Network Working Group
Request for Comments: 2622
Obsoletes: 2280
Category: Standards Track

C. Alaettinoglu
USC/Information Sciences Institute
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

June 1999

Язык описания политики маршрутизации (RPSL)

Routing Policy Specification Language (RPSL)

Статус документа

В этом документе содержится спецификация протокола, предложенного сообществу Internet. Документ служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола вы можете узнать из документа "Internet Official Protocol Standards" (STD 1). Документ может распространяться без ограничений.

Авторские права

Аннотация

Язык RPSL позволяет сетевым операторам задавать правила маршрутизации на различных уровнях иерархии Internet (например, на уровне автономной системы). В то же время правила в RPSL могут задаваться достаточно детально, что обеспечивает возможность генерации на базе этих правил конфигурационных параметров маршрутизаторов. Язык RPSL является расширяемым - новые протоколы маршрутизации и новые возможности протоколов могут добавляться в язык в любой момент.

Оглавление

1 Введение

Этот документ является справочником по языку описания правил маршрутизации RPSL[1]. Язык RPSL позволяет сетевым операторам задавать правила маршрутизации на различных уровнях иерархии Internet (например, на уровне автономной системы - AS[2]). В то же время правила в RPSL могут задаваться достаточно детально, что обеспечивает возможность генерации на базе этих правил конфигурационных параметров маршрутизаторов. Язык RPSL является расширяемым - новые протоколы маршрутизации и новые возможности протоколов могут добавляться в язык в любой момент.

RPSL предназначен для замены используемого в Internet языка описания правил, известного как RIPE-181 [6] или RFC-1786 [7]. RIPE-81 [8] был первым языком, использованным в Internet, для описания политики маршрутизации. Позднее ему на смену пришел язык RIPE-181 [6]. В процессе работы с языком RIPE-181 стало ясно, что некоторые правила не могут быть выражены в рамках этого языка и требуется расширенный и более обобщенный язык. RPSL свободен от ограничений, присущих RIPE-181.

Язык RPSL был спроектирован так, чтобы представление глобальной политики маршрутизации могло содержаться в одной коллективно поддерживаемой распределенной базе данных для обеспечения целостности картины маршрутизации Internet. RPSL не предназначен для использования в качестве языка настройки конфигурации маршрутизаторов. Однако RPSL позволяет генерировать конфигурационные параметры маршрутизаторов из описания политики автономной системы (класс aut-num), вкупе с описанием маршрутизатора (класс inet-rtr), представленным значением router ID, номером автономной системы, указанием интерфейсов и партнёров маршрутизатора, а также с глобальными отображениями наборов AS в номера AS (класс as-set) и отображениями исходных AS и наборов маршрутов в маршрутные префиксы (классы route и route-set). Аккуратное заполнение базы данных RPSL может помочь в решении таких задач, как создание конфигурации маршрутизатора, которая защитит от случайного или преднамеренного распространения неточных маршрутных данных, верификация картины маршрутизации в Internet и границ агрегирования за пределами одной AS.

RPSL является объектно-ориентированным языком; т. е., объекты содержать компоненты политики и административных данных. Эти объекты регистрируются в реестре маршрутизации Internet (IRR[3]) уполномоченными на то организациями. Процесс регистрации выходит за рамки данного документа. Дополнительную информацию о реестре маршрутизации IRR вы найдёте в [1, 17, 4] .

В следующих параграфах рассматриваются классы, используемые для определения различных объектов политики и администрирования. Класс mntner определяет объекты, уполномоченные добавлять, удалять или изменять набор объектов. Классы person и role описывают контакты с техническим и административным персоналом. Автономные системы задаются с использованием класса aut-num. Для задания маршрутизаторов служит класс route. Наборы объектов могут определяться с использованием классов as-set, route-set, filter-set, peering-set и rtr-set. Класс dictionary обеспечивает расширяемость языка. Класс inet-rtr служит для указания маршрутизаторов. Многие из этих классов были изначально определены в ранних документах [6, 13, 16, 12, 5] и впоследствии расширены.

Этот документ является самодостаточным. Однако читателям рекомендуется ознакомиться с RIPE-181 [7] и связанными с ним документами [13, 16, 12, 5], поскольку они обеспечивают понимание мотивации и основополагающих принципов RIPE-181 и RPSL. В качестве учебного пособия по языку RPSL рекомендуется прочесть документ по применению языка RPSL [4].

2 Имена RPSL, зарезервированные слова и представление

Каждый класс имеет набор атрибутов, сохраняющих часть информации об объектах класса. Атрибуты могут быть обязательными и опциональными. Обязательные атрибуты относятся ко всем объектам класса, необязательные могут отсутствовать. Атрибуты могут иметь одно или множество значений. Каждый объект уникально идентифицируется набором атрибутов, который будем называть «ключом» (key).

Значение атрибута имеет тип. Наиболее распространённые типы перечислены ниже. Отметим, что в языке RPSL могут использоваться только символы набора ASCII; регистр символов значения не имеет.

<object-name>

Многие объекты в RPSL имеют имя. Значение <object-name> может включать буквы, цифры, символ подчёркивания «_» и дефис «-»; первым символом имени должна быть буква, а последним - буква или цифра. Перечисленные ниже слова зарезервированы в RPSL и не могут служить именами:

```
any as-any rs-any peeras
```

[1] Routing Policy Specification Language.
[2] Autonomous System.
[3] Internet Routing Registry.

```
and or not
atomic from to at action accept announce except refine
networks into inbound outbound
```

Имена, начинающиеся с некоторых префиксов, зарезервированы для отдельных типов объектов - префикс «as-» служит для имён наборов автономных систем, «rs-» - для наборов маршрутов, «rtrs-» - для наборов маршрутизаторов, «fltr-» - для наборов фильтров и «prng-» - для наборов партнёров (peering).

<as-number>

Автономная система (AS) с номером x представляется, как Asx (т. е., AS 226 представляется в виде AS226).

<ipv4-address>

Адрес IPv4 представляется в форме четырёх целых чисел (от 0 до 255), разделённых точками. Например, 128.9.128.5 является корректной записью адреса IPv4. В оставшейся части документа адреса IPv4 будут называться просто адресами IP.

<address-prefix>

Адресный префикс представляется в виде адреса IPv4, за которым следует символ дробной черты «/» и целое число от 0 до 32. Записи 128.9.128.5/32, 128.9.0.0/16, 0.0.0.0/0 являются корректным представлением префиксов, а 0/0, 128.9/16 - некорректны, поскольку 0 и 128.9 не являются строками из 4 целых чисел.

<address-prefix-range>

Диапазон префиксов задаётся адресным префиксом, за которым может следовать оператор диапазона. Операторы диапазона включают:

^-

оператор специфичности с исключением, которому соответствуют все более специфичные (длинные) префиксы, за исключением заданного (например, 128.9.0.0/16^ соответствуют все более длинные префиксы из 128.9.0.0/16, за исключением 128.9.0.0/16);

^+

оператор специфичности с включением, которому соответствуют все более специфичные (длинные) префиксы, включая заданный (например, 5.0.0.0/8^+ соответствуют все более специфичные префиксы из 5.0.0.0/8, включая 5.0.0.0/8);

^n

оператор заданной специфичности (n - целое число), которому соответствуют все префиксы, размером n в указанном префиксе (например, 30.0.0.0/8^16 соответствуют все префиксы длиной 16 из 30.0.0.0/8, такие, как 30.9.0.0/16);

^n-m

оператор заданного диапазона специфичности (n и m - целые числа), которому соответствуют все префиксы длиной от n до m в указанном префиксе (например, 30.0.0.0/8^24-32 соответствуют все префиксы длиной от 24 до 32 из префикса 30.0.0.0/8, такие, как 30.9.9.96/28).

Операторы диапазона применимы и для множеств префиксов. В этом случае они относятся ко всем префиксам множества. Например, для множества маршрутов (route-set) rs-foo, запись rs-foo^+ будет обозначать все префиксы из множества rs-foo и все более длинные префиксы, соответствующие им.

Операторы диапазона не могут следовать один за другим lt (например, 30.0.0.0/8^24-28^+ является ошибочной записью). Однако оператор диапазона может быть применён к множеству адресных префиксов, члены которого имеют операторы диапазона (например, запись {30.0.0.0/8^24-28}^27-30 будет корректной). В таких случаях внешний оператор (^n-m) распространяет своё действие на внутренний (^k-l) и результат принимает форму ^max(n,k)-m, если m ≥ max(n,k), а в противном случае префикс удаляется из набора. Отметим, что оператор ^n эквивалентен to ^n-n, prefix/l^+ будет эквивалентом prefix/l^l-32, prefix/l^- будет эквивалентом prefix/l^(l+1)-32, {prefix/l^n-m}^+ - эквивалентом {prefix/l^n-32}, а {prefix/l^n-m}^- будет эквивалентом {prefix/l^(n+1)-32}. Например,

```
{128.9.0.0/16^+}^-     == {128.9.0.0/16^-}
{128.9.0.0/16^-}^+     == {128.9.0.0/16^-}
{128.9.0.0/16^17}^24   == {128.9.0.0/16^24}
{128.9.0.0/16^20-24}^26-28 == {128.9.0.0/16^26-28}
{128.9.0.0/16^20-24}^22-28 == {128.9.0.0/16^22-28}
{128.9.0.0/16^20-24}^18-28 == {128.9.0.0/16^20-28}
{128.9.0.0/16^20-24}^18-22 == {128.9.0.0/16^20-22}
{128.9.0.0/16^20-24}^18-19 == {}
```

<date>

Дата, представленная 8-значным целым числом в форме YYYYMMDD, где YYYY представляет год, MM - месяц (01 - 12), а DD - число месяца (01 - 31). Все даты отсчитываются от UTC, если явно не указано иное. Например, 24 июня 1996 года будет представляться в виде 19960624.

<email-address>

Адрес электронной почты в соответствии с RFC 822 [10].

<dns-name>

Доменное имя в соответствии с RFC1034 [17].

<nic-handle>

Уникальный идентификатор, используемый в реестрах маршрутизации, распределения адресов и т. п., для однозначного указания контактной информации. Классы person и role отображают эти идентификаторы на имена реальных людей и сведения о них.

<free-form>

Последовательность символов ASCII.

<X-name>

Имя объекта типа X (т. е., <mntner-name> задаёт имя объекта класса mntner).

<registry-name>

Имя реестра IRR (реестры маршрутизации перечислены в Приложении A.

Значение атрибута может также быть списком элементов одного из перечисленных типов. Список содержит набор элементов, разделённых запятыми. Например, «AS1, AS2, AS3, AS4» задаёт список номеров AS. Отметим, что списки и множественные значения являются ортогональными. Атрибут со множеством значений может иметь несколько значений, являющихся или не являющихся списками. С другой стороны, атрибут с одним значением, может быть представлен списком.

Объекты RPSL представляются в форме пар «атрибут-значение». Каждая такая пара записывается в отдельной строке. В начале строки (позиция 0) указывается имя атрибута, затем символ двоеточия «:», после которого указывается значение атрибута. Атрибуты, имена которых совпадают с именем класса объектов, следует указывать первыми. Представление объектов завершается пустой строкой. Если значения атрибута на помещаются в одной строке, каждая новая строка начинается с символа пробела « », табуляции или знака «+», которые обозначают конкатенацию строк. Знак «+» позволяет включать в значения атрибутов пустые строки. Для улучшения восприятия текста в начале каждой строки многострочных значений могут помещаться дополнительные символы пробела. Порядок размещения пар «attribute-value» имеет значение.

Описание объекта может включать комментарии. Комментарий может располагаться в любом месте определения объекта и начинается с первого символа # в строке, а заканчивается первым символом завершения строки. Для улучшения восприятия в комментариях могут использоваться дополнительные символы пробела.

Целые числа задаются (1) с использованием нотации языка C (например, 1, 12345); (2) в виде четырёх 1-октетных значения (от 0 до 255), разделённых точками (например, 1.1.1.1, 255.255.0.0) - в этом случае 4-октетное целое число образуется путём конкатенации однооктетных компонент от старшего октета к младшему; (3) в форме двух 2-октетных целых чисел (от 0 до 65535), разделённых двоеточием «:» (например, 3561:70, 3582:10) - в этом случае 4-октетное целое число образуется за счёт конкатенации 2-октетных компонент от старшего к младшему.

