AS2;
  from AS3 action pref = 1; accept community(3560:10) AND AS3;
  from AS3 action pref = 2; accept community(3560:20) AND AS3;
}
```

Noter que les appariements communs entre "from AS1" et "from AS-ANY" sont les appariements dans "from AS1". Bien qu'on n'ait pas défini formellement "appariements communs", on déduit directement la définition de celles des appariements (voir le paragraphe 5.6).

Considérons l'exemple suivant :

```
import: {
  from AS-ANY action med = 0; accept {0.0.0.0/0^0-18};
} refine {
  from AS1 at 7.7.7.1 action pref = 1; accept AS1;
  from AS1      action pref = 2; accept AS1;
}
```

où seuls les chemins de longueur 0 à 18 sont acceptés et la valeur de med est réglée à 0 pour désactiver l'effet de med pour tous les appariements ; de plus, de AS1 seuls les chemins de AS1 sont pris en charge, et les chemins de AS1 importés à 7.7.7.1 sont préférés aux autres appariements. Cela est équivalent à :

```
import: {
  from AS1 at 7.7.7.1 action med=0; pref=1; accept {0.0.0.0/0^0-18} AND AS1;
  from AS1      action med=0; pref=2; accept {0.0.0.0/0^0-18} AND AS1;
}
```

La syntaxe et la sémantique ci-dessus s'appliquent aussi aux politiques d'exportation structurées avec "from" remplacé par "to" et "accept" remplacé par "announce".

7 Classe dictionary

La classe dictionary permet l'extensibilité de RPSL. Les objets dictionary définissent les attributs, types, et protocoles d'acheminement de politique d'acheminement. Les attributs de politique d'acheminement, qu'on appellera rp-attributs, peuvent correspondre à des attributs réels du protocole, tels que les attributs de chemin BGP (par exemple, community, dpa, et AS-path) ou ils peuvent correspondre aux caractéristiques du routeur (par exemple, amortissement de fluctuations de chemin BGP). Lorsque de nouveaux protocoles, de nouveaux attributs de protocole, ou de nouvelles caractéristiques de routeur sont introduits, l'objet dictionary est mis à jour pour inclure les rp-attribut et les définitions de protocole appropriées.

Un rp-attribut est une classe abstraite ; c'est-à-dire qu'une représentation de données n'est pas disponible. À la place, on y accède par des méthodes d'accès. Par exemple, le rp-attribut pour l'attribut BGP AS-path est appelé aspath, et il a une méthode d'accès appelée prepend qui bourne de numéros d'AS supplémentaires les attributs AS-path. Les méthodes d'accès peuvent prendre des arguments. Les arguments sont fortement typés. Par exemple, la méthode prepend ci-dessus prend des numéros d'AS comme arguments.

Une fois qu'un rp-attribut est défini dans le dictionnaire, il peut être utilisé pour décrire les filtres et les actions de politique. Les outils d'analyse de politique sont nécessaires pour aller chercher l'objet dictionary et reconnaître les nouveaux rp-attributs, types, et protocoles définis. Les outils d'analyse peuvent approximer les analyses de politique sur

les rp-attributs qu'ils ne comprennent pas : une méthode filter peut toujours correspondre, et une méthode d'action peut toujours effectuer no-operation. Les outils d'analyse peuvent même charger du code pour effectuer les opérations appropriées en utilisant des mécanismes qui sortent du domaine d'application de RPSL.

On décrit ensuite la syntaxe et la sémantique de la classe dictionary. Cette description n'est pas essentielle pour comprendre les objets dictionary (mais elle est essentielle pour en créer un). On peut sauter le paragraphe sur le dictionnaire RPSL initial (paragraphe 7.1).

Les attributs de la classe dictionary sont présentés à la Figure 24. L'attribut dictionary est le nom de l'objet dictionary, obéissant aux règles de désignation de RPSL. Il peut y avoir beaucoup d'objets dictionary, cependant, il y a toujours un objet dictionary bien connu, "RPSL". tous les outils utilisent ce dictionnaire par défaut.

Attribut	Valeur	Type
dictionary	<nom-d'objet>	obligatoire, une seule valeur, clé de classe
rp-attribute	voir la description dans le texte	facultatif, multi valeurs
typedef	voir la description dans le texte	facultatif, multi valeurs
protocol	voir la description dans le texte	facultatif, multi valeurs

Figure 24 : Attributs de la classe dictionary

L'attribut rp-attribute a la syntaxe suivante :

```
rp-attribute: <name> <method-1>(<type-1-1>, ..., <type-1-N1> [, "..."])
...
<method-M>(<type-M-1>, ..., <type-M-NM> [, "..."])
```

où <name> est le nom du rp-attribute, et <method-i> est le nom d'une méthode d'accès pour le rp-attribute, prenant Ni arguments où le j^{ème} argument est du type <type-i-j>. Un nom de méthode est soit un nom RPSL soit un des opérateurs définis à la Figure 25. Les méthodes d'opérateur à l'exception de operator() et operator[] peuvent prendre seulement un argument.

```
operator=      operator==
operator<<=   operator<
operator>>=   operator>
operator+=    operator>=
operator-=    operator<=
operator*=    operator!=
operator/=    operator()
operator.=    operator[]
```

Figure 25 : Opérateurs

Un rp-attribute peut avoir de nombreuses méthodes définies. Certaines des méthodes peuvent même avoir le même nom, auquel cas leurs arguments sont de types différents. Si la liste des arguments est suivie par "...", la méthode prend un nombre variable d'arguments. Dans ce cas, les arguments réels après le N^{ème} argument sont du type <type-N>.

Les arguments sont fortement typés. Un <type> dans RPSL est un type prédéfini, un type union, un type liste, ou un type défini dans le dictionnaire. Les types prédéfinis sont énumérés à la Figure 26.

```
integer[lower, upper]   ipv4_address
real[lower, upper]     address_prefix
enum[name, name, ...]  address_prefix_range
string                  dns_name
boolean                 filter
rpsl_word               as_set_name
free_text               route_set_name
email                   rtr_set_name
as_number               filter_set_name
                        peering_set_name
```

Figure 26 : Types prédéfinis

Les types prédéfinis integer et real peuvent être suivis par une limite inférieure et supérieure pour spécifier l'ensemble des valeurs valides de l'argument. La spécification de la gamme est facultative. On utilise les conventions du langage C de l'ANSI pour représenter les valeurs d'entier, de réel et de chaîne. Le type enum est suivi par une liste de noms RPSL qui

sont les valeurs valides du type. Le type booléen peut prendre les valeurs vrai ou faux. Les types `as_number`, `ipv4_address`, `address_prefix` et `dns_name` sont comme à la Section 2. Le type `filter` est un filtre de politique comme à la Section 6. Il est suggéré que les valeurs du type `filter` soient incluses entre parenthèses.

La syntaxe d'un type union est la suivante :

```
union <type-1>, ... , <type-N>
```

où `<type-i>` est un type RPSL. Le type union est l'un ou l'autre des types `<type-1>` à `<type-N>` (analogue aux unions dans C [LANG-C]).

La syntaxe du type list est la suivante :

```
list [<min_elems>:<max_elems>] of <type>
```

Dans ce cas, les éléments de la liste de `<type>` et la liste contiennent au moins `<min_elems>` et au plus `<max_elems>` éléments. La spécification `size` est facultative. Si elle n'est pas spécifiée, il n'y a pas de restriction au nombre des éléments de la liste. Une valeur d'un type list est représentée par une séquence d'éléments séparés par le caractère "," et est incluse entre les caractères "{" et "}".

L'attribut `typedef` dans le dictionnaire définit des types désignés comme suit :

```
typedef: <nom> <type>
```

où `<nom>` est un nom pour le type `<type>`. L'attribut `typedef` est particulièrement utile lorsque le type défini n'est pas un type prédéfini (par exemple liste d'unions, liste de listes, etc.).

Un attribut `protocol` de la classe `dictionary` définit un protocole et un ensemble de paramètres d'appariement pour ce protocole (qui sont utilisés dans la classe `inet-rtr` à la Section 9). Sa syntaxe est la suivante :

