Transition to RIPE DB v3.0

João Luis Silva Damas

Database Group
RIPE NCC

Document: NewDB–4
Date: December 27, 2000

ABSTRACT

This document describes the issues involved in the transition from RIPE DB v2.x to RIPE DB v3.0. We discuss issues resulting from fundamental differences in software functionality and a migration plan aimed at minimizing service problems and user inconvenience.

Contents

1 Introduction

For the past few months the RIPE NCC has embarked in a project to rewrite the software supporting the RIPE Database.

In this document we look at the new features introduced in the new version of the software, analyzing how it impacts users. Finally, we present a transition plan for moving operations from the current system to the new one.

For a description of the current database refer to the documentation for RIPE Database versions before 3.0 [1, 2, 3].

2 New features

2.1 RPSL

One of the major differences in the new RIPE DB software is the language used to express routing registry objects: RPSL.

RPSL (Routing Policy Specification Language[4]) is a language to enable the representation of internet routing policies in databases such as the RIPE Database.

Currently, the RIPE database system uses the language known as RIPE 181 [2] as its routing policy specification language. RPSL is a superset of RIPE 181 with somewhat different syntax and much more functionality.

As a consequence of this change of language, the objects used to express routing policy have been modified and some new objects have been defined.

For a list of templates of the new and modified RPSL objects, see Appendix A.

See section 4 for a description of the proposed transition to the new RIPE Database with RPSL support.

2.1.1 Reserved strings in RPSL

The RPSL language defines a few reserved strings which can only appear in the names of specific objects.

These are the prefixes AS-, RS- , RTRS-, FLTR- and PRNG- (case insensitive) which can only appear in the names of as set, route set, router set, filter set and peering set objects respectively. Furthermore each of these object types must have names starting with these prefixes.

2.1.2 Transition issues derived from reserved strings

RIPE Database versions previous to v3.0 allowed the reserved strings mentioned above to appear in any object name. This has resulted in the existence of objects that have

names starting with these prefixes but are not of the corresponding object type. For example, some maintainer names that were created for the purpose of protecting autnum objects have names that look like AS-XXX-MNT.

These names are not allowed by RPSL. Therefore, these objects will need to be renamed for conformance with the new standard.

2.1.3 Differences with the current RPSL standard

No admin-c, tech-c in route objects

The RPSL proposed standard calls for the addition of admin-c and tech-c attributes to the route object, contrary to the current RIPE specification. The RIPE DB will not support admin-c and tech-c in route objects.

Future revisions of RPSL will withdraw these two attributes from the route object.

Additionally, the admin-c attribute is mandatory in all other attributes, not only in the autnum object.

2.2 Routing Policy System Security (rps-auth)

Apart from the changes introduced by RPSL itself, further changes have been introduced to support the proposal specified in the Routing Policy System Security [5].

The RIPE Database implements the core functionality of this draft. For a complete description, refer to the draft.

2.2.1 New attributes

mnt-routes

This new attribute may appear in the autnum, inetnum and route objects. Its arguments are a maintainer name, optionally followed by a list of prefixes enclosed in curly braces or the word “ANY”. The default is ANY or all more specific prefixes. For a complete description see [5].

In an autnum object this attribute controls which maintainers can create route objects containing the AS number as origin.

In inetnum and route objects it controls which maintainers can create route objects with exact or more specific prefixes.

referral-by

The referral-by attribute is mandatory in maintainer objects. It stores the name of the maintainer that created the maintainer object, meaning that it points to the *parent* object. Currently all maintainers in the RIPE Database are created by the RIPE NCC upon user request.

The RIPE NCC will therefore add an extra attribute to all maintainers automatically when migrating data to the new Database:

referral-by: RIPE-DBM-MNT

auth-override

This is a way of dealing with maintainers that have become unresponsive. This attribute can only be set by the maintainer referred to in the referral-by.