3 Контактная информация

Классы mntner, person и role вместе с атрибутами admin-c, tech-c, mnt-by, changed и source всех классов задают контактную информацию. Класс mntner задаёт также идентификационные данные, требуемые для создания, удаления и изменения других объектов. Перечисленные классы не задают правил маршрутизации и в каждом реестре к этим класса могут предъявляться разные требования. Здесь для полноты картины приведены требования к этим классам, принятые в базе данных RIPE [16]. Получить текущую версию спецификации этих классов вы можете в своём реестре маршрутизации. В документе Routing Policy System Security [20] более подробно описаны требования по идентификации и аутентификации.

Атрибут	**Значение**	**Тип**
mntner	<object-name>	Обязательный, одно значение, ключ класса
descr	<free-form>	Обязательный, одно значение
auth	см. ниже	Обязательный, много значений
upd-to	<email-address>	Обязательный, много значений
mnt-nfy	<email-address>	Необязательный, много значений
tech-c	<nic-handle>	Обязательный, много значений
admin-c	<nic-handle>	Необязательный, много значений
remarks	<free-form>	Необязательный, много значений
notify	<email-address>	Необязательный, много значений
mnt-by	Список <mntner-name>	Обязательный, много значений
changed	<email-address> <date>	Обязательный, много значений
source	<registry-name>	Обязательный, одно значение

Рисунок 1. Атрибуты класса mntner.

3.1 Класс mntner

Класс mntner задаёт идентификационную информацию, требуемую для создания, удаления или изменения объектов RPSL. Прежде, чем провайдер сможет создать тот или иной объект RPSL, он должен создать объект mntner. Атрибуты класса mntner показаны на рисунке 1. Класс mntner изначально был описан в документе [13].

Атрибут mntner является обязательным и является ключом класса. Значение этого атрибута является именем RPSL. Атрибут auth задаёт схему, используемую для идентификации и аутентификации запросов на обновление от держателя (maintainer). Этот атрибут использует синтаксис

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

Например,

```
auth: NONE
auth: CRYPT-PW dhjsdfhruewf
auth: MAIL-FROM .*@ripe\.net
```

В качестве вариантов аутентификации <scheme-id> определены: NONE, MAIL-FROM, PGP-KEY и CRYPT-PW. <auth-info> содержит дополнительную информацию, требуемую для конкретной схемы. В случае MAIL-FROM это регулярное выражение, соответствующее корректным почтовым адресам, для CRYPT-PW это пароль в формате UNIX crypt, для PGP-KEY - указатель на объект key-certif [22], содержащий открытый ключ PGP пользователя. Если задано множество атрибутов auth, запрос на обновление выполняется с любой из указанных форм аутентификации.

Атрибут upd-to представляет собой адрес электронной почты. При попытке несанкционированного изменения объекта по этому адресу отправляется соответствующее сообщение. По адресу, заданному атрибутом mnt-nfy пересылаются сообщения в тех случаях, когда поддерживаемый объект добавляется, удаляется или изменяется.

Атрибут descr содержит короткое текстовое описание объекта (в произвольной форме). Атрибут tech-c содержит NIC-идентификатор лица, отвечающего за технические вопросы. С этим человеком контактируют при возникновении технических проблем (например, некорректная конфигурация). Атрибут admin-c указывает NIC-идентификатор администратора. В атрибуте remarks содержится текстовое разъяснение в произвольной форме. Атрибут notify указывает адрес электронной почты, по которому отправляются уведомления об изменении объекта. Атрибут mnt-by содержит список имён объектов типа mntner. Полномочия по изменению данного объекта предоставляются всем указанным в списке держателям. В атрибуте changed указываются два последних изменения объекта с датами внесения изменений. Этот атрибут использует синтаксис

```
changed: <email-address> <YYYYMMDD>
```

Например,

```
changed: johndoe@terabit-labs.nn 19900401
```

Адрес <email-address> идентифицирует человека, внесшего последнее изменение, <YYYYMMDD> указывает дату изменения. Атрибут source указывает реестр, в котором зарегистрирован объект. Пример объекта mntner показан на рисунке 2 для случая использования аутентификации по паролю в формате UNIX crypt.

```
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
```

Рисунок 2. Пример объекта mntner.

Атрибуты descr, tech-c, admin-c, remarks, notify, mnt-by, changed и source используются для всех классов RPSL. Их синтаксис, семантика, обязательность или необязательность, однозначность или многозначность для всех классов RPSL одинаковы. Единственным исключением является атрибут admin-c, который является обязательным для класса aut-num и необязательным для остальных. Упомянутые атрибуты уже не будут рассматриваться в соответствующих классам параграфах.

3.2 Класс person

Класс person служит для описания информации о людях. Хотя эта информация не относится напрямую к политике маршрутизации, мы кратко рассмотрим её здесь, поскольку многие объекты политики содержат ссылки на объекты класса person. Класс person был введён и описан в документе [15].

Атрибут	Значение	Тип
person	<free-form>	Обязательный, одно значение
nic-hdl	<nic-handle>	Обязательный, одно значение, ключ класса
address	<free-form>	Обязательный, много значений
phone	см. ниже	Обязательный, много значений
fax-no	см. ниже	Необязательный, много значений
e-mail	<email-address>	Обязательный, много значений

Рисунок 3. Атрибуты класса person.

Атрибуты класса person показаны на рисунке 3. Атрибут person указывает полное имя человека. Атрибуты phone и fax-no используют синтаксис

```
phone: +<country-code> <city> <subscriber> [ext. <extension>]
```

Например,

```
phone: +31 20 12334676
phone: +44 123 987654 ext. 4711
```

На рисунке 4 показан пример объекта 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
```

Рисунок 4. Пример объекта person.

3.3 Класс role

Класс role похож на класс person, но вместо контактных данных описывает роль одного или нескольких человек. Примерами ролей являются службы поддержки (help desk), центры мониторинга сетей (network monitoring center), системные администраторы (system administrator) и т. п. Объекты role весьма полезны, поскольку исполняющий роль человек может смениться, а сама роль останется.

Атрибут	Значение	Тип
role	<free-form>	Обязательный, одно значение
nic-hdl	<nic-handle>	Обязательный, одно значение, ключ класса
trouble	<free-form>	Необязательный, много значений
address	<free-form>	Обязательный, много значений
phone	см. ниже	Обязательный, много значений
fax-no	см. ниже	Необязательный, много значений
e-mail	<email-address>	Обязательный, много значений

Рисунок 5. Атрибуты класса person.

Атрибуты класса role показаны на рисунке 5. Атрибуты nic-hdl для классов person и role используют общее пространство имён. Атрибут trouble объекта role может содержать дополнительную контактную информацию, которая может использоваться при возникновении проблем на любом из объектов, ссылающихся на данный объект role. Пример объекта role показан на рисунке 6.

```
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
```

Рисунок 6. Пример объекта role.

4 Класс route

Каждый маршрут между автономными системами (interAS[1]), порождённый AS, задаётся с использованием объекта route. Атрибуты класса route показаны на рисунке 7. Атрибут route представляется адресным префиксом маршрута, а атрибут origin указывает номер автономной системы, которая порождает маршрут и передает в систему междоменной маршрутизации. Пара атрибутов route и origin определяет ключ класса.

Атрибут	Значение	Тип
route	<address-prefix>	Обязательный, одно значение, ключ класса
origin	<as-number>	Обязательный, одно значение, ключ класса
member-of	Список <route-set-names>	Необязательный, много значений (см. раздел 5)
inject	см. раздел 8	Необязательный, много значений
components	см. раздел 8	Необязательный, одно значение
aggr-bndry	см. раздел 8	Необязательный, одно значение
aggr-mtd	см. раздел 8	Необязательный, одно значение
export-comps	см. раздел 8	Необязательный, одно значение
holes	см. раздел 8	Необязательный, много значений

Рисунок 7. Атрибуты класса person.

На рисунке 8 показан пример с 4 объектами класса route (контактные атрибуты типа admin-c и tech-c для краткости опущены). Отметим, что два последних объекта route имеют одинаковый адресный префикс 128.8.0.0/16. Однако эти объекты различаются, поскольку они происходят из разных AS (т. е., имеют разные ключи).

```
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
```

Рисунок 8. Объекты класса route.

5 Классы множеств

При задании правил часто бывает работать с наборами (множествами) объектов. Для этого определены классы as-set, route-set, rtr-set, filter-set и peering-set, как именованные множества. Компоненты каждого множества могут быть заданы напрямую, путём перечисления в определении множества, опосредованно, за счёт ссылки на множество в объектах-компонентах, или с помощью комбинации обоих методов.

В качестве имён множеств в RPSL используются слова, начинающиеся с определённого здесь префикса:

- имена наборов AS - «as-»;
- имена наборов маршрутов - «rs-»;
- имена наборов маршрутизаторов - «rtrs-»;
- имена наборов фильтров - «fltr-»;
- имена наборов партнёров - «prng-».

Например, as-foo будет корректным именем для множества автономных систем.

Имена множеств могут иметь иерархическую структуру. Иерархическое имя представляет собой последовательность имён множеств и номеров AS, разделённых двоеточием «:». По крайней мере одна компонента иерархического имени должна быть именем реального множества (т. е., начинаться с одного из перечисленных выше префиксов). Все имена множеств в иерархическом имени должны относиться к одному типу. Например, имена AS1:AS-CUSTOMERS, AS1:RS-EXPORT:AS2, RS-EXCEPTIONS:RS-BOGUS являются корректными иерархическими именами.

Цель введения иерархических имён заключается в разделении пространства имён множеств таким образом, чтобы держатель набора X1 контролировал все пространство производных имён (т. е., X1:...:Xn-1). Таким образом, объект-множество с именем X1:...:Xn-1:Xn может создать только держатель объекта с именем X1:...:Xn-1. Только держатель AS1 сможет создать множество с именем AS1:AS-FOO, а создание множества с именем AS1:AS-FOO:AS-BAR доступно будет только держателю объекта AS1:AS-FOO. Более подробно этот вопрос рассмотрен в документе [20].

[1]Их называют также междоменными маршрутами (interdomain route).

5.1 Класс as-set

Атрибут	Значение	Тип
as-set	<object-name>	Обязательный, одно значение, ключ класса
members	Список <as-numbers> и <as-set-names>	Необязательный, много значений
mbrs-by-ref	Список <mntner-names>	Необязательный, много значений

Рисунок 9. Атрибуты класса as-set.

Атрибуты класса as-set показаны на рисунке 9. Атрибут as-set определяет имя множества (имя RPSL, начинающееся префиксом «as-»). Атрибут members задаёт список элементов множества и может включать номера AS и имена других множеств класса as-set.

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

Рисунок 10. Объекты класса as-set.

На рисунке 10 показаны два объекта класса as-set. Множество as-foo включает две AS (AS1 и AS2), множество as-bar включает в себя множество as-foo и автономную систему AS3 (таким образом, в состав as-foo входят AS1, AS2, AS3), множество as-empty не включает ни одного элемента.

Атрибут mbrs-by-ref содержит список имён держателей или ключевое слово ANY. При использовании этого атрибута во множество AS включаются также автономные системы, чьи объекты aut-num зарегистрированы одним из указанных в списке держателей, а атрибуты member-of содержат имя данного множества AS. Если mbrs-by-ref имеет значение ANY, все объекты AS, ссылающиеся на это множество, являются членами множества. Если атрибут mbrs-by-ref отсутствует, во множество входят только автономные системы, перечисленные в атрибуте members.

```
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
```

Рисунок 11. Объект класса as-set.

На рисунке 11 приведён пример объекта as-set с атрибутом mbrs-by-ref. Множество as-foo содержит AS1, AS2 и AS3. AS4 не входит во множество as-foo, несмотря на то, что объект aut-num ссылается на as-foo. Это обусловлено тем, что MNTR-OTHER не включён в атрибут mbrs-by-ref множества as-foo.

5.2 Класс route-set

Атрибут	Значение	Тип
route-set	<object-name>	Обязательный, одно значение, ключ класса
members	Список <address-prefix-range>, <route-set-name> и <route-set-name><range-operator>	Необязательный, много значений
mbrs-by-ref	Список <mntner-names>	Необязательный, много значений

Рисунок 12. Атрибуты класса route-set.

Атрибуты класса route-set показаны на рисунке 12. Атрибут route-set задаёт имя множества (имя RPSL с префиксом «rs-»). Атрибут members содержит список элементов множества и может включать адресные префиксы и имена других множеств класса route-set. Отметим, что класс route-set определяет множество префиксов маршрутов, а не объектов route.