```
protocol: <nom>
  OBLIGATOIRE | FACULTATIF <parameter-1>(<type-1-1>, ..., <type-1-N1> [, "..."])
  ...
  OBLIGATOIRE | FACULTATIF <parameter-M>(<type-M-1>, ..., <type-M-NM> [, "..."])
```

Où `<nom>` est le nom du protocole ; `OBLIGATOIRE` et `FACULTATIF` sont des mots clés ; et `<parameter-i>` est un paramètre d'appariement pour ce protocole, qui prend `Ni` arguments. La syntaxe et la sémantique des arguments sont comme dans `rp-attribute`. Si le mot clé `OBLIGATOIRE` est utilisé, le paramètre est obligatoire et doit être spécifié pour chaque appariement de ce protocole. Si le mot clé `FACULTATIF` est utilisé, le paramètre peut être sauté.

7.1 Dictionnaire RPSL initial et exemple d'actions et de filtres de politique

dictionnaire : RPSL

`rp-attribute` : # préférence, les plus petites valeurs représentent les plus grandes préférences

`pref`

opérateur=(entier[0, 65535])

`rp-attribute` :

`med`

attribut BGP multi_exit_discriminator

pour régler med à 10 : med = 10;

pour régler med à la métrique IGP : med = igp_cost;

opérateur=(union entier[0, 65535], enum[igp_cost])

`rp-attribute`: # attribut BGP de préférence de destination (dpa)

`dpa`

opérateur=(entier[0, 65535])

`rp-attribute`:

`aspath`

attribut BGP aspath

ajouter les numéros d'AS dans l'ordre du dernier au premier

`prepend(as_number, ...)`

`typedef`:

une valeur de communauté dans RPSL est soit :

- un entier de 4 octets (la notation 3561:70 est acceptable)

- internet, no_export, no_advertise (voir la [RFC1997])

`community_elm union`

```

    entier[1, 4294967295],
    enum[internet, no_export, no_advertise],
typedef:
    community_list liste de community_elm
rp-attribute:
    community
    # liste de valeurs de communauté { 40, no_export, 3561:70 }

    opérateur=(community_list)
    # attribut BGP community
    # réglé à une liste de communautés
    # ajouter les valeurs de community

    opérateur=(community_list)
    append(community_elm, ...)
    # supprimer les valeurs de community

    delete(community_elm, ...)
    # un filtre : vrai si une des valeurs de community est contenue
    # raccourci pour contenir : community(no_export, 3561:70)

    operator()(community_elm, ...)
    # comparaison d'égalité indépendante de l'ordre

    operator==(community_list)
    # routeur de prochain bond dans un chemin statique
rp-attribute:
    next-hop
    # pour régler à 7.7.7.7 : next-hop = 7.7.7.7;
    # pour régler à la propre adresse du routeur : next-hop = self;

    operator=(union ipv4_address, enum[self])
    # coût d'un chemin statique
rp-attribute:
    cost
    opérateur=(entier[0, 65535])
protocol: BGP4
    # numéro d'AS du routeur homologue

    OBLIGATOIRE asno(as_number)
    # ACTIVE l'amortissement de fluctuations (flap damping)

    FACULTATIF flap_damp()
    FACULTATIF flap_damp(entier[0,65535],
        entier[0,65535],
        # pénalité par fluctuation
        entier[0,65535],
        # valeur de la pénalité pour la suppression
        entier[0,65535],
        # valeur de la pénalité pour la réutilisation
        entier[0,65535],
        # demie vie en secondes lorsque ouverte
        entier[0,65535],
        # demis vie en secondes lorsque fermée
        entier[0,65535])
        # pénalité maximum

protocol: OSPF
protocol: RIP
protocol: IGRP
protocol: IS-IS
protocol: STATIC
protocol: RIPng
protocol: DVMRP
protocol: PIM-DM
protocol: PIM-SM
protocol: CBT
protocol: MOSPF

```

Figure 27 : Dictionnaire RPSL

La Figure 27 montre le dictionnaire initial RPSL. Il a sept rp-attributes : pref pour allouer la préférence locale aux chemins acceptés ; med pour allouer une valeur à l'attribut BGP MULTI_EXIT_DISCRIMINATOR ; dpa pour allouer une valeur à l'attribut BGP DPA ; aspath pour ajouter une valeur à l'attribut BGP AS_PATH ; community pour allouer une valeur ou pour vérifier la valeur de l'attribut BGP community ; next-hop pour allouer les routeurs de prochain bond à des chemins statiques ; et cost pour allouer un coût aux chemins statiques. Le dictionnaire définit deux types : community_elm et

community_list. Le type community_elm est soit un entier non signé de quatre octets, soit un des mots clés internet, no_export ou no_advertise (définis dans la [RFC1997]). Un entier peut être spécifié en utilisant des entiers de deux octets séparés par ":" pour partager l'espace de numéros de la communauté afin qu'un fournisseur puisse utiliser son numéro d'AS comme les deux premiers octets, et allouer une sémantique de son choix aux deux derniers octets.

Le dictionnaire initial de la Figure 27 définit seulement les options pour le protocole de routeur frontière 'BGP : asno et flap_damp. L'option asno obligatoire est le numéro d'AS du routeur homologue. L'option facultative flap_damp donne pour instruction au routeur d'amortir les fluctuations de chemin [RFC2439] lorsque il importe des chemins du routeur homologue.

Il peut être spécifié avec ou sans paramètre. Si des paramètres manquent, ils prennent par défaut :

```
flap_damp(1000, 2000, 750, 900, 900, 20000)
```

C'est-à-dire qu'une pénalité de 1000 est allouée à chaque fluctuation de chemin, le chemin est supprimé lorsque la pénalité atteint 2000. La pénalité est réduite de moitié après 15 minutes (900 secondes) de stabilité sans considération de si le chemin est ouvert ou fermé. Un chemin supprimé est réutilisé lorsque la pénalité tombe en dessous de 50. La pénalité maximum qui peut être allouée à un chemin est de 20 000 (c'est-à-dire que le temps de suppression maximum après qu'un chemin est devenu stable est d'environ 75 minutes). Ces paramètres sont cohérents avec les paramètres d'amortissement de fluctuation par défaut dans plusieurs routeurs.

Actions et filtres de politique utilisant de sRP-Attribute

La syntaxe d'une action ou filtre de politique qui utilisé un rp-attribute x est la suivante :

```
x.method(arguments)
x "op" argument
```

où method est une méthode et "op" est une méthode d'opérateur du rp-attribute x. Si une méthode d'opérateur est utilisée pour spécifier un filtre composite de politique, elle évalue avant les opérateurs de filtre composite de politique (c'est-à-dire, AND, OR, NOT, et implicit ou operator).

Le rp-attribute pref peut recevoir une valeur d'entier positif comme suit :

```
pref = 10;
```

Le rp-attribute med peut recevoir une valeur d'entier positif ou le mot "igp_cost" comme suit :

```
med = 0;
med = igp_cost;
```

Le rp-attribute dpa peut recevoir une valeur d'entier positif comme suit :

```
dpa = 100;
```

L'attribut BGP community est une valeur de liste, c'est à dire que c'est une liste d'entiers de quatre octets dont chacun représente une "communauté". Les exemples suivants montrent comment ajouter des communautés à ce rp-attribute :

```
community . = { 100 };
community . = { NO_EXPORT };
community . = { 3561:10 };
```

Dans le dernier cas, un entier de 4 octets est construit où les deux octets de poids fort sont égaux à 3561 et les deux octets de moindre poids sont égaux à 10. Les exemples suivants montrent comment supprimer des communautés du rp-attribute community :

```
community.delete(100, NO_EXPORT, 3561:10);
```

Les filtres qui utilisent le rp-attribute community peuvent être définis comme montré par les exemples suivants :

```
community.contains(100, NO_EXPORT, 3561:10);
community(100, NO_EXPORT, 3561:10);      # shortcut
```

Le rp-attribute community peut être réglé à une liste de communautés comme suit :

```
community = {100, NO_EXPORT, 3561:10, 200};
community = {};
```

Dans le premier cas, le rp-attribute community contient les communautés 100, NO_EXPORT, 3561:10, et 200. Dans le second cas, le rp-attribute community est supprimé. Le rp-attribute community peut être comparé à une liste de communautés comme suit :

```
community == {100, NO_EXPORT, 3561:10, 200};           # correspondance exacte
```

Pour influencer le choix de chemin, le rp-attribute BGP as_path peut être allongé en y ajoutant devant les numéros d'AS comme suit :

```
aspath.prepend(AS1);
aspath.prepend(AS1, AS1, AS1);
```

Les exemples suivants sont invalides :

```
med = -50;                                           # -50 n'est pas dans la gamme
med = igp;                                           # igp n'est pas une des valeur de enum
med.assign(10);                                       # method assign n'est pas défini
community.append(AS3561:20);                         # le premier argument devrait être 3561
```

La Figure 28 montre un exemple plus évolué utilisant le rp-attribute community. Dans cet exemple, AS3561 fonde sa préférence de choix de chemin sur l'attribut community. D'autres AS peuvent affecter indirectement le choix de chemin de AS3561 en incluant les communautés appropriées dans leurs annonces de chemin.