Note: To enable software interoperability with other Internet Routing Registry databases, this functionality will be present in RIPE DB v3.0. It will, however, not be used by the RIPE NCC (the maintainer that has, so far, created all maintainers in the RIPE DB) until a debate takes place in the RIPE working groups involved in Database developments.

2.2.2 Modified attributes

mnt-by

This attribute is mandatory in all objects except domains, persons and roles.

In the current RIPE database a lot of objects that need a maintainer according to this definition, do not currently have one.

To solve this problem in a way that minimizes user inconvenience and exceptions in the software, the RIPE NCC will create a new maintainer object in the database (see figure 1).

The attribute “mnt-by: RIPE-NCC-NONE-MNT” (see figure 1) will be added to all objects (except persons, roles and domains which are not involved in any way with RPSL) which at the time of data migration do not have any *mnt-by* attributes.

This maintainer object will never be used for any sort of data protection or update control and is created solely for the purpose of making data conformant and raising user awareness about the convenience of acquiring a proper maintainer object.

mnt-lower

The mnt-lower attribute may now appear in more object types. Previously it was only accepted in the inetnum, inet6num and domain objects.

Beginning with version 3.0 of the software it will also be accepted in route, as-block and aut-num objects.

When it appears in a route object, it controls authorization for creation of new route objects with prefixes more specific (longer) than the one in the object where this attribute appears.

When appearing in a as-block object it controls the creation of autnum or as-block objects contained in the range specified in the as-block object where the attribute appears.

When appearing in an autnum object it controls the creation of objects with hierarchical names containing this autnum. For an explanation of hierarchical names, refer to [4].

```
mntner:      RIPE-NCC-NONE-MNT
description: This maintainer will never have auth other than NONE
remarks:     This maintainer was created to deal with the fact that
remarks:     mnt-by fields are mandatory in the RIPE Database.
remarks:     All objects (except person, role and domain) which at
remarks:     the time of transition to RIPE DB v3.0 did not have
remarks:     any mnt-by attribute will have "mnt-by: RIPE-NCC-NONE-MNT"
remarks:     added to them.
remarks:     The RIPE NCC will never use this maintainer object to
remarks:     enforce any sort of control over user's objects.
remarks:     It is intended solely to raise user awareness about the convenience
remarks:     of obtaining a real maintainer object for themselves
remarks:     while dealing with the new standards. More information is
remarks:     available at http://www.ripe.net/db
remarks:     The RIPE NCC strongly advises users to obtain their own
remarks:     maintainer and substitute references to this maintainer.
remarks:     *** NOTE: Remember that an update to an object protected by several
remarks:     maintainer objects need only satisfy the authentication
remarks:     of one of the maintainers in order to succeed. Therefore,
remarks:     object protection is only as strong as the weakest
remarks:     maintainer in the object.
admin-c:     JLSD1-RIPE
auth:        NONE
upd-to:      ripe-dbm@ripe.net
mnt-by:      RIPE-DBM-MNT
changed:     ripe-dbm@ripe.net 19990909
source:      RIPE
```

Figure 1: The maintainer object created for rps-auth[5] compliance.

2.3 MIME encapsulation of e-mail updates

Starting with RIPE Database v3.0, users will be able to send updates in MIME codified e-mails.

This new feature is meant mainly to make the cryptographic signing of the message easier when using mail agents that place the signature in a separate MIME part than the rest of the message body.

It also allows the definition of scopes of authentication within the message (eg. parts where different passwords apply) and nested signing of messages which may be necessary under some conditions when updating objects whose authorization must be derived from more than one party.

This last condition is a consequence of the implementation of the Routing Policy System Security standard [5]. One example of where nested signing may be required is when registering a new route object, where authorization must be granted by the party which has been allocated the IP address space and the party whose autonomous system will be originating that route.