```
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
```

Рисунок 13. Объекты класса route-set.

На рисунке 13 показан пример объектов route-set. Множество rs-foo содержит два адресных префикса 128.9.0.0/16 и 128.9.0.0/24. Множество rs-bar включает элементы множества rs-foo и префикс 128.7.0.0/16.

За адресным префиксом или именем route-set в атрибуте может следовать оператор диапазона. Например, множество

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

содержит все префиксы, более специфичные, чем 5.0.0.0/8 и сам префикс 5.0.0.0/8, все более специфичные префиксы в 30.0.0.0/8, размер которых составляет от 24 до 32 (такие, как 30.9.9.96/28), а также все более специфичные префиксы для списка из множества rs-foo.

Атрибут mbrs-by-ref содержит список имён держателей или ключевое слово ANY. При использовании этого атрибута множество будет включать все адресные префиксы, чьи объекты route зарегистрированы указанными в списке держателями, а атрибут member-of содержит имя этого множества. Если атрибут mbrs-by-ref содержит значение ANY, во множество включаются все объекты route, ссылающиеся на данное множество. Если атрибут mbrs-by-ref отсутствует, во множество включаются только префиксы, перечисленные в списке атрибута members.

На рисунке 14 показан пример объектов route-set с атрибутом mbrs-by-ref. Множество rs-foo содержит два адресных префикса 128.8.0.0/16 b 128.9.0.0/16, поскольку объекты route для 128.8.0.0/16 и 128.9.0.0/16 указывают на имя rs-foo в своём атрибуте member-of. Множество rs-bar содержит префиксы 128.7.0.0/16 и 128.8.0.0/16. Маршрут 128.7.0.0/16 явно указан в атрибуте members записи rs-bar, а объект route для 128.8.0.0/16 указывает на множество с именем rs-bar в своём атрибуте member-of.

Отметим, что для случаев, когда адресный префикс указан в атрибуте members множества route, он является членом данного множества маршрутов. Объект route, соответствующий этому префиксу, не обязан включать атрибут member-

of, указывающий на имя множества. Атрибут member-of класса route является дополнительным механизмом для неявного указания членов класса.

```
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
```

Рисунок 14. Объекты класса route-set.

5.3 Предопределённые объекты-множества

В контексте, который предполагает множество маршрутов (например, атрибут members в классе route-set), автономная система ASx определяет множество маршрутов, исходящих из ASx, а множество (as-set) AS-X определяет набор маршрутов, исходящих из автономных систем в составе AS-X. О маршруте p говорят, что он исходит из ASx, если существует объект класса route для p с ASx в качестве значения атрибута origin. Например, множество маршрутов rs-special на рисунке 15 содержит префикс 128.9.0.0/16, маршруты AS1 и AS2, а также маршруты автономных систем из множества AS-FOO.

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

Рисунок 15 Использование номеров AS и наборов AS в наборе маршрутов

Предопределённое множество rs-any содержит все маршруты, зарегистрированные в IRR, а множество as-any - все автономные системы, зарегистрированные в IRR.

5.4 Фильтры и класс filter-set

Атрибуты класса filter-set показаны на рисунке 16. Объект filter-set определяет множество маршрутов, которые соответствуют фильтрам данного объекта. Атрибут filter-set определяет имя фильтра (имя RPSL с префиксом «fltr-»).

Атрибут	Значение	Тип
filter-set	<object-name>	Обязательный, одно значение, ключ класса
filter	<filter>	Обязательный, одно значение

Рисунок 16. Атрибуты класса filter-set.

Атрибут filter определяет политику фильтрации для множества. Фильтр представляет собой логическое выражение, которое применяется к набору маршрутов для выделения из него подмножества маршрутов, соответствующих политике фильтрации. В фильтрах может проверяться соответствие любых атрибутов пути BGP (таких, префикс адресата или NLRI, AS-path, атрибуты группы - community).

Фильтры политики могут объединяться с помощью логических операторов AND (и), OR (или), NOT (не). Для выбора подмножества маршрутов могут использоваться перечисленные ниже фильтры.

ANY

Ключевому слову ANY соответствует любой маршрут.

Address-Prefix Set

Явный список адресных префиксов, заключённых в фигурные скобки {}. Этому фильтру соответствуют маршруты, для которых адресные префиксы получателей находятся в списке. Примеры списков приведены ниже.

```
{ 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 }
{ }
```

С адресным префиксом может использоваться оператор диапазона. Например,

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

содержит все префиксы, более специфичные, чем 5.0.0.0/8, и сам префикс 5.0.0.0/8, все префиксы, более специфичные, чем 128.9.0.0/16, без включения 128.9.0.0/16, все префиксы, более специфичные, чем 30.0.0.0/8, размер которых равен 16 (скажем, 30.9.0.0/16), а также все префиксы, более специфичные, чем 30.0.0.0/8, с размером от 24 до 32 (например, 30.9.9.96/28).

Route Set Name

Имени набора маршрутов соответствует множество маршрутов, являющихся членами этого набора. Имя набора маршрутов может быть именем объекта route-set, номером AS или именем объекта as-set (номера AS и имена as-set неявно определяют наборы маршрутов - см. параграф 5.3). Например,

```
aut-num: AS1
import: from AS2 accept AS2
import: from AS2 accept AS-FOO
import: from AS2 accept RS-FOO
```

Вместо номера партнерской автономной системы может использоваться ключевое слово PeerAS. Особенно полезно это в тех случаях, когда партнерские отношения задаются с использованием AS-выражения. Например,

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

aut-num: AS1
import: from AS-FOO accept PeerAS
```

эквивалентно записи

```
aut-num: AS1
import: from AS2 accept AS2
```

```
import: from AS3 accept AS3
```

За именем набора маршрутов может указываться один из операторов «^-» или «^+» (например, { 5.0.0.0/8, 6.0.0.0/8 }^+ эквивалентно { 5.0.0.0/8^+, 6.0.0.0/8^+ }, а AS1^- эквивалентно всем более специфичным маршрутам, исходящим из AS1).

AS Path Regular Expression

Регулярные выражения AS-path могут использоваться в качестве фильтров политики путём заключения выражения в угловые скобки «<» и «>». Фильтру AS-path соответствует набор маршрутов, проходящих через последовательность AS, соответствующих регулярному выражения AS-path. Маршрутизатор может проверить соответствие, используя атрибут AS_PATH протокола BGP[1] [19] или атрибут RD_PATH протокола IDRP[2] [18].

Регулярные выражения AS-path соответствуют стандарту POSIX для регулярных выражений в «алфавите» номеров AS. Регулярные выражения могут включать перечисленные ниже элементы.

ASN

Номер автономной системы. ASN соответствует AS-path размером 1, содержащий указанный номер AS (например, регулярному выражению AS-path вида AS1 соответствует AS-path «1»).

Вместо номера партнерской AS может использоваться ключевое слово PeerAS.

AS-set

Имя набора AS. Значению AS-set соответствуют все AS-path, соответствующие одной из AS в AS-set.

.

Соответствует путям AS-path, которые соответствуют любому номеру AS.

[...]

Набор номеров AS. Такому набору соответствуют пути AS-path, соответствующие номерам AS, указанным в скобках. Номера AS в наборе разделяются символами пробела. Символ "-" между двумя номерами AS в этом списке указывает, что все номера интервала включены в набор. Имя as-set задаёт включение всех AS из as-set.

[^...]

Дополнение к набору номеров AS. Ему соответствует любой AS-path, не соответствующих номерам AS в наборе.

^

Соответствует пустой строке в начале AS-path.

$

Соответствует пустой строке в конце AS-path.

Далее перечислены операторы регулярных выражений в порядке снижения очерёдности выполнения (операторы в записи выполняются слева направо).

Унарные суффиксы * + ? {m} {m,n} {m,}

Для регулярного выражения A, A* будет соответствовать 0 или более вхождений A; A+ - 1 или более вхождений A; A? - 0 или 1 вхождение A; A{m} - m вхождений A; A{m,n} - от m до n вхождений A; A{m,} - m или более вхождений A. Например, выражению [AS1 AS2]{2} соответствуют AS1 AS1, AS1 AS2, AS2 AS1 и AS2 AS2.

Унарные суффиксы ~* ~+ ~{m} ~{m,n} ~{m,}

Эти операторы похожи по функциональности на перечисленные выше, но все вхождения регулярного выражения соответствуют одному шаблону. Например, выражению [AS1 AS2]~{2} соответствуют AS1 AS1 и AS2 AS2, но не соответствуют AS1 AS2 и AS2 AS1.

Бинарный оператор конкатенации

Этот неявный оператор существует между двумя регулярными выражениями A и B, когда не задано явно других операторов. Результирующему регулярному выражению A B соответствует путь AS-path, если некая часть этого пути соответствует A, а оставшаяся часть AS-path соответствует B.

Бинарный оператор «или» |

Для регулярных выражений A и B, записи A | B будет соответствовать любой путь AS-path, который соответствует A или B.

Для смены порядка применения операторов могут использоваться скобки. Для придания более выразительной формы в регулярных выражениях могут использоваться дополнительные пробелы.

Ниже приведены примеры фильтров AS-path:

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

Первому выражению соответствуют все маршруты, в которых AS-path включает AS3, второму соответствуют маршруты, в которых AS-path начинается с AS1, третьему - маршруты, в которых AS-path заканчивается в AS2, четвёртому - маршруты с AS-path «1 2 3», а пятому - маршруты, в которых AS-path начинается с AS1 и заканчивается в AS2, включая между ними произвольные AS.

Композитные фильтры политики

Ниже перечислены (в порядке снижения очерёдности исполнения) операторы, с помощью которых можно создавать композитные фильтры политики.

NOT

Для данного фильтра x, выражению NOT x будет соответствовать набор маршрутов, которые не соответствуют x. Т. е., этот оператор задаёт отрицание политики фильтра x.

AND

Для данных фильтров x и y, выражению x AND y будет соответствовать пересечение маршрутов, соответствующих фильтрам x и y.

OR

Для данных фильтров x и y, выражению x OR y будет соответствовать объединение маршрутов, соответствующих фильтрам x и y.

Отметим, что оператор OR может использоваться в неявном виде («x y» взамен «x OR y»).

Например,

```
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}
```

[1]Border Gateway Protocol.
[2]Inter-Domain Routing Protocol.

Первому выражению будут соответствовать все маршруты, за исключением 128.9.0.0/16 и 128.8.0.0/16. Второму выражению будут соответствовать маршруты AS226, AS227 и AS228. Третьему выражению будут соответствовать маршруты AS226, за исключением 128.9.0.0/16. Четвёртому выражению будут соответствовать маршруты AS226, длина которых не превышает 18.

Атрибуты политики маршрутизации

В фильтрах политики для сравнения могут использоваться также значения других атрибутов. Атрибуты, значения которых могут использоваться для сравнения, указаны в словаре RPSL (см. раздел 7). Пример использования атрибута BGP community показан ниже:

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

Фильтры, использующие атрибуты политики маршрутизации, которые определены в словаре, применяются до выполнения операторов AND, OR и NOT.

Имена наборов фильтров

Имени набора фильтров соответствует набор маршрутов, соответствующих атрибутам фильтров. Отметим, что атрибут фильтра из набора фильтров может рекурсивно использовать имена других наборов фильтров. Например, на рисунке 17 набору fltr-foo соответствует { 5.0.0.0/8, 6.0.0.0/8 }, а набору fltr-bar соответствуют маршруты AS1 или { 5.0.0.0/8, 6.0.0.0/8 }, если для них AS-path включает AS2.