```
aut-num: AS1
export: to AS2 action community={3561:90};
       to AS3 action community={3561:80};
announce AS1
```

```
as-set: AS3561:AS-PEERS
members: AS2, AS3
```

```
aut-num: AS3561
import: from AS3561:AS-PEERS
       action pref = 10;
       accept community(3561:90)
import: from AS3561:AS-PEERS
       action pref = 20;
       accept community(3561:80)
import: from AS3561:AS-PEERS
       action pref = 20;
       accept community(3561:70)
import: from AS3561:AS-PEERS
       action pref = 0;
       accept ANY
```

Figure 28 : Exemple de politique utilisant le rp-attribute community

8. Classe Advanced route

8.1 Spécification de chemins agrégés

Les composants aggr-bndry, aggr-mtd, export-comps, inject, et l'attribut holes sont utilisés pour spécifier des chemins agrégés [RFC1519]. Un objet route spécifie un chemin agrégé si un de ces attributs, à l'exception de inject, est spécifié. L'attribut origin pour un chemin agrégé est l'AS qui effectue l'agrégation, c'est-à-dire, l'AS agrégateur. Dans ce paragraphe, on a utilisé le terme "agrégat" pour se référer au chemin généré, le terme "composant" pour se référer aux chemins utilisés pour générer les attributs de chemin de l'agrégat, et le terme "plus spécifique" pour se référer à tout chemin qui est le plus spécifique de l'agrégat sans considération de si il a été utilisé pour former les attributs du chemin.

L'attribut composants définit quels chemins composants sont utilisés pour former l'agrégat. Sa syntaxe est la suivante :

```
composants: [ATOMIC] [[<filtre>] [protocole <protocole> <filtre> ...]]
```

où <protocole> est un nom de protocole d'acheminement tel que BGP4, OSPF ou RIP (les noms valides sont définis dans le dictionnaire) et <filtre> est une expression de politique. Les chemins qui satisfont à un de ces filtres et sont appris du protocole correspondant sont utilisés pour former l'agrégat. Si <protocole> est omis, il prend par défaut n'importe quel protocole. <filtre> contient implicitement un terme "AND" avec le plus spécifique de l'agrégat de sorte que seuls les chemins composants sont choisis. Si le mot clé ATOMIC est utilisé, l'agrégation est faite atomiquement [RFC1519]. Si un <filtre> n'est pas spécifié, il prend par défaut le plus spécifique. Si l'attribut composants manque, tout le plus spécifique sans le mot clé ATOMIC est utilisé.

```
route: 128.8.0.0/15
origin: AS1
components: <^AS2>

route: 128.8.0.0/15
origin: AS1
components: protocol BGP4 {128.8.0.0/16^+}
              protocol OSPF {128.9.0.0/16^+}
```

Figure 29 : Deux objets route agrégés

La Figure 29 montre deux objets route. Dans le premier exemple, le plus spécifique de 128.8.0.0/15 avec les chemins d'AS commençant par AS2 sont agrégés. Dans le second exemple, des chemins appris de BGP et des chemins appris de OSPF sont agrégés.

L'attribut aggr-bndry est une expression d'AS sur des numéros et des ensembles d'AS (voir au paragraphe 5.6). Le résultat définit l'ensemble des AS qui forment la limite de l'agrégation. Si l'attribut aggr-bndry manque, l'AS d'origine est la seule limite de l'agrégation. En dehors de la limite d'agrégation, seul l'agrégat est exporté et les plus spécifiques sont supprimés. Cependant, au sein de la limite, les plus spécifiques sont aussi échangés.

L'attribut aggr-mtd spécifie comment l'agrégat est généré. Sa syntaxe est la suivante :

```
aggr-mtd: inbound
          | outbound [<as-expression>]
```

où <as-expression> est une expression sur les numéros et ensembles d'AS (voir au paragraphe 5.6). Si <as-expression> manque, elle prend par défaut AS-ANY. Si l'agrégation sortante (*outbound*) est spécifiée, les plus spécifiques de l'agrégat seront présents au sein de l'AS et l'agrégat sera formé à toutes les limites inter-AS avec les AS dans <as-expression> avant l'exportation, sauf pour les AS qui sont au sein de la limite de l'agrégation (c'est-à-dire que aggr-bndry est appliqué sans considération de <as-expression>). Si l'agrégation entrante est spécifiée, l'agrégat est formé à toutes les limites inter-AS avant d'importer des chemins dans l'AS agrégateur. Noter que <as-expression> peut n'être pas spécifié avec l'agrégation entrante. Si l'attribut aggr-mtd manque, il prend par défaut "outbound AS-ANY".

```
route: 128.8.0.0/15      route: 128.8.0.0/15
origin: AS1              origin: AS2
components: {128.8.0.0/15^-}  components: {128.8.0.0/15^-}
aggr-bndry: AS1 OR AS2      aggr-bndry: AS1 OR AS2
aggr-mtd: outbound AS-ANY    aggr-mtd: outbound AS-ANY
```

Figure 30 : Exemple d'agrégation multi-AS sortante

La Figure 30 montre un exemple d'agrégation sortante. Dans cet exemple, AS1 et AS2 coordonnent l'agrégation et annoncent seulement le moins spécifique 128.8.0.0/15 au monde extérieur, mais échangent les plus spécifiques entre eux. Cette forme d'agrégation est utile lorsque certains des composants sont dans AS1 et d'autres dans AS2.

Lorsque un ensemble de chemins est agrégé, l'intention est de n'exporter que le chemin agrégé et de supprimer l'exportation du plus spécifique en dehors de la limite d'agrégation. Cependant, pour satisfaire certaines contraintes de politique et de topologie (par exemple, un composant multi rattachements) il est souvent nécessaire d'exporter certains des composants. L'attribut export-comps est égal à un filtre RPSL qui satisfait au plus spécifique qui a besoin d'être exporté en dehors des limites de l'agrégation. Si cet attribut manque, les plus spécifiques ne sont pas exportés au dehors de la limite de

l'agrégation. Noter que le filtre export-comps contient un terme "AND" implicite avec le plus spécifique de l'agrégat.

La Figure 31 montre un exemple d'agrégation sortante. Dans cet exemple, le plus spécifique 128.8.8.0/24 est exporté hors de AS1 en plus de l'agrégat. C'est utile lorsque 128.8.8.0/24 est un site multi rattaché à AS1 avec quelques autres AS.

```
route: 128.8.0.0/15
origin: AS1
components: {128.8.0.0/15^-}
aggr-mtd: outbound AS-ANY
export-comps: {128.8.8.0/24}
```

Figure 31 : Agrégation sortante avec exception export

L'attribut inject spécifie quels routeurs effectuent l'agrégation et quand il l'effectuent. Sa syntaxe est la suivante :

```
inject: [at <router-expression>] ...
       [action <action>]
       [upon <condition>]
```

où <action> est une spécification d'action (voir au paragraphe 6.1.1), <condition> est une expression booléenne décrite plus loin, et <router-expression> est décrit au paragraphe 5.6.

Tous les routeurs dans <router-expression> et dans l'AS agrégateur effectuent l'agrégation. Si une <router-expression> n'est pas spécifiée, tous les routeurs à l'intérieur de l'AS agrégateur effectuent l'agrégation. La spécification <action> peut établir les attributs de chemin de l'agrégat, comme allouer des préférences à l'agrégat.

La clause upon est une condition booléenne. L'agrégat est généré si et seulement si cette condition est vraie. <condition> est une expression booléenne qui utilise les opérateurs logiques AND et OR (c'est-à-dire que l'opérateur NOT n'est pas permis) sur :

```
HAVE-COMPONENTS { liste de préfixes }
EXCLUDE { liste de préfixes }
STATIC
```

La liste des préfixes dans HAVE-COMPONENTS peut seulement être le plus spécifique de l'agrégat. Elle s'évalue à vrai lorsque tous les préfixes de la liste sont présents dans le tableau d'acheminement du routeur agrégateur. La liste peut aussi inclure des gammes de préfixes (c'est-à-dire utilisant les opérateurs ^-, ^+, ^n, et ^n-m). Dans ce cas, au moins un préfixe de chaque gamme de préfixe doit être présent dans le tableau d'acheminement pour que la condition soit vraie. La liste de préfixes dans EXCLUDE peut être arbitraire. Elle s'évalue à vrai lorsque aucun des préfixes de la liste n'est présent dans le tableau d'acheminement. La liste peut aussi inclure des gammes de préfixes, et aucun préfixe de cette gamme ne devrait être présent dans le tableau d'acheminement. Le mot clé static s'évalue toujours à vrai. Si aucune clause upon n'est spécifiée, l'agrégat est généré si et seulement si il y a un composant dans le tableau d'acheminement (c'est-à-dire un plus spécifique qui satisfait au filtre dans l'attribut components).