2.4 New objects

This section gives an overview of the objects that have no equivalent in versions of the RIPE Database prior to v3.0. It also notes objects that have changed name but have the same purpose as objects in previous versions of the database (though possibly with extended functionality). A full definition of these objects, their syntax and their purpose can be found in the RPSL definition [4].

route-set

In previous versions of the database some of this object's functionality was provided by the **community** object.

as-set

This object includes the functionality of the **as-macro** object present before v3.0

peering-set, filter-set, rtr-set

These objects have no equivalence in RIPE Database versions before 3.0.

as-block object

as-block is a new object introduced by the rps-auth draft in order to control delegation of blocks of autonomous system numbers.

Templates for all the above objects are available in Appendix A.

2.5 Objects with modified syntax

The **aut-num**, **route** and **inet-rtr** have had their functionality significantly expanded, according to the RPSL [4] definition.

2.6 Other changes

Following is a description of changes derived from RPSL but that affect all objects in the database.

For a full description of RPSL, refer to [4].

2.6.1 Line continuation

An attribute's value can be split over multiple lines, by having a space, a tab or a plus('+') character as the first character of the continuation lines. The character '+' allows attribute values to contain blank lines. For an example, see figure 2.

Note: From the above, it follows that **no** white space is allowed before an attribute's name in an object.

```
key-cert:    PGPKEY-29FED809
method:      PGP
owner:       ripe <ripet@ripe>
fingerpr:    1241 433F ABCD EF01 2345  0666 3112 1999 0101 2000
certif:      -----BEGIN PGP PUBLIC KEY BLOCK-----
             Version: 5.0i
+
             n6vQClCbAkbTCD1mpF1Bn5x8vYlLIhkmuquiXsNV6TILOwACAgf8Di3P62C9vXWM
             nUDg+R9BUCuj5RlKIgG71Ix0wBoAtLNjZOd3iEH1/D4mk1J0RfE+p69bqxaZijSr
             7F+5m8BPoXmcSPDFfUOGotmGBv2hmG+IfxASPAeG6RbaWkS0dVGKZJsEbyC/NTOm
             tTMsbd2NBBzx3Is9iE/85u/xEv9xEavDEmZbj6LHwG9fiyPzp2gJz1xPZfcoRfRg
             Eikzsmjs6ZkIA9eM11zwnzcj2h25S2wOuslKYo4QJzxLctD6oesmqMUpFFmIDnT2
             Drf1tnE3bjK400dGBGlQhOvYJPjnjRgEdtUH/77/8bSgmJUGp7QiUXV6YSBTdXBw
             b3J0IDxzdXBwb3J0QHVrLnF1emEuY29tPokASwQQEQIACwUCNtRAtQQLAwIBAAoJ
             EPM5y/N5MI41wBAAniuKAoWqx/fbGoT8yk3xK2jjKX/8AKC97GdMQVSVRx/Anllg
             MQznV0T3WLkCDQQ21EC1EAgA9kJXtwh/CBdyorrWqULzBej5UxE5T7bxbrlLOCDa
             AadWoxTpj0BV89AHxstDqZSt90xkhkn4DIO9ZekX1KHTUPj1WV/cdlJPPT2N286Z
             4VeSWc39uK50T8X8dryDxUcwYc58yWb/Ffm7/ZFexwGq01uejaClcjrUGvC/RgBY
             K+X0iP1YTknbzSC0neSRBzZrM2w4DUUdD3yIsxx8Wy2O9vPJI8BD8KVbGI2Ou1WM
             uF040zT9fBdXQ6MdGGzeMyEstSr/POGxKUAYEY18hKcKctaGxAMZyAcpesqVDNmW
             r27W1SdQJPFHZQCfYqJ+Hs674MaoFrnzgDWFvlLm7Gg=
             -----END PGP PUBLIC KEY BLOCK-----
notify:      ripe@ripe.net
mnt-by:      A-MNT
changed:     ripe@ripe.net 19990909
source:      RIPE
```

Figure 2: Example RPSL object with an attribute spanning multiple lines

2.6.2 Relevance of attribute order

When displaying the object on a query the attributes must appear in the same order as the user sent them to the database. This is very useful for example when using multiple *remarks* lines to comment on the values of the following or preceding attributes. For an example, see figure 3.

```
route: 193.0.0.0/23
remarks: This route is announced out of AS3333
origin: AS3333
remarks: It is maintained by the RIPE NCC
mnt_by: RIPE-NCC-MNT
...
```