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

filter-set: fltr-bar
filter: (AS1 or fltr-foo) and <AS2>
```

Рисунок 17. Объекты класса filter-set.

5.5 Класс rtr-set

Атрибут	Значение	Тип
rtr-set	<object-name>	Обязательный, одно значение, ключ класса
members	список <inet-rtr-name>, <rtr-set-name> или <ipv4_address>	Необязательный, много значений
mbrs-by-ref	список <mntner-name>	Необязательный, много значений

Рисунок 18. Атрибуты класса rtr-set.

Атрибуты класса rtr-set показаны на рисунке 18. Атрибут rtr-set определяет имя набора - имя RPSL, начинающееся с «rtrs-». Атрибут members содержит список элементов набора, которые указываются в виде списка имён inet-rtr, адресов ipv4_address или имён других наборов rtr-set.

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

Рисунок 19. Объекты rtr-set.

На рисунке 19 показаны два объекта rtr-set. Набор rtrs-foo включает два маршрутизатора - rtr1.isp.net и rtr2.isp.net. Набор rtrs-bar содержит объекты набора rtrs-foo и маршрутизатор rtr3.isp.net (т. е., маршрутизаторы rtr1.isp.net, rtr2.isp.net, rtr3.isp.net).

Атрибут mbrs-by-ref содержит список имён держателей или ключевое слово ANY. При использовании этого атрибута набор маршрутизаторов включает также маршрутизаторы, для которых объекты inet-rtr зарегистрированы одним из указанных держателей, а атрибут member-of указывает на данный набор маршрутизаторов. Если mbrs-by-ref = ANY, любой объект inet-rtr, ссылающийся на данный набор маршрутизаторов, является элементом набора. Если атрибут mbrs-by-ref отсутствует, в набор входят только маршрутизаторы, указанные в списке атрибута members.

```
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
```

Рисунок 20. Объекты rtr-set.

На рисунке 20 показан объект rtr-set, использующий атрибут mbrs-by-ref. Набор rtrs-foo включает маршрутизаторы rtr1.isp.net, rtr2.isp.net и rtr3.isp.net.

5.6 Партнерство и класс peering-set

Атрибуты класса peering-set показаны на рисунке 21. Объект peering-set определяет множество партнёров, которые перечислены в его атрибутах, а также имя множества (имя RPSL, начинающееся с «prng-»).

Атрибут	Значение	Тип
peering-set	<object-name>	Обязательный, одно значение, ключ класса
peering	<peering>	Обязательный, много значений

Рисунок 21. Атрибуты класса filter.

Атрибут peering определяет партнёров, которые могут использоваться для импорта или экспорта маршрутов.

При описании партнерских отношений здесь используется топология, показанная на рисунке 22. В этом примере используется три AS (AS1, AS2, AS3), две точки обмена (EX1, EX2) и 6 маршрутизаторов. Маршрутизаторы, подключённые к одной точке обмена, соединены между собой (каждый с каждым) и обмениваются маршрутной информацией. Т. е., маршрутизаторы 7.7.7.1, 7.7.7.2 и 7.7.7.3 соединены между собой, а маршрутизаторы 9.9.9.1, 9.9.9.2 и 9.9.9.3 - между собой.

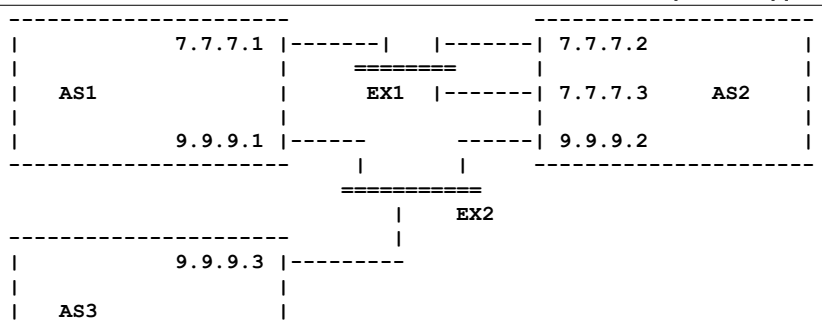

Рисунок 22. Пример топологии с тремя AS - AS1, AS2 и AS3, двумя точками обмена и шестью маршрутизаторами.

Синтаксис спецификации партнерских отношений имеет вид

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

где <as-expression> - выражение на основе номеров AS и наборов AS с использованием операторов AND, OR и EXCEPT, а <router-expression-1> и <router-expression-2> - выражения на основе IP-адресов маршрутизаторов, имён inet-rtr и rtr-set с использованием операторов AND, OR и EXCEPT. Бинарный оператор EXCEPT задаёт вычитание (исключение) множества и имеет такой же приоритет, как оператор AND (семантически он эквивалентен комбинации AND NOT). Т. е., (AS1 OR AS2) EXCEPT AS2 = AS1.

Такая запись идентифицирует все партнерские отношения между любым локальным маршрутизатором из <router-expression-2> и всеми его партнёрами в <router-expression-1>, относящимися к AS из <as-expression>. Если выражение <router-expression-2> не указано, по умолчанию это будет указывать на все маршрутизаторы локальной AS, которые имеют партнёров в AS из <as-expression>. Если не задано выражение <router-expression-1>, по умолчанию это будет указывать на все маршрутизаторы партнерских AS из <as-expression>, которые имеют партнерские отношения с локальной AS.

Если используется <peering-set-name>, партнерские отношения перечисляются в соответствующем объекте peering-set. Отметим, что объекты peering-set могут быть рекурсивными.

Возможно множество специальных форм указания партнерских отношений. Приведённые ниже примеры показывают наиболее распространённые варианты, использующие атрибут import класса aut-num. В первом примере 7.7.7.1 импортирует маршрут 128.9.0.0/16 от 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 }
```

В следующем примере маршрутизатор 7.7.7.1 импортирует 128.9.0.0/16 от 7.7.7.2 и 7.7.7.3.

(2) aut-num: AS1

```
    import: from AS2 at 7.7.7.1 accept { 128.9.0.0/16 }
```

В примере 3 маршрутизатор 7.7.7.1 импортирует 128.9.0.0/16 от 7.7.7.2 и 7.7.7.3, а 9.9.9.1 - 128.9.0.0/16 от 9.9.9.2.

(3) aut-num: AS1

```
    import: from AS2 accept { 128.9.0.0/16 }
```

В следующем примере 9.9.9.1 импортирует маршрут 128.9.0.0/16 от 9.9.9.2 и 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 }
```

В примере 5 маршрутизатор 9.9.9.1 импортирует 128.9.0.0/16 от 9.9.9.2 и 9.9.9.3, а 7.7.7.1 - маршрут 128.9.0.0/16 от 7.7.7.2 и 7.7.7.3.

(5) aut-num: AS1

```
    import: from AS-FOO                  accept { 128.9.0.0/16 }
```

В следующем примере AS1 импортирует маршрут 128.9.0.0/16 от AS3 на маршрутизаторе 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 }
```

Выражение «AS-FOO and not AS2» эквивалентно «AS3», а «not 7.7.7.1» = 9.9.9.1.

В примере 7 маршрутизатор 9.9.9.1 импортирует 128.9.0.0/16 от 9.9.9.2 и 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 Класс aut-num

Атрибут	Значение	Тип
aut-num	<as-number>	Обязательный, одно значение, ключ класса
as-name	<object-name>	Обязательный, одно значение
member-of	список <as-set-name>	Необязательный, много значений
import	см. параграф 6.1	Необязательный, много значений
export	см. параграф 6.2	Необязательный, много значений
default	см. параграф 6.5	Необязательный, много значений

Рисунок 23. Атрибуты класса aut-num.

Правила маршрутизации задаются с использованием класса aut-num, атрибуты которого показаны на рисунке 23. Значением атрибута aut-num является номер AS, описываемой данным объектом. Атрибут as-name показывает символьное имя AS (с использованием синтаксиса имён RPSL). Правила импорта, экспорта и маршрутизации по умолчанию для AS описываются атрибутами import, export и default, соответственно.

6.1 Атрибут import - спецификация правил импорта

В RPSL политика импорта определяется выражениями правил импорта, каждое из которых задаётся с использованием атрибута import. Синтаксис атрибута import показан ниже и более подробно описан в параграфах 6.3 и 6.6.

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

Указание действия (action) является необязательным. Семантика атрибута import следующая: набор маршрутов, соответствующих фильтру <filter>, импортируется от всех партнёров из <peerings>, а импортируемые маршруты из <peering-M>, <action-M> «выполняются».

Например, запись

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

говорит о том, что маршрут 128.9.0.0/16 принимается от AS2 с приоритетом 1. Спецификации партнерских отношений и фильтров были описаны ранее (см. параграфы 5.6 и 5.4, соответственно). Далее описана спецификация действий.

6.1.1 Спецификация действий

Действиями правил в RPSL являются установка или изменение атрибутов маршрута (например, задание приоритета, добавление группы BGP в атрибут пути BGP community или установка атрибута MULTI-EXIT-DISCRIMINATOR). Действия правил могут включать также инструкции маршрутизатору для выполнения специальных операций (например, подавления осцилляций маршрутов - route flap damping).

Атрибуты политики маршрутизации, значения которых могут меняться действиями политики, указаны в словаре RPSL (список этих атрибутов приведён в разделе 7). Каждое действие в RPSL завершается символом «;». Можно формировать составные действия политики, перечисляя одно действие за другим. В составных действиях элементарные акции выполняются слева направо. Например, запись

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

будет задавать установку pref = 10, med = 0, а также добавления 10250 и 3561:10 в конец атрибута пути BGP community. Атрибут pref является инверсией атрибута local-pref (т. е., local-pref = 65535 - pref). Маршрут с атрибутом local-pref всегда является более предпочтительным, чем маршрут без такого атрибута.

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

приведённый выше пример показывает, что маршруты AS4 воспринимаются от AS2 с приоритетом 1, а от AS3 - с приоритетом 2 (маршруты с меньшим значением приоритета являются более предпочтительными).

```
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
```

Пример выше показывает, что маршруты AS4 принимаются от AS2 через партнерскую связь 7.7.7.1-7.7.7.2 с приоритетом 1, а от других партнёров из AS2 — с приоритетом 2.

6.2 Атрибут export - спецификация политики экспорта

Для задания политики экспорта служат выражения правил экспорта с использованием атрибута export, синтаксис которого показан ниже.

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

Задание действия (action) является необязательным. Атрибут export имеет следующую семантику: набор маршрутов, соответствующих фильтру <filter>, экспортируется всем партнёрам, заданным в <peerings>, маршруты в <peering-M>, <action-M> «выполняются».

Например, запись

```
aut-num: AS1
```

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

показывает, что маршруты AS4 анонсируются в AS2 с med = 5 и добавлением группы 70 в список community.

Пример

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

показывает, что AS1 анонсирует все свои маршруты в набор AS с именем AS-FOO.

6.3 Другие протоколы маршрутизации, многопротокольной маршрутизации и вставки маршрутов

Более полный синтаксис атрибутов import и export показан ниже.

```
    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>
```

Необязательные спецификации протокола могут использоваться при задании правил для других протоколов маршрутизации, для вставки маршрутов одного протокола в маршруты другого протокола или политики многопротокольной маршрутизации. Поле <protocol-1> задаёт имя протокола, маршруты которого будут передаваться другим протоколам. Поле <protocol-2> указывает имя протокола, который будет принимать маршруты от другого протокола. По умолчанию для полей <protocol-1> и <protocol-2> используется протокол внешней маршрутизации Internet, которым в настоящее время является BGP.

В приведённом ниже примере все маршруты interAS передаются протоколу RIP.

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

В следующем примере AS1 принимает маршруты AS2, включая все более специфичные маршруты AS2, но не вставляет эти более специфичные маршруты в таблицы маршрутизации OSPF.

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

В следующем примере AS1 вставляет свои статические маршруты (маршруты из набора AS1:RS-STATIC-ROUTES) в протокол маршрутизации interAS и дважды добавляет AS1 в конец своих путей AS.

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

В заключительном примере AS1 импортирует другой набор unicast-маршрутов для обратного пути пересылаемых пакетов multicast от AS2:

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

6.4 Устранение неоднозначностей

Возможны ситуации, когда одни и те же партнерские отношения будут указаны в нескольких спецификациях партнёрства в выражениях политики. Например,

```
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
```

Такая ситуация не является ошибкой, хотя, определённо, нежелательна. Для устранения неоднозначности используется действие, соответствующее первой спецификации партнёрства. В примере это маршруты, принятые с приоритетом 2. Это правило будем называть specification-order[1].

Рассмотрим пример:

```
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
```

где обе спецификации покрывают партнерство 7.7.7.1-7.7.7.2, хотя вторая спецификация является более узкой (специфичной). Правило порядка спецификаций остаётся применимым и выполняется только действие pref = 2. Фактически, вторая пара peering-action не используется, поскольку первая пара peering-action всегда перекрывает её. Если целью правила является восприятие маршрутов с приоритетом 1 для данных партнерских отношений и с приоритетом 2 от всех других партнёров, следует задать спецификацию:

[1]Порядок спецификаций.

```
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
```

Возможны также случаи, когда более одного выражения политики будет перекрывать один набор маршрутов для одних партнерских отношений, как показано в примере.