```
route: 128.8.0.0/15
origin: AS1
components: {128.8.0.0/15^-}
aggr-mtd: outbound AS-ANY
inject: at 1.1.1.1 action dpa = 100;
inject: at 1.1.1.2 action dpa = 110;

route: 128.8.0.0/15
origin: AS1
components: {128.8.0.0/15^-}
aggr-mtd: outbound AS-ANY
inject: upon HAVE-COMPONENTS {128.8.0.0/16, 128.9.0.0/16}
holes: 128.8.8.0/24
```

Figure 32 : Exemples de inject

La Figure 32 donne deux exemples. Dans le premier cas, l'agrégat est injecté à deux routeurs dont chacun établit différemment l'attribut dpa path. Dans le second cas, l'agrégat n'est généré que si les deux préfixes 128.8.0.0/16 et 128.9.0.0/16 sont présents dans le tableau d'acheminement, à la différence du premier cas où la présence d'un seul d'entre

eux est suffisante pour l'injection.

L'attribut holes fait la liste des préfixes d'adresse de composant qui ne sont pas accessibles par le chemin agrégé (peut-être que cette partie de l'espace d'adresses n'est pas allouée). L'attribut holes est utile pour les besoins de diagnostic. Dans la Figure 32, le second exemple a un trou, à savoir 128.8.8.0/24. Cela peut être dû à des usagers qui changent de fournisseur et qui emportent avec eux cette partie de l'espace d'adresses.

8.1.1 Interaction avec les politiques dans la classe aut-num

Un agrégat formé n'est annoncé aux autres AS que si les politiques d'export de l'AS permettent d'exporter l'agrégat. Lorsque l'agrégat est formé, les plus spécifiques sont supprimés de l'exportation sauf aux AS dans aggr-bndry et sauf les composants dans export-comps. Pour que se produisent de telles exceptions, les politiques d'exportation de l'AS devraient explicitement permettre l'exportation de ces exceptions.

Si un agrégat n'est pas formé (du fait de la clause upon) le plus spécifique de l'agrégat peut alors être exporté aux autres AS, mais seulement si la politique d'export de l'AS le permet. En d'autres termes, avant qu'un chemin (agrégé ou plus spécifique) soit exporté, il est filtré deux fois, une fois sur la base des objets route, et une fois sur la base de la politique d'export de l'AS.

```

route: 128.8.0.0/16
origin: AS1

route: 128.9.0.0/16
origin: AS1

route: 128.8.0.0/15
origin: AS1
aggr-bndry: AS1 or AS2 or AS3
aggr-mtd: outbound AS3 or AS4 or AS5
components: {128.8.0.0/16, 128.9.0.0/16}
inject: upon HAVE-COMPONENTS {128.9.0.0/16, 128.8.0.0/16}

aut-num: AS1
export: to AS2 announce AS1
export: to AS3 announce AS1 and not {128.9.0.0/16}
export: to AS4 announce AS1
export: to AS5 announce AS1
export: to AS6 announce AS1

```

Figure 33 : Interaction avec les politiques dans la classe aut-num

La Figure 33 montre un exemple d'interaction. En examinant les objets route, les plus spécifiques 128.8.0.0/16 et 128.9.0.0/16 devraient être échangés entre AS1, AS2 et AS3 (c'est-à-dire, les frontières d'agrégation). L'agrégation sortante est faite sur AS4 et AS5 et pas sur AS3, car AS3 est dans la limite d'agrégation. L'objet aut-num permet d'exporter les deux composants à AS2, mais seulement le composant 128.8.0.0/16 à AS3. L'agrégé peut seulement être formé si les deux composants sont disponibles. Dans ce cas, seul l'agrégat est annoncé à AS4 et AS5. Cependant, si un des composants n'est pas disponible, l'agrégat ne sera pas formé, et tout composant disponible ou plus spécifique sera exporté à AS4 et AS5. Sans considérer si l'agrégation est effectuée ou non, seuls les plus spécifiques seront exportés à AS6 (il n'est pas cité dans la liste de l'attribut aggr-mtd).

Lors d'une agrégation entrante, les générateurs de configuration peuvent éliminer les déclarations d'agrégation sur les routeurs où la politique d'importation de l'AS prohibe l'importation de tout plus spécifique.

8.1.2 Résolution d'ambiguïté avec des agrégats qui se chevauchent

Lorsque plusieurs chemins agrégés sont spécifiés et qu'ils se chevauchent, c'est-à-dire que l'un est moins spécifique que l'autre, ils peuvent être évalués dans l'ordre du plus spécifique au moins spécifique. Lorsque une agrégation sortante est effectuée pour un homologue, l'agrégé et les composants cités dans l'attribut export-comps pour cet homologue sont disponibles pour générer le prochain agrégat moins spécifique. Les composants qui ne sont pas spécifiés dans l'attribut export-comps ne sont pas disponibles. Un chemin est exportable à un AS si il est l'agrégat le moins spécifique exportable à cet AS ou si il est cité dans l'attribut export-comps d'un chemin exportable. Noter que cette définition est récurrente.

```

route: 128.8.0.0/15
origin: AS1
aggr-bndry: AS1 or AS2
aggr-mtd: outbound
inject: upon HAVE-COMPONENTS {128.8.0.0/16, 128.9.0.0/16}

route: 128.10.0.0/15
origin: AS1
aggr-bndry: AS1 or AS3
aggr-mtd: outbound
inject: upon HAVE-COMPONENTS {128.10.0.0/16, 128.11.0.0/16}
export-comps: {128.11.0.0/16}

route: 128.8.0.0/14
origin: AS1
aggr-bndry: AS1 or AS2 or AS3
aggr-mtd: outbound
inject: upon HAVE-COMPONENTS {128.8.0.0/15, 128.10.0.0/15}
export-comps: {128.10.0.0/15}

```

Figure 34 : Chevauchement d'agrégations

Dans la Figure 34, l'AS1 avec l'AS2 agrègent 128.8.0.0/16 et 128.9.0.0/16 dans 128.8.0.0/15. Avec l'AS3, l'AS1 agrège 128.10.0.0/16 et 128.11.0.0/16 dans 128.10.0.0/15. Mais ensemble, ils agrègent ces quatre chemins dans 128.8.0.0/14. En supposant que les quatre composants sont disponibles, un routeur dans AS1 pour un AS extérieur, disons AS4, va d'abord générer 128.8.0.0/15 et 128.10.0.0/15. Cela rendra 128.8.0.0/15, 128.10.0.0/15 et son exception 128.11.0.0/16 disponibles pour générer 128.8.0.0/14. Le routeur va alors générer 128.8.0.0/14 à partir de ces trois chemins. Donc pour AS4, 128.8.0.0/14 et son exception 128.10.0.0/15 et son exception 128.11.0.0/16 seront exportables.

Pour AS2, un routeur dans AS1 va seulement générer 128.10.0.0/15. Donc, 128.10.0.0/15 et son exception 128.11.0.0/16 seront exportables. Noter que 128.8.0.0/16 et 128.9.0.0/16 sont aussi exportables car ils ne participent pas à un agrégat exportable à AS2.

De même, pour AS3, un routeur dans AS1 va seulement générer 128.8.0.0/15. Dans ce cas, 128.8.0.0/15, 128.10.0.0/16, 128.11.0.0/16 sont exportables.

8.2 Spécification de chemins statiques

L'attribut inject peut être utilisé pour spécifier des chemins statiques en utilisant "upon static" comme condition :

```

inject: [at <router-expression>] ...
        [action <action>]
        upon static

```

Dans ce cas, les routeurs dans <router-expression> exécutent <action> et injectent de façon statique le chemin dans le système d'acheminement inter AS. <action> peut établir certains attributs de chemins tels qu'un routeur de prochain bond ou un coût.

Dans l'exemple qui suit, le routeur 7.7.7.1 injecte le chemin 128.7.0.0/16. Les routeurs de prochain bond (dans cet exemple, il y a deux routeurs de prochain bond) pour ce chemin sont 7.7.7.2 et 7.7.7.3 et le chemin a un coût de 10 sur 7.7.7.2 et de 20 sur 7.7.7.3.