Figure 3: Example RPSL object with multiple occurrences of an attribute

2.6.3 End of line comments

RPSL offers the possibility of including comments on any attribute value. This is done by typing the # character followed by the desired comment. Any text found after the

```
person:      Joao Luis Silva Damas                  # This is the name
address:     RIPE Network Coordination Centre (NCC) # The address
address:     Singel 258
address:     1016 AB Amsterdam
address:     The Netherlands
phone:       +31 20 535 4444
fax-no:      +31 20 535 4445
e-mail:      joao@ripe.net
nic-hdl:     JLSD1-RIPE                             # Acid free
notify:      joao@ripe.net
changed:     roderik@ripe.net 19970926
source:      RIPE
```

Figure 4: Example of an object using end of line comments

character is considered a comment and not parsed for syntax. For an example, see figure 4.

2.6.4 Empty attributes

In the past, some updates have been sent to the database in which some object contained attributes with no values, that is, only the attribute name was present in the object.

This **is a syntax error** but has been handled silently by the database by deleting the problematic parts and proceeding with the update.

With the increasing need for security and in order to be able to preserve data as it was sent by the user, the RIPE DB v3.0 software will treat these syntax errors as real errors, resulting in failed updates if it encounters such data.

2.6.5 Undocumented attribute aliases

Currently the RIPE Database software silently corrects some mis-spellings in attribute names, converting them to proper ones and proceeding with the update.

For the same reasons as above this can no longer be the case. The database will not alter any values sent by the user.

As above the objects must really be syntactically correct in order for the update to succeed.

2.6.6 Attributes to which the database can add data

There are a few attributes to which the RIPE Database software can add data not initially included by the user.

These are the attributes of type *generated*, which are added to the object by the database. These appear within the *key-cert* object (*method*, *owner* and *fingerprint* attributes) and the inet6num object (*status*).

This is also the case for the **changed** attribute which appears on all objects. When the user does not provide a *date*, but only an e-mail address, the database will add the date corresponding to the time where the object was processed.

Users verifying the authenticity of these objects must keep in mind that, in these cases, the object sent in the update message, where the PGP signature (if any) appears, may not be identical to the one stored in the database. This kind of verification is only possible if the user has access to the original update message.

2.7 New query types

2.7.1 Queries for tool support. RAToolset

- -x flag

  This flag applies to lookups for inet(6)num and route objects. It specifies that only objects with an exact match should be returned. By default (without this flag) the database will return the smallest less specific range if it does not find an exact match for an IP address range or prefix.

- -K flag

  Requests that only the object's primary keys are to be returned. The exception are the set objects, where the *members* attributes will also be returned. By default the server will return full objects. This flag does **not** apply to person and role objects.

- -l flag

  This flag requests that the server **NOT** return the exact match but only the **smallest** of the IP ranges that are **bigger** than the user supplied range and that **fully contains** it. This is usually referred to as the **one level less specific** range.

- -q flag

  This flag is a way of retrieving information about the server. Currently it supports two arguments:

  - version - Returns the server version.
  - sources - Returns the list of sources (databases) currently used for queries.

  This query is most useful when using persistent connections (request by issuing the -k flag).

2.7.2 Membership of set objects in RPSL as implemented in RIPE DB v3.0

RPSL introduces a much more complex and flexible set-member relationship than was previously available in RIPE 181.

In RPSL[4] there are two ways in which an object can be a member of a set object.

```
route-set:   RS-FOO          route:     193.0.0.0/22
mbrs-by-ref: MNT-FOOBAR      origin:    AS3333
...                          member-of: RS-FOO
                             mnt-by:    MNT-FOOBAR
                             ...

route: 192.168.0.0/24
origin:    AS3333
member-of: RS-FOO
...

as-set:   AS-BAR             aut-num: AS3333
members:  AS3333             ...
...
```

Figure 5: A few objects illustrating membership relations for RPSL set objects

The first one is by listing objects in a *members* attribute in the set object. This is the kind of member relationship present in RIPE 181[2] (eg. as-list in as-macro objects).