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

В этом случае правило specification-order также действует. Т. е., маршруты AS4 будут восприниматься от AS2 с приоритетом 2. Если фильтры перекрываются, но не совпадают:

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

маршруты AS4 воспринимаются от AS2 с приоритетом 2, однако маршруты AS5 тоже будут приниматься, но с приоритетом 1.

Ниже приводится в общем виде спецификация правила порядка для удобства реализации RPSL. Рассмотрим два выражения политики:

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

Приведённые выше выражения эквиваленты следующему однозначному набору выражений:

```
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
```

где отношения peerings-3 покрываются одновременно peerings-1 и peerings-2, а peerings-4 - перекрывается peerings-2, но не перекрывается peerings-1 (filter-2 AND NOT filter-1 соответствует маршрутам, которые соответствуют filter-2, но не соответствуют filter-1).

Пример:

```
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}
```

Рассмотрим два партнёрства с AS2 - 7.7.7.1-7.7.7.2 и 9.9.9.1-9.9.9.2. Оба выражения правил покрывают 7.7.7.1-7.7.7.2. В этом партнёрстве маршрут 128.9.0.0/16 воспринимается с приоритетом 2, а маршрут 75.0.0.0/8 - с приоритетом 1. Партнерство 9.9.9.1-9.9.9.2 покрывается только вторым выражением. Следовательно, оба маршрута 128.9.0.0/16 и 75.0.0.0/8 будут приниматься с приоритетом 1 через партнерские отношения 9.9.9.1-9.9.9.2.

Отметим, что такие же правила устранения неоднозначностей применимы к выражениям правил для export и default.

6.5 Атрибут default - задание политики, используемой по умолчанию

Используемые по умолчанию правила маршрутизации задают с помощью атрибута default, синтаксис которого показан ниже:

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

Поля <action> и <filter> являются необязательными. Атрибут имеет следующую семантику: спецификация <peering> показывает AS (и маршрутизатор, если он указан) для маршрута по умолчанию; спецификация <action> (при её наличии) указывает различные атрибуты для маршрутизации по умолчанию (например, относительные предпочтения разных маршрутов, используемых по умолчанию), а спецификация <filter> (при её наличии) указывает правила фильтрации. Маршрутизатор использует политику маршрутизации по умолчанию только при получении от данного партнёра маршрутов, соответствующих заданному фильтру <filter>.

В приведённом ниже примере AS1 использует по умолчанию AS2 для маршрутизации.

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

В следующем примере маршрутизатор 7.7.7.1 в AS1 использует по умолчанию маршрутизатор 7.7.7.2 из AS2.

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

В приведённом ниже примере AS1 использует по умолчанию AS2 и AS3, предпочитая AS2.

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

В следующем примере AS1 использует по умолчанию AS2, применяя 128.9.0.0/16 в качестве заданной по умолчанию сети.

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

6.6 Структурированная спецификация политики

Правила импорта и экспорта могут быть структурированными. Рекомендуется использовать структурирование политики только подготовленным пользователям RPSL. Если вы не планируете использовать структурирование, этот раздел можно пропустить.

Для задания структурированной политики используется показанный ниже синтаксис.

```
<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>
```

Следует отметить точку с запятой в конце <import-factor>. Если спецификация политики не структурирована (как во всех примерах других параграфов), этот знак является необязательным (синтаксис и семантика <import-factor> определены в параграфе 6.1).

Поле <import-term> содержит последовательность значений <import-factor>, заключённых в фигурные скобки ({ }) или просто <import-factor>. Семантика <import-term> заключается в объединении <import-factor> с использованием правила порядка спецификаций. Поле <import-expression> представляет собой <import-term> или <import-term>, за которым следует одно из ключевых слов except или refine и другое выражение <import-expression>. Отметим, что наше определение допускает вложенные выражения. Следовательно, возможны «исключения из исключений», «уточнение уточнённого» и даже «уточнение исключений» и т. п.

Оператор исключения (except) имеет следующую семантику: результатом использования операции except является другое значение <import-term>. Результирующий набор правил содержит правила правой части, но фильтры правил изменяются так, чтобы включались только маршруты, соответствующие левой части. После этого включаются правила левой части, а их фильтры изменяются для исключения маршрутов, соответствующих правой части. Обратите внимание на то, что в процессе обработки фильтры изменяются, но действия копируются без изменений. При наличии множества вложенных уровней операции (как except, так и refine) выполняются справа налево.

Рассмотрим пример.

```
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};
           }
        }
```

где маршрут 128.9.0.0/16 исходит из AS226, являющейся элементом набора as-foo. В этом примере маршрут 128.9.0.0/16 воспринимается от AS3, все прочие маршруты (не 128.9.0.0/16), порождённые AS226, воспринимаются от AS2, а маршруты из других AS в as-foo воспринимаются от AS1.

Мы можем сделать такое же заключение, используя описанные выше принципы («алгебру»). Рассмотрим спецификацию внутреннего исключения:

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

её эквивалентом является

```
  {
   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};
  }
```

Следовательно, исходное выражение эквивалентно:

```
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};
        }
```

которое, в свою очередь, эквивалентно

```
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});
   }
```

Поскольку AS226 входит в as-foo и 128.9.0.0/16 относится к AS226, выражение можно упростить до:

```
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;
      }
```

Для оператора refine результирующий набор конструируется путём выполнения декартова перемножения сторон: для каждого правила l в левой части и каждого правила r в правой части партнерство в результирующей политики будет представлять собой партнерство, общее для r и l, фильтр результирующей политики будет определяться пересечением фильтров l и r, а действие результирующей политики будет определяться последовательным выполнением действий l и r в указанном порядке. Если общего партнёрства нет или пересечение фильтров пусто, результирующее правило не создаётся.

Рассмотрим пример

```
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;
        }
```

Здесь любой маршрут с 3560:10 получает приоритет 1, а любой маршрут с 3560:20 - приоритет 2, независимо от источника импорта. Однако из AS1 импортируются только маршруты AS1, из AS2 - только маршруты AS2, из AS3 - только маршруты AS3, а из других AS маршруты не импортируются. Мы можем достичь такого же результата, используя описанную выше «алгебру». Приведённый выше пример эквивалентен:

```
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;
}
```

Отметим, что общее партнерство между «from AS1» и «from AS-ANY» совпадает с партнёрством «from AS1». Хотя понятие «общего партнёрства» (common peerings), формально не было определено, такое определение можно получить дедукцией определённого в параграфе 5.6 понятия партнёрства.

Рассмотрим пример

```
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;
    }
```

где воспринимаются только маршруты, длиной от 0 до 18 и устанавливается значение med = 0 для предотвращения влияния атрибута med на партнерские отношения. В дополнение к этому из AS1 импортируются только маршруты AS1 и маршруты, импортированные из AS1 в 7.7.7.1 являются предпочтительными по сравнению с маршрутами от других партнёров. Это эквивалентно выражению:

```
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;
}
```

Описанные выше синтаксис и семантика применимы также к структурированной политики экспорта при замене from на to и accept на announce.

7 Класс dictionary

Класс dictionary обеспечивает расширяемость RPSL. Объекты словаря определяют атрибуты типы и протоколы маршрутизации для политики маршрутизации. Атрибуты политики маршрутизации (далее rp-attribute) могут соответствовать реальным атрибутам протоколов, таким, как атрибуты пути BGP (например, community, dpa, AS-path), или свойствам маршрутизаторов (например, подавление осцилляций маршрутов BGP). По мере внедрения новых протоколов, протокольных атрибутов или новых возможностей маршрутизаторов объект dictionary обновляется для включения соответствующих определений rp-attribute и протоколов.

Класс rp-attribute является абстрактным (т. е., представление данных для него невозможно). Вместо этого используются различные методы доступа. Например, атрибут rp-attribute для атрибута BGP AS-path называется aspath и имеет метод доступа, называемый prepend (добавление в начало), который служит для добавления номеров AS в атрибуты AS-path. Методы доступа могут использовать аргументы, которые являются строго типизованными. Например, упомянутый выше метод prepend использует в качестве аргументов номера AS.

Как только атрибут rp-attribute определен в словаре, его можно использовать для описания фильтров и действий политики маршрутизации. Для привлечения новых объектов словаря и распознавания вновь определённых атрибутов rp-attribute, типов и протоколов требуются средства анализа политики. Такие средства позволят аппроксимировать анализ политики с атрибутами rp-attribute, которых они ещё не понимают: метод filter может всегда возвращать соответствие, а метод action — не выполнять никаких действий. Средства анализа могут даже загрузить код для выполнения подходящих операций с использованием механизмов, выходящих за пределы RPSL.

Далее будет описан синтаксис и семантика класса dictionary. Это описание не столь важно для понимания объектов класса dictionary (но существенно для их создания). Вы можете пропустить текст до начала параграфа 7.1 Исходный словарь RPSL, примеры действий и фильтров политики.

Атрибуты класса dictionary показаны на рисунке 24. Атрибут dictionary указывает имя объекта dictionary в соответствии с правилами именования RPSL. Может существовать множество объектов dictionary, однако есть один общеизвестный объект словаря по имени RPSL. Все средства по умолчанию используют этот словарь.

Атрибут	Значение	Тип
dictionary	<object-name>	Обязательный, одно значение, ключ класса
rp-attribute	<object-name>	Необязательный, много значений
typedef	список <as-set-name>	Необязательный, много значений
protocol	см. параграф 6.1	Необязательный, много значений

Рисунок 24. Атрибуты класса dictionary.

Атрибут rp-attribute использует синтаксис:

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

где поле <name> указывает имя rp-attribute, <method-i> задаёт имя метода доступа для rp-attribute, принимающего Ni аргументов (j-й аргумент имеет тип <type-i-j>). В качестве имени метода может служить имя RPSL или один из операторов, указанных на рисунке 25. Операторные методы, за исключением operator() и operator[], могут принимать только по 1 аргументу.

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

Рисунок 25. Операторы.

Атрибут rp-attribute может иметь множество определённых для него методов. Некоторые методы могут даже совпадать по именам, если они различаются по типам принимаемых атрибутов. Если за списком атрибутов следует многоточие «...», это говорит о переменном числе аргументов данного метода. В таких случаях реальные аргументы после N-го имеют тип <type-N>.