```

route: 128.7.0.0/16
origin: AS1
inject: at 7.7.7.1 action next-hop = 7.7.7.2; cost = 10; upon static
inject: at 7.7.7.1 action next-hop = 7.7.7.3; cost = 20; upon static

```

9. Classe inet-rtr

Les routeurs sont spécifiés en utilisant la classe inet-rtr. Les attributs de la classe inet-rtr sont indiqués à la Figure 35. L'attribut inet-rtr est un nom DNS valide du routeur décrit. Chaque attribut alias, si il en est présent, est un nom DNS

canonique pour le routeur. L'attribut local-as spécifie le numéro d'AS de l'AS qui contient/fait fonctionner ce routeur.

Attribut	Valeur	Type
inet-rtr	<dns-name>	obligatoire, une seule valeur, clé de classe
alias	<dns-name>	facultatif, multi-valeurs
local-as	<as-number>	obligatoire, une seule valeur
ifaddr	voir la description du texte	obligatoire, multi-valeurs
peer	voir la description du texte	facultatif, multi-valeurs
member-of	liste de <rtr-set-names>	facultatif, multi-valeurs

Figure 35 : Attributs de classe inet-rtr

La valeur d'un attribut ifaddr a la syntaxe suivante :

```
<adresse-IPv4> masklen <entier> [action <action>]
```

L'adresse IP et la longueur de gabarit sont obligatoires pour chaque interface. Facultativement, une action peut être spécifiée pour établir d'autres paramètres de cette interface.

La Figure 36 présente un exemple d'objet inet-rtr. Le nom du routeur est "amsterdam.ripe.net". "amsterdam1.ripe.net" est un nom canonique pour le routeur. Le routeur est connecté à quatre réseaux. Ses adresses IP et longueurs de gabarit dans ces réseaux sont spécifiées dans les attributs ifaddr.

```
inet-rtr: Amsterdam.ripe.net
alias: amsterdam1.ripe.net
local-as: AS3333
ifaddr: 192.87.45.190 masklen 24
ifaddr: 192.87.4.28 masklen 24
ifaddr: 193.0.0.222 masklen 27
ifaddr: 193.0.0.158 masklen 27
peer: BGP4 192.87.45.195 asno(AS3334), flap_damp()
```

Figure 36 : Objets inet-rtr

Chaque attribut peer, si il est présent, spécifie un protocole d'appariement avec un autre routeur. La valeur d'un attribut peer a la syntaxe suivante :

```
<protocole> <adresse-ipv4> <options>
| <protocole> <inet-rtr-name> <options>
| <protocole> <rtr-set-name> <options>
| <protocole> <peering-set-name> <options>
```

où <protocole> est un nom de protocole, <adresse-ipv4> est l'adresse IP du routeur homologue, et <options> est une liste séparée par des virgules d'options d'appariement pour <protocole>. Au lieu de l'adresse IP de l'homologue, son nom inet-rtr-name peut être utilisé. Les noms et attributs de protocole possibles sont définis dans le dictionnaire (voir à la Section 7). Dans l'exemple précédent, le routeur a un appariement BGP avec le routeur 192.87.45.195 dans AS3334 et active l'atténuation de fluctuations lorsque il importe des chemins de ce routeur.

Au lieu d'un seul homologue, un groupe d'homologues peut être spécifié en utilisant les formes <rtr-set-name> et <peering-set-name>. Si la forme <peering-set-name> est utilisée, seuls les appariements dans l'ensemble d'appariements correspondant qui sont avec ce routeur sont inclus. La Figure 37 montre un exemple d'objet inet-rtr avec des groupes d'appariement.

```

rtr-set: rtrs-ibgp-peers
members: 1.1.1.1, 2.2.2.2, 3.3.3.3

peering-set: prng-ebgp-peers
peering: AS3334 192.87.45.195
peering: AS3335 192.87.45.196