The other way of specifying a membership relation is through the use of the *member-of* attribute. This attribute can be used in the *route*, *aut-num* and *inet-rtr*. The value of the *member-of* identifies a set object that this object wants to be a member of.

However, specifying *member-of* is not enough. The set object must also have a *mbrs-by-ref* attribute listing the maintainer of the object wanting to be a member of the set. The set owner must validate the membership claim of an object with a *member-of* attribute, and it does that by matching the *mnt-by* line of the object with one the maintainers in the *mbrs-by-ref* attribute of the set object.

In the example shown in figure 5, object *route: 193.0.0.0/22* is a member of *RS-FOO*, since it has a *member-of* attribute referencing *RS-FOO* **and** is *mnt-by* the maintainer referenced in *RS-FOO*'s *mbrs-by-ref* attribute.

Object *route: 192.168.0.0/24* is not a member of *RS-FOO* because, even though it has a *member-of* attribute referencing *RS-FOO* it is not maintained by the proper maintainer object.

Object *aut-num: AS3333* is a member of *AS-BAR* because *AS-BAR* explicitly says so.

The Database software prevents the insertion of objects with *member-of* attributes that are not maintained by maintainer objects not listed in the *mbrs-by-ref* attribute of the referenced set object.

However, the "owner" of the set object can modify the *mbrs-by-ref* attribute at any time and invalidate the relationship with a series of objects that point to it using the *member-of* attribute. The software will not prevent this operation since the membership relationship shoudl be under the control of the "owner" of the set object.

In order to minimize confusion, a whois query will validate the membership relation at query time and return only the objects with a valid *member-of* relation.

Therefore to find out all the members of a specific set object, the user must issue a whois query for the set object, extract the list of explicit members, with possible recursion, and then issue an inverse query for the objects with the appropriate *member-of* values (recursively, that is including the ones pointing to possible set objects contained in the *members* attribute of the containing set object).

Tools such as the RAToolset include clients that perform this recursive expansion.

2.7.3 New queries for in-addr.arpa domains

Version 3.0 of the RIPE Database supports the -M, -m, -L and -l flags for lookups of domain objects that are in the in-addr.arpa tree. For an example, see figure 6.

```
Query:

  whois -L -Tdn 193.0.0.0/24

Output:

  domain:       193.in-addr.arpa
  descr:        European Regional Registry
  descr:        193.0.0.0/8 CIDR block
  ...
  source:       RIPE

  domain:       0.0.193.in-addr.arpa
  descr:        Reverse for RIPE NCC address space
  ...
  source:       RIPE
```

Figure 6: Example of new query support for in-addr.arpa domain names.

3 Interaction with other databases

Version 3.0 of the RIPE DB will only support RPSL. All responses from the Database will be in RPSL format. Updates to the database itself will need to be in RPSL although during the transition process, updates in RIPE181 format will be accepted and translated to RPSL for storage in the database (see below).

The mirroring of other databases may need translation depending on the format in which they are available.

The database files (except the person and role objects) will be available at ftp.ripe.net as RPSL files. If there is enough community demand, the RIPE NCC can produce software to backtranslate RPSL into RIPE 181, bearing in mind that some information may be lost in the process due to the fact that RIPE 181 can only express a subset of what RPSL can.

4 Transition plan

4.1 Stages of the transition process

1. Phase I

   Near real time mirroring from the current database with translation to RPSL. New server available for query from whois://rpsl.ripe.net New server provides responses in RPSL format (for routing registry objects).

   Estimated time for implementation: End January 2000. Completed.

2. Phase II

   New server processes updates on a test database. The near real time mirror from the production is maintained as above.