Аргументы строго типизованы. Тип <type> в RPSL является предопределенным, объединением (union), списком (list) или определенным в словаре типом. Предопределённые типы указаны на рисунке 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
```

Рисунок 26. Предопределенные типы.

Предопределённые типы integer (целое число) и real (действительное число) могут иметь верхнюю и нижнюю границу диапазона допустимых значений аргумента. Задание диапазона является опциональным. Для представления значений типов integer, real и string здесь используются правила языка ANSI C. Для типа enum указывается список имён RPSL, которые являются корректными значениями типа. Тип boolean может принимать значения true (истина) и false (ложь). Типы as_number, ipv4_address, address_prefix и dns_name описаны в разделе 2. Тип filter представляет собой фильтр политики, описанный в разделе 6. Предполагается, что значение типа filter заключается в круглые скобки.

Тип union использует синтаксис:

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

где <type-i> - тип RPSL. Тип union является одним из типов <type-1> - <type-N> (аналогично типу union в C[14]).

Тип list использует синтаксис:

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

В этом случае список элементов относится к типу <type> и содержит не менее <min_elems> и не более <max_elems> элементов. Спецификация размера является опциональной. Если размер не задан, число элементов списка не ограничивается. Значение типа list представляется в форме последовательности элементов, разделённых запятыми, и помещается в фигурные скобки ({ }).

Атрибут typedef в словаре определяет именованные типы:

```
typedef: <name> <type>
```

где <name> - имя типа для <type>. Атрибут typedef особенно полезен для определения типов, не относящихся к предопределенным (например, список union, список list и т. п.).

Атрибут protocol класса dictionary определяет протокол и набор параметров партнёрства для этого протокола (параметры используются в классе inet-rtr, как показано в разделе 9). Атрибут использует синтаксис:

```
protocol: <name>
 MANDATORY | OPTIONAL <parameter-1>(<type-1-1>,...,
                      <type-1-N1> [,"..."])
   ...
 MANDATORY | OPTIONAL <parameter-M>(<type-M-1>,...,
                      <type-M-NM> [,"..."])
```

где <name> - имя протокола, MANDATORY (обязательный) или OPTIONAL (необязательный) - ключевые слова, а <parameter-i> - параметр партнёрства для этого протокола, принимающий Ni аргументов. Синтаксис и семантика аргументов такие же, как для атрибута rp-attribute. При использовании ключевого слова MANDATORY параметр

является обязательным и должен быть задан для каждого партнёрства данного протокола. При использовании ключевого слова OPTIONAL параметр может быть пропущен.

7.1 Исходный словарь RPSL, примеры действий и фильтров политики

```
dictionary:   RPSL
rp-attribute: # приоритет, меньшее значение говорит о предпочтительности
              pref
              operator=(integer[0, 65535])
rp-attribute: # атрибут BGP multi_exit_discriminator
              med
              # для установки med = 10: med = 10;
              # для установки в качестве med значения метрики IGP: med = igp_cost;
              operator=(union integer[0, 65535], enum[igp_cost])
rp-attribute: # атрибут BGP для предпочтения адресатов (dpa)
              dpa
              operator=(integer[0, 65535])
rp-attribute: # атрибут BGP AS-path
              aspath
              # добавление номеров AS в начало (prepend) в порядке от последнего к первому
              prepend(as_number, ...)
typedef:      # значение community в RPSL может быть
              #  - 4-байтовым целым числом (корректная нотация 3561:70)
              #  - internet, no_export, no_advertise (см. RFC 1997)
              community_elm union
                  integer[1, 4294967295],
                  enum[internet, no_export, no_advertise],
typedef:      # список значений community { 40, no_export, 3561:70 }
              community_list list of community_elm
rp-attribute: # атрибут BGP community
              community
              # задание списка community
              operator=(community_list)
              # добавить значения community в конец
              operator.=(community_list)
              append(community_elm, ...)
              # удалить значения community
              delete(community_elm, ...)
              # фильтр: true, если содержится одно из значений community
              contains(community_elm, ...)
              # сокращенная форма от community(no_export, 3561:70)
              operator()(community_elm, ...)
              # порядок независимого сравнения
              operator==(community_list)
rp-attribute: # следующий маршрутизатор на статическом маршруте
              next-hop
              # для задания маршрутизатора 7.7.7.7: next-hop = 7.7.7.7;
              # для установки адреса самомго маршрутизатора: next-hop = self;
              operator=(union ipv4_address, enum[self])
rp-attribute: # стоимость статического маршрута
              cost
              operator=(integer[0, 65535])
protocol: BGP4
          # номер AS партнерского маршрутизатора
          MANDATORY asno(as_number)
          # разрешить подавление осцилляций (flap damping)
          OPTIONAL flap_damp()
          OPTIONAL flap_damp(integer[0,65535],
                             # наказание за факт осцилляции (per flap)
                             integer[0,65535],
                             # размер наказания для подавления осцилляций
                             integer[0,65535],
                             # размер наказания для повторного использования
                             integer[0,65535],
                             # время «полужизни» в секундах для активизации (up)
                             integer[0,65535],
                             # время «полужизни» в секундах для деактивации (down)
                             integer[0,65535])
                             # максимальное наказание
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
```

Рисунок 27. Словарь RPSL.

На рисунке 27 показан исходный словарь RPSL. Он включает 7 атрибутов rp-attribute: pref для установки локального предпочтения воспринятых маршрутов, med для присваивания значения атрибуту BGP MULTI_EXIT_DISCRIMINATOR, dpa для присваивания значения атрибуту BGP DPA, aspath для добавления значений в начало атрибута BGP

AS_PATH, community для присваивания или проверки значения атрибута BGP community, next-hop для задания следующего маршрутизатора на статических маршрутах и cost для задания стоимости статических маршрутов. Словарь определяет два типа: community_elm и community_list. Тип community_elm type задаётся 4-байтовым целым числом без знака или содержит ключевое слово internet, no_export или no_advertise (определены в [9]). Целочисленное значение можно задавать с использованием 2-байтовых целых чисел, разделённых двоеточием, для формирования пространства номеров групп (community) так, чтобы провайдер мог использовать свой номер AS в качестве двух первых байтов, а два оставшихся выделять в соответствии с семантикой своих групп.

Исходный словарь (рисунок 27) определяет только две опции для протокола BGP: asno и flap_damp. Обязательная опция asno указывает номер AS партнерского маршрутизатора. Необязательная опция flap_damp рекомендует маршрутизатору подавлять осцилляции маршрутов (damp route flap) [21] при импорте маршрутов от партнёра.

Эту опцию можно указывать с параметрами или без таковых. При отсутствии параметров используется принятый по умолчанию набор значений:

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

Т. е., для каждой осцилляции устанавливается наказание 1000, маршрут «подавляется», когда суммарное значение «штрафа» достигает 2000. Значение штрафа снижается вдвое после каждых 15 минут (900 секунд) стабильности, независимо от того, подавлен маршрут или активен. Отключенный маршрут снова начинает использоваться, когда уровень штрафа опускается ниже 750. Максимальный уровень штрафа для маршрута составляет 20 000 (т. е., максимальное время подавления маршрута после установки стабильного состояния составит примерно 75 минут). Эти параметры согласуются с принятыми по умолчанию параметрами подавления осцилляций в некоторых маршрутизаторах.

Действия и фильтры политики, использующие RP-Attribute

Действия политики или фильтры, использующие rp-attribute x имеют синтаксис:

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

где method задаёт используемый метод, а "op" указывает операцию атрибута rp-attribute x. Если операция используется в спецификации композитного фильтра политики, она проверяется до выполнения операторов композитного фильтра (т. е., AND, OR, NOT и неявного оператора «ИЛИ»).

В качестве значения pref атрибута rp-attribute может использоваться положительное целое число:

```
pref = 10;
```

В качестве значения med атрибута rp-attribute может использоваться положительное целое число или слово «igp_cost»:

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

В качестве значения dpa атрибута rp-attribute может использоваться положительное целое число:

```
dpa = 100;
```

Атрибут BGP community имеет форму списка 4-байтовых целых чисел, каждое из которых представляет группу (community). В приведённом ниже примере показаны варианты добавления групп в атрибут rp-attribute.

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

В последнем случае 4-байтовое целое число представлено в форме двух 2-байтовых значений, из которых старшее равно 3561, а два младших байта задают значение 10. Следующий пример показывает, как удалить группы из атрибута rp-attribute:

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

Фильтры, использующие community атрибута rp-attribute, можно определить, как показано в следующих примерах:

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

Значение community атрибута rp-attribute можно задать в виде списка групп, как показано ниже:

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

В первом случае значение community атрибута rp-attribute содержит группы 100, NO_EXPORT, 3561:10 и 200. Во втором атрибут community в rp-attribute сброшен. Значения community атрибута rp-attribute можно сравнивать со списком групп, как показано ниже:

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

Для влияния на выбор пути BGP можно удлинять значение as_path атрибута rp-attribute путём добавления в начало номеров AS, как показано ниже:

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

Далее приведено несколько примеров некорректных записей:

```
med = -50;                    # значение -50 выходит за пределы допустимого диапазона
med = igp;                    # igp не входит в число перечисляемых значений
med.assign(10);               # метод assign не определен
community.append(AS3561:20);  # первый аргумент должен иметь значение 3561 (целое число)
```

На рисунке 28 показан расширенный пример использования rp-attribute community. В этом примере выбор маршрутов в AS3561 основывается на сравнении значений атрибута community. Остальные AS могут опосредованно влиять на выбор маршрута в AS3561 путём включения соответствующих групп в свои маршрутные анонсы.

```
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
```

Рисунок 28. Пример политики с использованием группы rp-attribute.

8 Класс Advanced route

8.1 Задание маршрутных агрегатов

Для задания агрегированных маршрутов могут использоваться атрибуты components, aggr-bndry, aggr-mtd, export-comps, inject и holes [11]. Объект route задаёт агрегированный маршрут, если указан любой из перечисленных атрибутов (за исключением inject). В качестве атрибута origin для агрегированного маршрута указывается номер AS, выполнившей агрегирование. В этом разделе термин «агрегат» или «агрегированный маршрут» используется для сгенерированных маршрутов, термин «компонента» - для маршрутов, использованных при генерации агрегата, а термин «более специфичный» относится к более специфичному маршруту, нежели агрегат, независимо от его использования при формировании атрибутов пути.

Атрибуты компонент определяют включение компоненты в агрегат и используют синтаксис, показанный ниже.

```
components: [ATOMIC] [[<filter>] [protocol <protocol> <filter> ...]]
```

где <protocol> указывает протокол маршрутизации (например, BGP4, OSPF или RIP - допустимые имена определяются в словаре), а <filter> является выражением политики. Маршруты, соответствующие одному из таких фильтров и полученные от указанного протокола, служат для формирования агрегированного маршрута. Если поле <protocol> отсутствует, по умолчанию используются все протоколы. Поле <filter> неявно содержит операцию AND с наиболее специфичным маршрутом агрегата, поэтому выбирается только одна компонента. При использовании ключевого слова ATOMIC агрегирование осуществляется на «атомарном уровне» [11]. Если поле <filter> не задано, по умолчанию используется более специфичный маршрут. Если отсутствуют атрибуты компонент, используются все более специфичные без ключевого слова ATOMIC.

```
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^+}
```

Рисунок 29. Два объекта агрегированных маршрутов.

На рисунке 29 показаны два объекта route. В первом примере агрегируется более специфичный маршрут 128.8.0.0/15 с путём, начинающимся в AS2. Во втором примере агрегируются некоторые маршруты, полученные от BGP, и некоторые маршруты от OSPF.

Атрибут aggr-bndry представляет собой выражение AS через номера AS и наборы (см. параграф 5.6). Результат определяет набор AS, которые формируют границу агрегирования. Если атрибут aggr-bndry отсутствует, границу агрегирования определяет исходная (origin) AS. За пределы границы агрегирования экспортируется только агрегированный маршрут, а более специфичные маршруты «подавляются». Однако в обмене информацией внутри границы агрегирования участвуют и более специфичные маршруты.

Атрибут aggr-mtd задаёт способ генерации агрегата и использует синтаксис

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

где <as-expression> задаётся с использованием номеров AS и наборов (см. параграф 5.6). Если <as-expression> отсутствует, по умолчанию используется значение AS-ANY. Если задано исходящее агрегирование, внутри AS будут сохраняться более специфичные маршруты, а агрегат будет формироваться перед экспортом на всех границах между данной AS и AS из <as-expression>, исключая AS, находящиеся внутри границы агрегирования (т. е., aggr-bndry применяется независимо от <as-expression>). Если задано входное агрегирование, агрегат формируется на всех границах между AS перед импортом маршрута в агрегирующую AS. Отметим, что для входного агрегирования <as-expression> не может использоваться. Если параметр aggr-mtd отсутствует, используется значение 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
```

Рисунок 30. Пример выходного агрегирования multi-AS.

На рисунке 30 показан пример исходящего агрегирования. В этом примере AS1 и AS2 координируют агрегирование и анонсируют во внешний мир только менее специфичный маршрут 128.8.0.0/15, но обмениваются между собой более специфичными маршрутами. Такая форма агрегирования полезна в тех случаях, когда часть компонент относится к AS1, а другая часть - к AS2.

При агрегировании набора маршрутов задача состоит в том, чтобы экспортировать только агрегированный маршрут, подавляя распространение более специфичных маршрутов за пределы границы агрегирования. Однако для удовлетворения некоторых требований политики и ограничений топологии (например, многодомный сайт) зачастую приходится экспортировать часть маршрутов-компонент. Атрибут export-comps эквивалентен фильтру RPSL, которому соответствуют более специфичные маршруты, что требуют экспорта за пределы границы агрегирования. Если этот атрибут отсутствует, более специфичные маршруты не экспортируются за пределы границы агрегирования. Отметим, что фильтр export-comps неявно содержит операцию AND с более специфическим маршрутом.

```
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}
```

Рисунок 31. Выходное агрегирование с исключением экспорта.