inet-rtr: Amsterdam.ripe.net
alias: amsterdam1.ripe.net
local-as: AS3333
ifaddr: 192.87.45.190 masklen 24
ifaddr: 192.87.4.28 masklen 24
ifaddr: 193.0.0.222 masklen 27
ifaddr: 193.0.0.158 masklen 27
peer: BGP4 rtrs-ibgp-peers asno(AS3333), flap_damp()
peer: BGP4 prng-ebgp-peers asno(PeerAS), flap_damp()

```

Figure 37 : Objets inet-rtr avec groupes d'appariement

10 Extension à RPSL

Notre expérience de langages antérieurs de politique d'acheminement et de formats de données ([PRDB], [RIPE-81], et [RIPE-181]) nous a enseigné que RPSL doit être extensible. Il en résulte que l'extensibilité était un objectif principal de la conception de RPSL. De nouveaux protocoles d'acheminement ou de nouvelles caractéristiques des protocoles d'acheminement existants peuvent facilement être traités en utilisant la classe dictionary de RPSL. De nouvelles classes ou de nouveaux attributs des classes existantes peuvent aussi être ajoutés.

Cette section donne des lignes directrices pour étendre RPSL. Ces lignes directrices sont conçues en ayant en vue le maintien de la rétro compatibilité avec les outils et bases de données existants. On fait la liste des options disponibles pour étendre RPSL dans l'ordre de leur préférence.

10.1 Extensions en changeant la classe dictionary

La classe dictionary est le principal mécanisme fourni pour étendre RPSL. Les objets du dictionnaire définissent les attributs de politique d'acheminement, les types, et les protocoles d'acheminement.

On recommande de mettre à jour le dictionnaire RPSL de façon à y inclure les définitions appropriées de rp-attribute et de protocole lorsque de nouveaux attributs de chemin ou caractéristiques de routeur sont introduites. Par exemple, dans une version antérieure du document RPSL, il était seulement possible de spécifier qu'un routeur effectue l'amortissement de fluctuations de chemin sur un homologue, mais il n'était pas possible de spécifier les paramètres de l'amortissement de fluctuation de chemin. Les paramètres ont été ajoutés ensuite en changeant le dictionnaire.

Lorsque on change le dictionnaire, la pleine compatibilité devrait être conservée. Par exemple, dans le cas de l'amortissement de fluctuations, on a rendu facultative la spécification des paramètres pour le cas où ce niveau de détail ne serait pas désiré par certains FAI. Cela contribue aussi à la compatibilité. Tout objet enregistré sans les paramètres va continuer d'être valide. Tout outil fondé sur RPSL est supposé faire une action par défaut sur les attributs de politique d'acheminement qui ne sont pas compris (par exemple, de produire un avertissement et de l'ignorer par ailleurs). Donc, les vieux outils qui rencontrent une spécification d'amortissement de fluctuations avec des paramètres vont ignorer les paramètres.

10.2 Extensions en ajoutant de nouveaux attribut à des classes existantes

De nouveaux attributs peuvent être ajoutés à toute classe. Pour assurer la pleine compatibilité, les nouveaux attributs ne devraient pas contredire la sémantique des objets auxquels ils sont attachés. Tout outil qui utilise l'IRR devrait être conçu pour ignorer les attributs qu'il ne comprend pas. La plupart des outils existants adhèrent à ce principe de conception.

On recommande d'ajouter de nouveaux attributs aux classes existantes lorsque un nouvel aspect d'une classe est découvert. Par exemple, la classe route RPSL étend son prédécesseur RIPE-181 en incluant plusieurs nouveaux attributs qui permettent la spécification de chemins agrégés et statiques.

10.3 Extensions par ajout de nouvelles classes

De nouvelles classes peuvent être ajoutées à RPSL pour mémoriser de nouveaux types de données de politique. Fournir la pleine compatibilité découle naturellement de la conservation de la compréhension des classes existantes. Comme un outil ne devrait interroger l'IRR que pour les classes qu'il comprend, la pleine compatibilité ne devrait pas poser de problème dans ce cas.

Avant d'ajouter une nouvelle classe, on devrait s'interroger sur le point de savoir si les informations contenues dans les objets de la nouvelle classe ne devraient pas appartenir à une autre classe. Par exemple, si la localisation géographique d'un routeur doit être mémorisée dans l'IRR, il peut être tentant d'ajouter une nouvelle classe appelée, par exemple, classe router-location. Cependant, les informations se trouvent mieux dans la classe inet-rtr, peut-être dans un nouvel attribut appelé location.

10.4 Extensions en changeant la syntaxe des attributs RPSL existants

Si toutes les méthodes décrites ci-dessus ne réussissent pas à fournir l'extension désirée, il peut être nécessaire de changer la syntaxe de RPSL. Tout changement dans la syntaxe de RPSL doit assurer la rétro compatibilité, et devrait être considéré seulement comme un dernier recours lorsque la pleine compatibilité n'est pas réalisable. Cependant, on exige que la vieille syntaxe soit toujours valide.

11. Considérations pour la sécurité

Le présent document décrit RPSL, un langage pour exprimer les politiques d'acheminement. Le langage définit un objet maintenir (classe mntner) qui est l'entité qui contrôle ou "maintient" les objets mémorisés dans une base de données exprimée par RPSL. Les demandes des mainteneurs peuvent être authentifiées par diverses techniques définies par l'attribut "auth" de l'objet maintenir.

Les protocoles exacts utilisés par les IRR pour communiquer les objets RPSL sortent du domaine d'application du présent document, mais il est envisagé que plusieurs techniques puissent être utilisées, allant de protocoles interactifs d'interrogation/mise à jour pour mémoriser et transmettre des protocoles similaires à, ou fondés sur, la messagerie électronique (ou même les appels téléphoniques vocaux). Sans considération des protocoles qui sont utilisés dans une certaine situation, il est prévu que les techniques de sécurité appropriées telles que IPsec, TLS ou PGP/MIME seront utilisées.

12. Remerciements

Nous tenons à remercier Jessica Yu, Randy Bush, Alan Barrett, Bill Manning, Sue Hares, Ramesh Govindan, Kannan Varadhan, Satish Kumar, Craig Labovitz, Rusty Eddy, David J. LeRoy, David Whipple, Jon Postel, Deborah Estrin, Elliot Schwartz, Joachim Schmitz, Mark Prior, Tony Przygienda, David Woodgate, Rob Coltun, Sanjay Wadhwa, Ardas Cilingiroglu, et les participants au groupe de travail RPS de l'IETF pour leurs divers commentaires et suggestions.

Références

- [IRR] "Internet routing registry. procedures". <http://www.ra.net/RADB.tools.docs/> , <http://www.ripe.net/db/doc.html> .
- [IDRP] Y. Rekhter. "Inter-domain routing protocol (idrp)". Journal of Internetworking Research and Experience, 4:61--80, 1993.
- [LANG-C] B. W. Kernighan and D. M. Ritchie. "The C Programming Language". Prentice-Hall, 1978.
- [PRDB] "Nsfnet policy routing database (prdb)". Tenu par MERIT Network Inc., Ann Arbor, Michigan. Contenu disponible à nic.merit.edu./nsfnet/announced.networks/nets.tag.now par FTP anonyme.
- [RFC0822] D. Crocker, "Norme pour le [format des messages de texte](#) de l'ARPA-Internet", STD 11, août 1982. (*Obsolète, voir la RFC5322*)
- [RFC1034] P. Mockapetris, "Noms de domaines - [Concepts et facilités](#)", STD 13, novembre 1987.
- [RFC1519] V. Fuller, T. Li, J. Yu et K. Varadhan, "Acheminement inter domaine sans classe (CIDR) : stratégie d'allocation

et d'agrégation d'adresses", septembre 1993. (*D.S.*, *Obsolète voir RFC4632*)

- [RFC1771] Y. Rekhter, T. Li, "Protocole de routeur frontière v. 4 (BGP-4)", mars 1995. (*Obsolète, voir RFC4271*) (*D.S.*)
- [RFC1786] T. Bates, E. Gerich, L. Joncheray, J-M. Jouanigot, D. Karrenberg, M. Terpstra, J. Yu, "Représentation des politiques d'acheminement IP dans un registre d'acheminement (ripe-81++)", mars 1995. (*Information*)
- [RFC1997] R. Chandra, P. Traina, T. Li, "[Attribut Community de BGP](#)", août 1996. (*P.S.*)
- [RFC2280] C. Alaettinoglu, T. Bates, E. Gerich, D. Karrenberg, D. Meyer, M. Terpstra, C. Villamizar, "Langage de spécification des politiques d'acheminement (RPSL)", janvier 1998. (*Obsolète, voir RFC2622*) (*P.S.*)
- [RFC2439] C. Villamizar, R. Chandra, R. Govindan, "Élimination des oscillations de chemin dans BGP", novembre 1998. (*P.S.*)
- [RFC2650] D. Meyer, J. Schmitz, C. Orange, M. Prior, C. Alaettinoglu, "RPSL pratique", août 1999. (*Information*)
- [RFC2725] C. Villamizar et autres, "[Sécurité du système de politique](#) d'acheminement", décembre 1999. (*MàJ par RFC4012*) (*P.S.*)
- [RFC2726] J. Zsako, "[Authentification de PGP](#) pour mise à jour de base de données RIPE", décembre 1999. (*P.S.*)
- [RIPE-81] T. Bates, J-M. Jouanigot, D. Karrenberg, P. Lothberg, and M. Terpstra. "Representation of ip routing policies in the ripe database". Technical Report ripe-81, RIPE, RIPE NCC, Amsterdam, Netherlands, février 1993.
- [RIPE-104] D. Karrenberg and T. Bates. "Description of inter-as networks in the ripe routing registry". Technical Report RIPE-104, RIPE, RIPE NCC, Amsterdam, Netherlands, décembre 1993.
- [RIPE-119] A. Lord and M. Terpstra. "Ripe database template for networks and persons". Technical Report ripe-119, RIPE, RIPE NCC, Amsterdam, Netherlands, octobre 1994.
- [RIPE-120] D. Karrenberg and M. Terpstra. "Authorisation and notification of changes in the ripe database". Technical Report RIPE-120, RIPE, RIPE NCC, Amsterdam, Netherlands, octobre 1994.
- [RIPE-122] T. Bates. "Specifying an `internet router' in the routing registry". Technical Report RIPE-122, RIPE, RIPE NCC, Amsterdam, Netherlands, octobre 1994.
- [RIPE-157] A. M. R. Magee. "Ripe ncc database documentation". Technical Report RIPE-157, RIPE, RIPE NCC, Amsterdam, Netherlands, mai 1997.
- [RIPE-181] T. Bates, E. Gerich, L. Joncheray, J-M. Jouanigot, D. Karrenberg, M. Terpstra, and J. Yu. "Representation of ip routing policies in a routing registry". Technical Report ripe-181, RIPE, RIPE NCC, Amsterdam, Netherlands, octobre 1994.

A Sites des registres d'acheminement

En novembre 1996, les registres d'acheminement sont RIPE, RADB, CANet, MCI et ANS. On peut contacter un de ces registres pour obtenir la liste actuelle des registres.

B Règles de grammaire

La présente section donne les règles de grammaire formelle pour RPSL. Les types de données de base sont définis à la Section 2. On ne donne pas les règles de grammaire formelle pour les attributs dont les valeurs sont des types de base ou une liste de types de base. Les règles sont écrites en utilisant le langage d'entrée d'analyseur Bison GNU. Elles peuvent donc être coupées/collées dans ce programme.

//**** Attributs génériques *****

changed_attribute: ATTR_CHANGED TKN_EMAIL TKN_INT