   At the end of this phase, the RIPE NCC will scan the routing policies stored in the database to find objects that have used fields other than the intended ones to express policy in RIPE 181 comments (eg. by describing policy in remarks fields) and contact the maintainers of the objects, where possible, in order to raise awareness of the new functionality.

   In this phase extensive testing will be carried out.

   Estimated time for implementation: May 2000. Current phase.

3. Phase III

   The new server goes into production and becomes the only server answering whois queries.

   This can only happen after extensive testing and confidence checks with the RIPE community.

   RIPE-181 input is accepted at < auto-dbm@ripe.net> and translated into RPSL for storage, RPSL input is directly accepted at <auto-rpsl@ripe.net>

   <auto-dbm@ripe.net> will accept creation and modification of objects. Deletions need to be forwarded to <auto-rpsl@ripe.net> because of the interaction between the requirement to exactly reproduce the object to be deleted and the translation to RPSL taking place.

   Estimated time for implementation: **Subject to community consensus**. Expected towards of the end of the year 2000 or early 2001.

4. Phase IV

   RPLS is accepted at <auto-dbm@ripe.net>, becoming the primary means of updating the database. RIPE 181 input is accepted at <auto-181@ripe.net> as a secondary means.

   The duration of this phase should be short.

5. Phase V

   Only RPSL input is accepted. Dependant on the previous phases.

A Templates for RPSL and rps-auth objects

Note: These templates also reflect the modifications required to support the proposed Routing Policy System Security functionality [5].

A.1 Route object

```
route:          [mandatory]  [single]    [primary/look-up key]
descr:          [mandatory]  [multiple]
origin:         [mandatory]  [single]    [primary/inverse key]
holes:          [optional]   [single]              *** hole in RIPE 181 ***
member-of:      [optional]   [multiple]  [inverse key]  *** New in RPSL ***
inject:         [optional]   [multiple]                 *** New in RPSL ***
aggr-mtd:       [optional]   [single]                   *** New in RPSL ***
aggr-bndry:     [optional]   [single]                   *** New in RPSL ***
export-comps:   [optional]   [single]                   *** New in RPSL ***
components:     [optional]   [single]                   *** New in RPSL ***
remarks:        [optional]   [multiple]
cross-mnt:      [optional]   [multiple]  [inverse key]
cross-nfy:      [optional]   [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-lower:      [optional]   [multiple]  [inverse key]  *** New in RPS auth ***
mnt-routes:     [optional]   [multiple]  [inverse key]  *** New in RPS auth ***
mnt-by:         [mandatory]  [multiple]  [inverse key]
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

A.2 Autonomous system object

```
aut-num:        [mandatory]  [single]    [primary/look-up key]
as-name:        [mandatory]  [single]
descr:          [mandatory]  [multiple]
member-of:      [optional]   [multiple]  [inverse key] *** New in RPSL ***
import:         [optional]   [multiple]      ***  as-in in RIPE 181 ***
export:         [optional]   [multiple]      *** as-out in RIPE 181 ***
default:        [optional]   [multiple]
remarks:        [optional]   [multiple]
admin-c:        [mandatory]  [multiple]  [inverse key]
tech-c:         [mandatory]  [multiple]  [inverse key]
cross-mnt:      [optional]   [multiple]  [inverse key]
cross-nfy:      [optional]   [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-lower:      [optional]   [multiple]  [inverse key]  *** From RPS auth ***
mnt-routes:     [optional]   [multiple]  [inverse key]  *** From RPS auth ***
mnt-by:         [mandatory]  [multiple]  [inverse key]
```

```
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

A.3 Internet router object

```
inet-rtr:       [mandatory]  [single]    [primary/look-up key]
descr:          [mandatory]  [multiple]
alias:          [optional]   [multiple]                *** New in RPSL ***
local-as:       [mandatory]  [single]    [inverse key]  *** localas in RIPE 181 ***
ifaddr:         [mandatory]  [multiple]  [look-up key]
peer:           [optional]   [multiple]  [inverse key]
member-of:      [optional]   [multiple]  [inverse key]  *** New in RPSL ***
remarks:        [optional]   [multiple]
admin-c:        [mandatory]  [multiple]  [inverse key]
tech-c:         [mandatory]  [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-by:         [mandatory]  [multiple]  [inverse key]
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

A.4 Autonomous system set object

This object was **as-macro** in RIPE 181