На рисунке 31 показан пример исходящего агрегирования. В этом примере более специфичный маршрут 128.8.8.0/24 экспортируется за пределы AS1 в дополнение к агрегату. Такое решение полезно для случаев, когда сайт 128.8.8.0/24 является многодомным (подключён к AS1 и неким другим AS).

Атрибут inject указывает, какой из маршрутизаторов и в какой момент выполняет агрегирование. Синтаксис атрибута показан ниже.

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

где <action> - спецификация действия (см. параграф 6.1.1), <condition> - логическое выражение, описанное ниже, а <router-expression> - выражение, описанное в параграфе 5.6.

Все маршрутизаторы из <router-expression> и в агрегирующей AS участвуют в агрегировании. Если <router-expression> отсутствует, агрегирование выполняют все маршрутизаторы внутри агрегирующей AS. Спецификация действия <action> может задавать атрибуты пути для агрегата (например, задание предпочтений).

Опишем приведённое выше логическое выражение. Агрегат генерируется тогда и только тогда, когда это выражение даёт результат true. Параметр <condition> представляет собой логическое выражение с использованием операций AND и OR (операция NOT не разрешается) для:

```
HAVE-COMPONENTS { list of prefixes }
EXCLUDE { list of prefixes }
STATIC
```

Список префиксов (list of prefixes) в HAVE-COMPONENTS (компоненты присутствуют) может быть только более специфичным по сравнению с агрегатом. Параметр имеет значение true, когда все префиксы из списка присутствуют в таблице агрегирующего маршрутизатора. Список может включать также диапазоны префиксов (т. е., могут использоваться операторы ^-, ^+, ^n, ^n-m). В этом случае по крайней мере один префикс из каждого диапазона в списке должен присутствовать в таблице агрегирующего маршрутизатора, чтобы выражение имело значение true. Список префиксов в параметре EXCLUDE (исключить) может быть произвольным. Параметр имеет значение true, если ни одного из перечисленных в списке префиксов нет в таблице агрегирующего маршрутизатора. Этот список также может включать диапазоны префиксов и ни одного из префиксов указанных в списке диапазонов не должно быть в таблице маршрутизации. Ключевое слово STATIC всегда даёт значение true. Если условия агрегирования (см. выше) не заданы, агрегат генерируется тогда и только тогда, когда есть компонента в таблице маршрутизации (т. е., более специфичный маршрут, нежели соответствующий фильтру в атрибуте 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
```

Рисунок 32. Пример вставки.

На рисунке 32 показаны два примера. В первом примере агрегат помещается в анонсы на 2 маршрутизаторах, каждый из которых устанавливает своё значение атрибута пути dpa. Во втором случае агрегат генерируется только в тех случаях, когда в таблице маршрутизации имеются префиксы 128.8.0.0/16 и 128.9.0.0/16, в отличие от первого случае, где для генерации агрегата достаточно присутствия в таблице одного из этих префиксов.

Атрибут holes содержит список адресных префиксов компонент, которые не достижимы через агрегированный маршрут (возможно по причине того, что часть адресов префикса агрегата просто не распределена). Атрибут holes полезен при диагностике. Во втором примере на рисунке 32 имеется «дыра» 128.8.8.0/24. Наличие её может быть результатом смены провайдера и утраты части адресного пространства.

8.1.1 Взаимодействие с правилами в классе aut-num

Сформированный агрегат анонсируется в другие AS только в тех случаях, когда политика экспорта AS разрешает экспортировать его. Когда агрегат сформирован, более специфичные маршруты подавляются при экспорте (за

исключением AS из aggr-bndry и компонент, исключённых в export-comps). При наличии таких исключений в политику экспорта AS следует явно включать экспорт исключений.

Если агрегат не сформирован, более специфичные маршруты могут экспортироваться в другие AS, если правила экспорта AS разрешают это. Иными словами, перед экспортом маршрут фильтруется дважды (агрегат и более специфичный маршрут) - по объектам route и правилам экспорта в 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
```

Рисунок 33. Взаимодействие с правилами в классе aut-num.

На рисунке 33 показан пример взаимодействия. Проверка объектов route покажет, что более специфичные маршруты 128.8.0.0/16 и 128.9.0.0/16 следует передавать между sAS1, AS2 b AS3 (т. е., внутри границы агрегирования). Исходящее агрегирование выполняется для AS4 и AS5, но не для AS3, поскольку та находится внутри границы агрегирования AS3. Объект aut-num позволяет экспортировать обе компоненты в AS2 и только компоненту 128.8.0.0/16 в AS3. Агрегат может быть сформирован только при доступности обеих компонент. В этом случае агрегат анонсируется только в AS4 и AS5. Однако при недоступности одной из компонент агрегат формироваться не будет и все доступные компоненты и более специфичные маршруты будут экспортироваться в AS4 и AS5. Независимо от выполнения агрегирования в AS6 будут экспортироваться только более специфичные маршруты (эта AS не указана в списке атрибута aggr-mtd).

При входящем агрегировании конфигурация может блокировать настройки агрегирования в маршрутизаторах, на которых политика импорта AS запрещает импорт более специфичных маршрутов.

8.1.2 Устранение неоднозначностей при перекрывающихся агрегатах

Когда задано несколько агрегированных маршрутов и они перекрываются (т. е., один является более специфичным, чем другой), эти маршруты должны оцениваться в порядке от более специфичного к менее специфичному. При выполнении исходящего агрегирования для партнёра агрегат и компоненты, перечисленные в атрибуте export-comps для этого партнёра, доступны для генерации следующего менее специфичного агрегата. Компоненты, не заданные в атрибуте export-comps, недоступны для агрегирования. Маршрут может быть экспортирован в AS, если менее специфичный агрегат может быть экспортирован в данную AS или он указан в списке атрибута export-comps экспортируемого маршрута. Отметим, что это определение является рекурсивным.

```
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}
```

Рисунок 34. Агрегирование с перекрытием.

На рисунке 34 AS1 совместно с AS2 агрегирует префиксы 128.8.0.0/16 и 128.9.0.0/16 в 128.8.0.0/15. AS3 и AS1 вместе агрегирует 128.10.0.0/16 и 128.11.0.0/16 в 128.10.0.0/15. Все эти AS совместно могут агрегировать 4 маршрута в 128.8.0.0/14. Предполагая доступность всех четырёх компонент, маршрутизатор в AS1 для внешних AS (например, AS4) будет сначала генерировать 128.8.0.0/15 и 128.10.0.0/15. Это сделает маршруты 128.8.0.0/15, 128.10.0.0/15 и исключение 128.11.0.0/16 доступными для генерации 128.8.0.0/14. Маршрутизатор тогда будет генерировать 128.8.0.0/14 из этих трёх маршрутов. Следовательно, для AS4 маршрут 128.8.0.0/14 и его исключение 128.10.0.0/15 вместе с его исключением 128.11.0.0/16 будут экспортируемыми.

Для AS2 маршрутизатор в AS1 будет генерировать только 128.10.0.0/15. Следовательно, 128.10.0.0/15 и его исключение 128.11.0.0/16 будут экспортируемыми. Отметим, что 128.8.0.0/16 и 128.9.0.0/16 также являются экспортируемыми, поскольку они не участвуют в агрегировании экспорта для AS2.

Аналогично, для AS3 маршрутизатор в AS1 будет генерировать только 128.8.0.0/15. В этом случае экспортируемыми маршрутами будут 128.8.0.0/15, 128.10.0.0/16 и 128.11.0.0/16.

8.2 Задание статических маршрутов

Атрибут inject можно использовать для задания статических маршрутов с помощью условия upon static:

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

В этом случае маршрутизаторы из <router-expression> выполняют действие <action> и вставляют маршрут в систему маршрутизации interAS статически. Действие <action> может устанавливать некоторые атрибуты маршрута (например, адрес следующего маршрутизатора или стоимость).

В приведённом ниже примере маршрутизатор 7.7.7.1 вставляет маршрут 128.7.0.0/16. Маршрутизаторами следующего интервала (в этом примере два таких маршрутизатора) для этого маршрута являются 7.7.7.2 и 7.7.7.3, стоимость маршрута составляет 10 через маршрутизатор 7.7.7.2 и 20 - через 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 Класс inet-rtr

Атрибут	Значение	Тип
inet-rtr	<dns-name>	Обязательный, одно значение, ключ класса
alias	<dns-name>	Необязательный, много значений
local-as	<as-number>	Обязательный, одно значение
ifaddr	см. ниже	Обязательный, много значений
peer	см. ниже	Необязательный, много значений
member-of	список <rtr-set-name>	Необязательный, много значений

Рисунок 35. Атрибуты класса inet-rtr.

Маршрутизаторы указываются с помощью класса inet-rtr, атрибуты которого показаны на рисунке 35. Атрибут inet-rtr содержит доменное имя (DNS) описываемого маршрутизатора. Каждый атрибут alias, при его наличии, представляет собой каноническое DNS-имя маршрутизатора. Атрибут local-as указывает номер AS, которая владеет/управляет этим маршрутизатором.

Атрибут ifaddr использует синтаксис:

```
<ipv4-address> masklen <integer> [action <action>]
```

Адрес IP и размер маски обязательно задавать для каждого интерфейса. Дополнительно могут быть указаны значения action для установки других параметров интерфейса.

```
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()
```

Рисунок 36. Объекты inet-rtr.

На рисунке 36 приведён пример объекта inet-rtr. Имя маршрутизатора amsterdam.ripe.net, а каноническим именем для него является amsterdam1.ripe.net. Маршрутизатор подключён к 4 сетям. IP-адреса интерфейсов в эти сети и размеры масок указаны в атрибутах ifaddr.

Каждый атрибут peer (при его наличии) задаёт протокольное партнерство с другим маршрутизатором. Атрибут peer использует синтаксис:

```
  <protocol> <ipv4-address>      <options>
| <protocol> <inet-rtr-name>     <options>
| <protocol> <rtr-set-name>      <options>
| <protocol> <peering-set-name>  <options>
```

где <protocol> - имя протокола, <ipv4-address> - адрес IP партнерского маршрутизатора, а <options> содержит список разделённых запятыми опций партнёрства для протокола <protocol>. Взамен IP-адресов партнеров могут использоваться их имена inet-rtr-name. Возможные имена и атрибуты протоколов определены в словаре (см. раздел 7). В приведённом выше примере маршрутизатор имеет партнерские отношения BGP с маршрутизатором 192.87.45.195 в AS3334 и поддерживает подавление осцилляций при импорте маршрутов от этого партнёра.

```
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()
```

Рисунок 37. Объект inet-rtr с группой партнёров.

Вместо одного партнёра можно задать группу, используя формы <rtr-set-name> и <peering-set-name>. При использовании формы <peering-set-name> включаются только партнёры маршрутизатора из указанного набора. На рисунке 37 показан пример объекта inet-rtr с группой партнёров.

10 Расширение RPSL

Опыт использования предыдущих версий языка и форматов данных описания политики маршрутизации (PRDB [2], RIPE-81 [8], RIPE-181 [7]) показывает потребность в расширении RPSL. С учётом этого расширяемость была одной из основных целей при разработке RPSL. Новые протоколы маршрутизации или новые возможности существующих протоколов могут быть без значительных усилий адаптированы с использованием класса dictionary в языке RPSL. Можно также добавлять новые классы или новые атрибуты к имеющимся классам.

В этом разделе приводятся рекомендации по расширению RPSL. Эти рекомендации разработаны с учётом совместимости с существующими средствами и базами данных. Далее рассматриваются возможные варианты расширения RPSL в порядке снижения предпочтительности этих вариантов.

10.1 Расширение путём замены класса dictionary

Класс dictionary является основным механизмом возможного расширения RPSL. Объекты dictionary определяют атрибуты и типы политики маршрутизации, а также протоколы маршрутизации.

Рекомендуется обновлять словарь RPSL с целью включения соответствующих определений rp-attribute и протоколов по мере введения новых атрибутов пути и появления новых возможностей у маршрутизаторов. Например, в предшествующей спецификации RPSL было можно указать, что маршрутизатор выполняет подавление осцилляций для партнёра, но не было возможности управлять параметрами этого подавления. Позднее эти параметры были добавлены путём изменения словаря.