```

/**** classe aut-num ****
////////////////////////////////////

//// as_expression //////////////////////////////////////

opt_as_expression:
| as_expression

as_expression: as_expression OP_OR as_expression_term
| as_expression_term

as_expression_term: as_expression_term OP_AND as_expression_factor
| as_expression_term KEYW_EXCEPT as_expression_factor
| as_expression_factor

as_expression_factor: '(' as_expression ')'
| as_expression_operand

as_expression_operand: TKN_ASNO
| TKN_ASNAME

//// router_expression //////////////////////////////////////

opt_router_expression:
| router_expression

opt_router_expression_with_at:
| KEYW_AT router_expression

router_expression: router_expression OP_OR router_expression_term
| router_expression_term

router_expression_term: router_expression_term OP_AND router_expression_factor
| router_expression_term KEYW_EXCEPT router_expression_factor
| router_expression_factor

router_expression_factor: '(' router_expression ')'
| router_expression_operand

router_expression_operand: TKN_IPV4
| TKN_DNS
| TKN_RTRSNAME

//// peering //////////////////////////////////////

peering: as_expression opt_router_expression opt_router_expression_with_at
| TKN_PRNGNAME

//// action //////////////////////////////////////

opt_action:
| KEYW_ACTION action

action: single_action
| action single_action
single_action: TKN_RP_ATTR '!' TKN_WORD '(' generic_list ')' ';'
| TKN_RP_ATTR TKN_OPERATOR list_item ';'
| TKN_RP_ATTR '(' generic_list ')' ';'
| TKN_RP_ATTR '[' generic_list ']' ';'
| ';'

//// filter //////////////////////////////////////

filter: filter OP_OR filter_term

```

```

| filter filter_term %prec OP_OR
| filter_term

filter_term : filter_term OP_AND filter_factor
| filter_factor

filter_factor : OP_NOT filter_factor
| '(' filter ')'
| filter_operand

filter_operand: KEYW_ANY
| '<' filter_aspath '>'

| filter_rp_attribute
| TKN_FLTRNAME
| filter_prefix

filter_prefix: filter_prefix_operand OP_MS
| filter_prefix_operand

filter_prefix_operand: TKN_ASNO
| KEYW_PEERAS
| TKN_ASNAME
| TKN_RSNAME
| '{' opt_filter_prefix_list '}'

opt_filter_prefix_list:
| filter_prefix_list

filter_prefix_list: filter_prefix_list_prefix
| filter_prefix_list ',' filter_prefix_list_prefix

filter_prefix_list_prefix: TKN_PREFX4
| TKN_PREFX4RNG

filter_aspath: filter_aspath '|' filter_aspath_term
| filter_aspath_term

filter_aspath_term: filter_aspath_term filter_aspath_closure
| filter_aspath_closure

filter_aspath_closure: filter_aspath_closure '*'
| filter_aspath_closure '?'
| filter_aspath_closure '+'
| filter_aspath_factor

filter_aspath_factor: '^'
| '$'
| '(' filter_aspath ')'
| filter_aspath_no

filter_aspath_no: TKN_ASNO
| KEYW_PEERAS
| TKN_ASNAME
| '.'
| '[' filter_aspath_range ']'
| '[' '^' filter_aspath_range ']'

filter_aspath_range:
| filter_aspath_range TKN_ASNO
| filter_aspath_range KEYW_PEERAS
| filter_aspath_range '.'
| filter_aspath_range TKN_ASNO '-' TKN_ASNO

```

```

| filter_aspath_range TKN_ASNAME

filter_rp_attribute: TKN_RP_ATTR '!' TKN_WORD '(' generic_list ')'
| TKN_RP_ATTR TKN_OPERATOR list_item
| TKN_RP_ATTR '(' generic_list ')'
| TKN_RP_ATTR '[' generic_list ']'

//// paire d'actions d'appariement //////////////////////////////////////

import_peering_action_list: KEYW_FROM peering opt_action
| import_peering_action_list KEYW_FROM peering opt_action

export_peering_action_list: KEYW_TO peering opt_action
| export_peering_action_list KEYW_TO peering opt_action

//// facteur d'import/export //////////////////////////////////////

import_factor: import_peering_action_list KEYW_ACCEPT filter

import_factor_list: import_factor ';'
| import_factor_list import_factor ';'

export_factor: export_peering_action_list KEYW_ANNOUNCE filter

export_factor_list: export_factor ';'
| export_factor_list export_factor ';'

//// terme d'import/export //////////////////////////////////////

import_term: import_factor ';'
| '{' import_factor_list '}'

export_term: export_factor ';'
| '{' export_factor_list '}'

//// expression d'import/export //////////////////////////////////////

import_expression: import_term
| import_term KEYW_REFINE import_expression
| import_term KEYW_EXCEPT import_expression

export_expression: export_term
| export_term KEYW_REFINE export_expression
| export_term KEYW_EXCEPT export_expression

//// protocole //////////////////////////////////////

opt_protocol_from:
| KEYW_PROTOCOL tkn_word

opt_protocol_into:
| KEYW_INTRO tkn_word

//**** attributs d'import/export *****/

import_attribute: ATTR_IMPORT
| ATTR_IMPORT opt_protocol_from opt_protocol_into import_factor

export_attribute: ATTR_EXPORT
| ATTR_EXPORT opt_protocol_from opt_protocol_into export_factor

opt_default_filter:
| KEYW_NETWORKS filter

```

```

default_attribute: ATTR_DEFAULT KEYW_TO peering

filter_attribute: ATTR_FILTER filter

peering_attribute: ATTR_PEERING peering

//**** classe inet-rtr ****
ifaddr_attribute: ATTR_IFADDR TKN_IPV4 KEYW_MASKLEN TKN_INT opt_action

//// attribut peer //////////////////////////////////////

opt_peer_options:
| peer_options

peer_options: peer_option
| peer_options ',' peer_option

peer_option: tkn_word '(' generic_list ')'

peer_id: TKN_IPV4
| TKN_DNS
| TKN_RTRSNAME
| TKN_PRNGNAME

peer_attribute: ATTR_PEER tkn_word peer_id opt_peer_options

//**** classe route ****

aggr_bndry_attribute: ATTR_AGGR_BNDRY as_expression

aggr_mtd_attribute: ATTR_AGGR_MTD KEYW_INBOUND
| ATTR_AGGR_MTD KEYW_OUTBOUND opt_as_expression

//// attribut inject //////////////////////////////////////

opt_inject_expression:
| KEYW_UPON inject_expression

inject_expression: inject_expression OP_OR inject_expression_term
| inject_expression_term

inject_expression_term: inject_expression_term OP_AND inject_expression_factor
| inject_expression_factor

inject_expression_factor: '(' inject_expression ')'
| inject_expression_operand

inject_expression_operand: KEYW_STATIC
| KEYW_HAVE_COMPONENTS '{' opt_filter_prefix_list '}'
| KEYW_EXCLUDE '{' opt_filter_prefix_list '}'

inject_attribute: ATTR_INJECT opt_router_expression_with_at opt_action opt_inject_expression

//// attribut components //////////////////////////////////////

opt_atomic:
| KEYW_ATOMIC

components_list:
| filter
| components_list KEYW_PROTOCOL tkn_word filter

```

```

components_attribute: ATTR_COMPONENTS opt_atomic components_list

/**** route-set *****/

opt_rs_members_list: /* empty list */
| rs_members_list

rs_members_list: rs_member
| rs_members_list ',' rs_member

rs_member: TKN_ASNO
| TKN_ASNO OP_MS
| TKN_ASNAME
| TKN_ASNAME OP_MS
| TKN_RSNAME
| TKN_RSNAME OP_MS
| TKN_PREFIXV4

| TKN_PREFIXV4RNG

rs_members_attribute: ATTR_RS_MEMBERS opt_rs_members_list

/**** dictionary *****/

rpattr_attribute: ATTR_RP_ATTR TKN_WORD methods
| ATTR_RP_ATTR TKN_RP_ATTR methods

methods: method
| methods method

method: TKN_WORD '(' ')'
| TKN_WORD '(' typedef_type_list ')'
| TKN_WORD '(' typedef_type_list ',' TKN_3DOTS ')'
| KEYW_OPERATOR TKN_OPERATOR '(' typedef_type_list ')'
| KEYW_OPERATOR TKN_OPERATOR '(' typedef_type_list ',' TKN_3DOTS ')'