```
as-set:         [mandatory]  [single]    [primary/look-up key]
descr:          [mandatory]  [multiple]
members:        [optional]   [multiple]      *** as-list in RIPE 181 ***
mbrs-by-ref:    [optional]   [multiple]      *** New in RPSL ***
remarks:        [optional]   [multiple]
tech-c:         [mandatory]  [multiple]  [inverse key]
admin-c:        [mandatory]  [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-by:         [mandatory]  [multiple]  [inverse key]
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

A.5 Route set object

This object was **community** in RIPE 181

```
route-set:      [mandatory]  [single]    [primary/look-up key]
descr:          [mandatory]  [multiple]
members:        [optional]   [multiple]  *** New in RPSL ***
mbrs-by-ref:    [optional]   [multiple]  *** New in RPSL ***
remarks:        [optional]   [multiple]
tech-c:         [mandatory]  [multiple]  [inverse key]
```

```
admin-c:        [mandatory]  [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-by:         [mandatory]  [multiple]  [inverse key]
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

A.6 Router set object

New object in RPSL

```
rtr-set:        [mandatory]  [single]    [primary/look-up key]
descr:          [mandatory]  [multiple]
members:        [optional]   [multiple]
mbrs-by-ref:    [optional]   [multiple]
remarks:        [optional]   [multiple]
tech-c:         [mandatory]  [multiple]  [inverse key]
admin-c:        [mandatory]  [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-by:         [mandatory]  [multiple]  [inverse key]
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

A.7 Filter set object

New object in RPSL

```
filter-set:     [mandatory]  [single]    [primary/look-up key]
descr:          [mandatory]  [multiple]
filter:         [mandatory]  [single]
remarks:        [optional]   [multiple]
tech-c:         [mandatory]  [multiple]  [inverse key]
admin-c:        [mandatory]  [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-by:         [mandatory]  [multiple]  [inverse key]
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

A.8 Peering set object

New object in RPSL

```
peering-set:    [mandatory]  [single]    [primary/look-up key]
descr:          [mandatory]  [multiple]
peering:        [mandatory]  [multiple]
remarks:        [optional]   [multiple]
tech-c:         [mandatory]  [multiple]  [inverse key]
```

```
admin-c:        [mandatory]  [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-by:         [mandatory]  [multiple]  [inverse key]
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

A.9 as-block object

New object in RPS auth

```
as-block:       [mandatory]  [single]    [primary/look-up key]
descr:          [optional]   [multiple]
remarks:        [optional]   [multiple]
tech-c:         [mandatory]  [multiple]  [inverse key]
admin-c:        [mandatory]  [multiple]  [inverse key]
notify:         [optional]   [multiple]  [inverse key]
mnt-lower:      [optional]   [multiple]  [inverse key]
mnt-by:         [mandatory]  [multiple]  [inverse key]
changed:        [mandatory]  [multiple]
source:         [mandatory]  [single]
```

References

[1] A. Magee. RIPE Database Documentation. http://www.ripe.net

[2] Tony Bates, Elise Gerich, Laurent Joncheray, Jean-Michel Jouanigot, Daniel Karrenberg, Marten Terpstra, Jessica Yu. Representation of IP Routing Policies in a Routing Registry,http://www.ripe.net

[3] Marek Bukowy, Janne Snabb. RIPE Database Documentation (Update for v2.2.1) , http://www.ripe.net

[4] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, M. Terpstra. Routing Policy Specification Language. RFC 2622

[5] C. Villamizar, C. Alaettinoglu, D. Meyer, S. Murphy. Routing Policy System Security. RFC 2725