При изменении словаря должна обеспечиваться полная совместимость. Например, для случая подавления осцилляций параметры подавления были заданы опциональными, поскольку эти параметры могут быть нежелательны для некоторых ISP. Таким образом была обеспечена совместимость. Любой объект, регистрируемый без параметров, сохраняет свою корректность. Предполагается, что все инструменты, работающие на базе RPSL, будут выполнять принятые по умолчанию действия для атрибутов политики маршрутизации, которые им непонятны (например, выдавать предупреждение оператору или просто игнорировать непонятное). Следовательно, старые инструменты, встречающиеся с параметризованным подавлением осцилляций, будут игнорировать неизвестные параметры подавления.

10.2 Расширение за счёт добавления атрибутов существующих классов

Новые атрибуты можно добавить к любому классу. Для обеспечения полной совместимости новые атрибуты не должны противоречить семантике объектов, к которым атрибуты добавляются. Все инструменты, использующие IRR, следует разрабатывать так, чтобы они игнорировали непонятные атрибуты. Большинство современных средств способны поступать таким образом.

Рекомендуется добавлять атрибуты к существующим классам при обнаружении новых аспектов данного класса. Например, класс route в RPSL является расширением своего предшественника из RIPE-181, обеспеченным за счёт включения новых атрибутов, позволяющих задавать агрегированные и статические маршруты.

10.3 Расширение путём добавления классов

В RPSL можно добавлять новые классы для хранения новых типов данных политики. Обеспечение совместимости не составляет проблем, поскольку существующие классы остаются понятными. Поскольку программы делают запросы IRR только для известных им классов, проблем в этом случае возникать не должно.

Перед добавлением класса следует получить ответ на вопрос - даст ли информация, содержащаяся в объектах этого класса, что-то новое и полезное по сравнению с существующими классами. Например, если в IRR нужно сохранить географическое положение маршрутизатора, можно добавить новый класс (скажем, router-location). Однако лучше будет сохранить эту информацию в классе inet-rtr, добавив к нему атрибут location.

10.4 Расширение путём изменения синтаксиса существующих атрибутов RPSL

Если описанные выше методы не позволяют реализовать желаемое расширение, для него может потребоваться изменение синтаксиса RPSL. Все изменения синтаксиса RPSL должны обеспечивать совместимость с предшествующими версиями и эти изменения следует использовать только в крайних случаях, поскольку полной совместимости обычно обеспечить не удаётся. Старый синтаксис после расширения должен сохранять корректность.

11 Вопросы безопасности

В этом документе описан язык RPSL, предназначенный для выражения политики маршрутизации. Язык определяет держателя объекта (класс mntner) который контролирует или «поддерживает» объекты, хранящиеся в базе данных RPSL. Запросы держателей могут аутентифицироваться различными способами, определёнными атрибутом auth данного объекта.

Конкретные протоколы, используемые IRR для связи с объектами RPSL, выходят за пределы этого документа, но предполагается использование разных методов, от интерактивных протоколов «запрос-отклик» для сохранения объектов до протоколов пересылки (типа электронной почты и даже обычного телефона). Независимо от используемых в конкретной ситуации протоколов, предполагается наличие подходящих средств и методов защиты (таких, как IPSEC, TLS или PGP/MIME.

12 Благодарности

Авторы благодарят 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, а также членов рабочей группы IETF RPS за их комментарии и предложения.

Литература

[1] Реестр маршрутизации Internet. Процедуры. http://www.ra.net/RADB.tools.docs/, http://www.ripe.net/db/doc.html.

[2] Nsfnet policy routing database (prdb). Maintained by MERIT Network Inc., Ann Arbor, Michigan. Contents available from nic.merit.edu.:/nsfnet/announced.networks/nets.tag.now by anonymous ftp[1].

[3] Alaettinouglu, C., Bates, T., Gerich, E., Karrenberg, D., Meyer, D., Terpstra, M. and C. Villamizer, "Routing Policy Specification Language (RPSL)", RFC 2280, January 1998.

[4] C. Alaettinouglu, D. Meyer, and J. Schmitz. Application of routing policy specification language (rpsl) on the internet. Work in Progress[2].

[5] T. Bates. Specifying an `internet router' in the routing registry. Technical Report RIPE-122[3], RIPE, RIPE NCC, Amsterdam, Netherlands, October 1994.

[6] 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[4], RIPE, RIPE NCC, Amsterdam, Netherlands, October 1994.

[7] Bates, T., Gerich, E., Joncheray, L., Jouanigot, J-M., Karrenberg, D., Terpstra, M. and J. Yu, "Representation of IP Routing Policies in a Routing Registry", RFC 1786, March 1995.

[8] 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[5], RIPE, RIPE NCC, Amsterdam, Netherlands, February 1993.

[9] Chandra, R., Traina, P. and T. Li, "BGP Communities Attribute", RFC 1997, August 1996.

[10] Crocker, D., "Standard for ARPA Internet Text Messages", STD 11, RFC 822, August 1982.

[11] Fuller, V., Li, T., Yu, J. and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519, September 1993.

[12] D. Karrenberg and T. Bates. Description of inter-as networks in the ripe routing registry. Technical Report RIPE-104, RIPE, RIPE NCC, Amsterdam, Netherlands, December 1993.

[13] D. Karrenberg and M. Terpstra. Authorisation and notification of changes in the ripe database. Technical Report ripe-120, RIPE, RIPE NCC, Amsterdam, Netherlands, October 1994.

[14] B. W. Kernighan and D. M. Ritchie. The C Programming Language. Prentice-Hall, 1978.

[15] A. Lord and M. Terpstra. Ripe database template for networks and persons. Technical Report ripe-119, RIPE, RIPE NCC, Amsterdam, Netherlands, October 1994.

[16] A. M. R. Magee. Ripe ncc database documentation. Technical Report RIPE-157, RIPE, RIPE NCC, Amsterdam, Netherlands, May 1997.

[17] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[18] Y. Rekhter. Inter-domain routing protocol (idrp). Journal of Internetworking Research and Experience, 4:61--80, 1993.

[19] Rekhter Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, March 1995.

[20] C. Villamizar, C. Alaettinouglu, D. Meyer, S. Murphy, and C. Orange, "Routing policy system security", Work in Progress[6].

[21] Villamizar, C., Chandra, R. and R. Govindan, "BGP Route Flap Damping", RFC 2439, November 1998.

[22] J. Zsako, "PGP authentication for ripe database updates", Work in Progress[7].

A Сайты с реестрами маршрутизации

В ноябре 1996 набор реестров маршрутизации включал RIPE, RADB, CANet, MCI и ANS. Для получения информации о действующих реестрах вы можете обратиться в один из указанных в списке.

B Правила грамматики

В этом разделе приведены формальные грамматические правила RPSL. Основные типы данных определены в разделе 2. Для атрибутов, значения которых относятся к базовым типам или содержат списки значений базовых типов, формальные правила грамматики здесь не определены. Приведенные правила описаны на входном языке GNU Bison. Следовательно, правила можно копировать и помещать в программы GNU Bison.

```
//**** Базовые атрибуты ************************************************

changed_attribute: ATTR_CHANGED TKN_EMAIL TKN_INT

//**** класс aut-num ***************************************************

//// as_expression ////////////////////////////////////////////////////
```

[1] В настоящее время недоступна. *Прим. перев.*
[2] Работа была опубликована в RFC 2650. *Прим. перев.*
[3] Доступен по ссылке ftp://ftp.ripe.net/ripe/docs/ripe-122.txt. *Прим. перев.*
[4] Доступен по ссылке http://www.ripe.net/ripe/docs/ripe-181 в формате ASCII и PDF. *Прим. перев.*
[5] Доступен по ссылке ftp://ftp.ripe.net/ripe/docs/ripe-081.txt. *Прим. перев.*
[6] Работа была опубликована в RFC 2725. *Прим. перев.*
[7] Работа была опубликована в RFC 2726. *Прим. перев.*

```
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

//// партнерство ///////////////////////////////////////////////////////

peering: as_expression opt_router_expression opt_router_expression_with_at
| TKN_PRNGNAME

//// действие //////////////////////////////////////////////////////////

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 ']' ';'
| ';'

//// фильтр ////////////////////////////////////////////////////////////

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_PRFXV4
| TKN_PRFXV4RNG

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 ']'

//// пара «партнерство - действие» ////////////////////////////////////

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

//// фактор 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 ';'

//// элемент import/export ////////////////////////////////////////////

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

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

//// выражение 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

//// протокол /////////////////////////////////////////////////////////

opt_protocol_from:
| KEYW_PROTOCOL tkn_word
```

```
opt_protocol_into:
| KEYW_INTO tkn_word

//**** атрибуты 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

//**** класс inet-rtr ************************************************

ifaddr_attribute: ATTR_IFADDR TKN_IPV4 KEYW_MASKLEN TKN_INT opt_action

//// атрибут 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

//**** класс 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

//// атрибут 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

//// атрибут 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_PRFXV4
| TKN_PRFXV4RNG

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 ')'

//// атрибут 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

//// атрибут protocol ///////////////////////////////////////////////

protocol_attribute: ATTR_PROTOCOL tkn_word protocol_options

protocol_options:
| protocol_options protocol_option

protocol_option: KEYW_MANDATORY method
| KEYW_OPTIONAL method

//**** определения маркеров ***********************************************
//// макросы flex, используемые в определениях //////////////////////////
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:]_-]+
//// определения маркеров ////////////////////////////////////////////
TKN_INT         {SINT}
TKN_INT         {INT}:{INT}             если каждое значение {INT} содержит 2 октета
TKN_INT         {INT}.{INT}.{INT}.{INT} если каждое значение {INT} содержит 1 октет
TKN_REAL        {REAL}
TKN_STRING      Как в языке программирования 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} если определено в словаре
TKN_WORD        {NAME}
TKN_DNS         {DNAME}("."{DNAME})+
TKN_EMAIL       {ENAME}@({DNAME}("."{DNAME})+|{IPV4})
```

C Отличия от RFC 2280

RFC 2280 [3] содержит предыдущую версию языка RPSL. В этом приложении перечислены основные отличия современной версии от прежней.

- В новой версии можно задавать целые числа в форме 1-октетных целочисленных значений, разделенных точками (например, 1.1.1.1) или двух 2-октетных целочисленных значений, разделенных двоеточием (например, 3561:70). См. раздел 2.
- Определение диапазона адресных префиксов расширено так, что префикс можно рассматривать, как вырожденный диапазон (см. раздел 2).
- Определена семантика оператора диапазона, применимая к множеству, содержащему диапазон адресных префиксов (например, {30.0.0.0/8^24-28}^27-30). См. раздел 2.
- Все даты указываются в формате UTC (см. раздел 2).
- Добавлен знак сложения (+) к символам пробела и табуляции для обеспечения возможности записи значения атрибута в несколько строк (например, каждая новая строка может начинаться с пробела, символа табуляции или знака +). См. раздел 2.
- Из языка удален атрибут withdrawn (отзыв) класса route.
- Добавлен класс filter-set (см. параграф 5.4).
- Добавлен класс rtr-set (см. параграф 5.5).
- Добавлен класс peering-set (см. параграф 5.6).
- Фильтры можно указывать по именам filter-set (см. параграф 5.4).
- Партнерство можно задавать по именам peering-set и rtr-set. Как локальный, так и удаленный маршрутизаторы-партнеры можно задать с использованием выражений router (см. параграф 5.6).
- Атрибут peer класса inet-rtr можно указывать по именам peering-set и rtr-set (см. раздел 9).
- Изменены синтаксис и сементика типов union и list, а также атрибута typedef (см. раздел 7).
- В исходном словаре изменен атрибут typedef, определяющий community_elm и rp-attribute в определении атрибута community (см. раздел 7).
- Добавлены рекомендации по расширению RPSL (см. раздел 10).
- Добавлены формальные парвила грамматики (см. Приложение B).

D Адреса авторов

Cengiz Alaettinoglu

USC/Information Sciences Institute

EMail: cengiz@isi.edu

Curtis Villamizar

Avici Systems

EMail: curtis@avici.com

Elise Gerich

At Home Network

EMail: epg@home.net

David Kessens

Qwest Communications

EMail: David.Kessens@qwest.net

David Meyer

University of Oregon

EMail: meyer@antc.uoregon.edu

Tony Bates

Cisco Systems, Inc.

EMail: tbates@cisco.com

Daniel Karrenberg

RIPE NCC

EMail: dfk@ripe.net

Marten Terpstra

c/o Bay Networks, Inc.

EMail: marten@BayNetworks.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Подтверждение

Финансирование функций RFC Editor обеспечено Internet Society.