//// attribut typedef //////////////////////////////////////

typedef_attribute: ATTR_TYPEDEF TKN_WORD typedef_type

typedef_type_list: typedef_type
| typedef_type_list ',' typedef_type

typedef_type: KEYW_UNION typedef_type_list
| KEYW_RANGE KEYW_OF typedef_type
| TKN_WORD
| TKN_WORD '[' TKN_INT ',' TKN_INT ']'
| TKN_WORD '[' TKN_REAL ',' TKN_REAL ']'
| TKN_WORD '[' enum_list ']'
| KEYW_LIST '[' TKN_INT ':' TKN_INT ']' KEYW_OF typedef_type
| KEYW_LIST KEYW_OF typedef_type

enum_list: tkn_word
| enum_list ',' tkn_word

//// attribut protocol //////////////////////////////////////

protocol_attribute: ATTR_PROTOCOL tkn_word protocol_options

protocol_options:
| protocol_options protocol_option

```

```
protocol_option: KEYW_MANDATORY method
| KEYW_OPTIONAL method
```

```
/***/ Définitions de jetons *****/
```

```
//// macros flex utilisées dans les définitions de jetons ////////////////
```

```
INT      [[:digit:]]+
SINT     [+]?{INT}
REAL     [+]?{INT}?\. {INT}({WS}*E{WS})*[+]?{INT}?
NAME     [[:alpha:]]([[:alnum:]]_)*[[:alnum:]]?
ASNO     AS{INT}
ASNAME   AS-[[:alnum:]]_]*[[:alnum:]]
RSNAME   RS-[[:alnum:]]_]*[[:alnum:]]
RTRSNAME RTRS-[[:alnum:]]_]*[[:alnum:]]
PRNGNAME PRNG-[[:alnum:]]_]*[[:alnum:]]
FLTRNAME FLTR-[[:alnum:]]_]*[[:alnum:]]
IPV4     [0-9]+(\.[0-9]+){3,3}
PRFXV4   {IPV4}\{0-9}+
PRFXV4RNG {PRFXV4}{"^+|"^-|"^" {INT}|"^" {INT}-{INT}}
ENAMECHAR [^()<,:;\\\"' \.[] \t\r]
ENAME    ({ENAMECHAR}+(\. {ENAMECHAR})*\.)?(\["^"@\r\n]+)
DNAME    [[:alnum:]]_]+
```

```
//// Définitions de jetons ////////////////
```

```
TKN_INT      {SINT}
TKN_INT      {INT};{INT}          if each {INT} is two octets
TKN_INT      {INT}. {INT}. {INT}. {INT} if each {INT} is one octet
TKN_REAL     {REAL}
TKN_STRING   Same as in programming language C
TKN_IPV4     {IPV4}
TKN_PRFXV4   {PRFXV4}
TKN_PRFXV4RNG {PRFXV4RNG}
TKN_ASNO     {ASNO}
TKN_ASNAME   (({ASNO}|peeras|{ASNAME}):)*{ASNAME}\ (:({ASNO}|peeras|{ASNAME}))*)
TKN_RSNAME   (({ASNO}|peeras|{RSNAME}):)*{RSNAME}\ (:({ASNO}|peeras|{RSNAME}))*)
TKN_RTRSNAME (({ASNO}|peeras|{RTRSNAME}):)*{RTRSNAME}\ (:({ASNO}|peeras|{RTRSNAME}))*)
TKN_PRNGNAME (({ASNO}|peeras|{PRNGNAME}):)*{PRNGNAME}\ (:({ASNO}|peeras|{PRNGNAME}))*)
TKN_FLTRNAME (({ASNO}|peeras|{FLTRNAME}):)*{FLTRNAME}\ (:({ASNO}|peeras|{FLTRNAME}))*)
TKN_BOOLEAN  true|false
TKN_RP_ATTR  {NAME} if defined in dictionary
TKN_WORD     {NAME}
TKN_DNS      {DNAME}("."{DNAME})+
TKN_EMAIL    {ENAME}@({DNAME}("."{DNAME})+){IPV4}
```

C. Changements depuis la RFC2280

La [RFC2280] contient une version antérieure de RPSL. On a résumé ici les changements intervenus depuis. Ce sont les suivants :

- o Il est maintenant possible d'écrire les entiers comme séquence de quatre entiers d'un octet (par exemple, 1.1.1.1) ou comme séquence d'entiers de deux octets (par exemple, 3561:70). Voir la Section 2.
- o La définition de la gamme de préfixe d'adresse est étendue afin qu'un préfixe d'adresse soit aussi une gamme de préfixes d'adresse. Voir la Section 2.
- o La sémantique d'un opérateur de gamme appliqué à un ensemble contenant des gammes de préfixes d'adresse est définie (par exemple, {30.0.0.0/8^24-28}^27-30). Voir la Section 2.
- o Toutes les dates sont maintenant en UTC. Voir la Section 2.
- o Le caractère plus (+) est ajouté aux caractères espace et tabulation pour partager la valeur d'un attribut en plusieurs lignes (c'est-à-dire en commençant la ligne suivante par un caractère espace, tabulation ou plus (+)). Voir la Section 2.
- o L'attribut retiré de classe route est supprimé du langage.
- o La classe filter-set est introduite. Voir le paragraphe 5.4.
- o La classe rtr-set est introduite. Voir le paragraphe 5.5.
- o La classe peering-set est introduite. Voir le paragraphe 5.6.
- o Les filtres peuvent maintenant se référer aux noms filter-set. Voir le paragraphe 5.4.

- o Les appariements peuvent maintenant se référer aux noms peering-set, rtr-set. Les routeurs aussi bien locaux qu'homologues peuvent être spécifiés en utilisant les expressions router. Voir le paragraphe 5.6.
- o L'attribut peer de la classe inet-rtr peut se référer aux noms peering-set, rtr-set. Voir à la Section 9.
- o La syntaxe et la sémantique des types union et list et d'attribut typedef ont changé. Voir à la Section 7.
- o Dans le dictionnaire initial, l'attribut typedef qui définit community_elm, rp-attribute qui définit l'attribut community a changé. Voir à la Section 7.
- o On a ajouté les lignes directrices pour l'extension de RPSL. Voir la Section 10.
- o Les règles de grammaire formelle ont été ajoutées. Voir à l'Appendice B.

D. Adresse des auteurs

Cengiz Alaettinoglu
USC/ISI
mél : cengiz@isi.edu

Curtis Villamizar
Avici Systems
mél : curtis@avici.com

Elise Gerich
At Home Network
mél : epg@home.net

Marten Terpstra
c/o Bay Networks, Inc.
mél : marten@BayNetworks.com

David Kessens
Qwest Communications
mél : David.Kessens@qwest.net

David Meyer
University of Oregon
mél : meyer@antc.uoregon.edu

Tony Bates
Cisco Systems, Inc.
mél : tbates@cisco.com

Daniel Karrenberg
RIPE NCC
mél : dfk@ripe.net

Déclaration complète de droits de reproduction

Copyright (C) The Internet Society (1999). Tous droits réservés.

Ce document et les traductions de celui-ci peuvent être copiés et diffusés, et les travaux dérivés qui commentent ou expliquent autrement ou aident à sa mise en œuvre peuvent être préparés, copiés, publiés et distribués, partiellement ou en totalité, sans restriction d'aucune sorte, à condition que l'avis de copyright ci-dessus et ce paragraphe soit inclus sur toutes ces copies et œuvres dérivées. Toutefois, ce document lui-même ne peut être modifié en aucune façon, par exemple en supprimant le droit d'auteur ou les références à l'Internet Society ou d'autres organisations Internet, sauf si c'est nécessaire à l'élaboration des normes Internet, auquel cas les procédures pour les droits de reproduction définis dans les processus des normes pour l'Internet doivent être suivies, ou si nécessaire pour le traduire dans des langues autres que l'anglais.

Les permissions limitées accordées ci-dessus sont perpétuelles et ne seront pas révoquées par la Société Internet, ses successeurs ou ayants droit.

Ce document et les renseignements qu'il contient sont fournis "TELS QUELS" et l'INTERNET SOCIETY et l'INTERNET ENGINEERING TASK FORCE déclinent toute garantie, expresse ou implicite, y compris mais sans s'y limiter, toute garantie que l'utilisation de l'information ici présente n'enfreindra aucun droit ou aucune garantie implicite de commercialisation ou d'adaptation à un objet particulier.

Remerciement

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