Whois queries in RIPE database v3.0

João Luis Silva Damas
Marek Bukowy
Engin Gündüz
Roman Karpiuk
Ambrose Magee
Chris Ottrey
Andrei Robachevksi

Database Group
RIPE NCC

Document: NewDB–1
Date: December 30, 1999

ABSTRACT

This document is a description of the processing of whois queries in the new RIPE Database

Contents

1 Introduction **3**

2 Queries using primary and lookup keys **3**

3 IP address lookups **4**
3.1 Default lookup for IP ranges in the RIPE Database 5
3.2 More and less specific queries 5
3.2.1 More specific queries 5
3.2.2 Less specific queries 6

4 Inverse queries **6**

5 Query support for tools. RAtoolset **7**
5.1 Example output illustrating the RAToolset support flags 8
5.1.1 -K flag 8
5.1.2 -x flag 9
5.1.3 -l flag 9
5.2 Persistent connections and keeping state 9

6 Getting all the members of set objects **11**

7 More/less specific lookups for in-addr.arpa and ip6.int domains **11**

8 Near real time mirroring **12**

9 Referral mechanism for domains **13**

10 Access control for queries **13**

A Whois flags supported by the RIPE whois server v3.0 **15**

B Contents of the database used in the examples **17**

1 Introduction

This document describes the functionality provided in version 3.0 of the RIPE Database to enable users to retrieve data.

The RIPE NCC has aimed at maintaing the whois syntax as close as possible to previous versions of the RIPE DB. However, the version 3.0 supports some new functionality which has required an extension of the query syntax.

Also described here is the newly incorporated mechanism to enable the whois server to automatically track query responses and limit the retrieval of contact information from the RIPE Database. The intention is to limit non-operational uses of the data (such as advertising) while permiting the operational uses of the database to be carried out.

2 Queries using primary and lookup keys

This section describes queries performed on primary and lookup keys as defined in the RIPE Database object templates. For a complete listing of these templates, please refer to [1].

Key type	Object type	Attributes to search in
name	person, role	person and role
e-mail	person, role	e-mail
nic handle	person and role	nic-hdl
Maintainer name	Maintainer	mntner
Key certificate ID	Key certificate	key-cert
Limerick name	Limerick	limerick
Domain name	Domain	domain
Network name	Inetnum and inet6num	netname
IPv4 address or range	Inetnum route	inetnum route
IPv6 address or range	Inet6num	inet6num
IPv4 address	inet_rtr	ifaddr
IPv4 prefix	route	route
AS number	Aut-num	aut-num
AS number or range of AS numbers	AS block	as-block
Domain name	inet-rtr	inet-rtr
AS set name	AS set	as-set
Route set name	Route set	route-set
Router set name	Router set	rtr-set
Filter set name	Filter set	filter-set
Peering set name	Peering set	peering-set

Table 1: Key types, objects and attributes used in primary and lookup key queries.

3 IP address lookups

Probably, the most important service provided by the RIPE Database is to provide information about IP networks in the Internet. This information is stored in the database in the form of inetnum and route objects, among others.

Both inetnum and route objects store information about ranges of IP addresses.

Route objects use prefix notation to specify the range of IP addresses that they contain information about. This type of notation specifies ranges using two components: the prefix and it's length.

For IPv4, the prefix is a 32 bit integer written in dotted quad notation and having the value of the lowest IP address in the range. The prefix length is an integer number in the range 0-32 (eg. 193.0.0.0/22 specifies the range of 1024 IPv4 addresses starting with, and including, 193.0.0.0)

Inetnum objects can also accept other types of notations. The most common one is range notation where the range is explicitly specified as two 32 bit integers written in dotted quad notation separated by a dash (eg. 193.0.0.0-193.0.3.255, which specifies the same range as above).

Other type of notation still seen in some old objects is called classful notation, where certain assumptions are made about the size of a range based on the first few bits of the IPv4 address. This notation is used less often.

When dealing with IPv6 address ranges, only the standard IPv6 prefix notation is allowed (prefix length must be in the range 0-128 and the prefix is a 128 bit integer, written in hexadecimal groups of 2 bytes separated by colons and with the possible use of short hand notation for strings of consecutive 0s).

When querying the database for information about a range of IP addresses, the user can use as search keys the following range notations:

- a prefix, which has the same meaning as above.
- an explicit range, also as above
- a single IP number, which when used as an argument to a query is interpreted as a range of exactly 1 address.

The remaining of this section describes how a user can request different types of information to be returned relative to a particular range of IP addresses.

Before going into details it is useful to define three concepts frequently used in this type of queries and which are defined relative to the user specified (reference) range:

- A **less specific** range is a range that contains the whole of the reference range and is bigger (contains more IP addresses) than the reference range.
- A **more specific** range is a range contained within the reference range and contains less IP addresses than the reference range.
- An **exact match** refers to a range that is identical to the reference range.

3.1 Default lookup for IP ranges in the RIPE Database

When no flags are specified and the query key sent to the whois server is a range of IP addresses (either IPv4 or IPv6, expressed as a single IP address, two IP addresses separated by a - or a range of IP addresses in prefix notation), the RIPE whois server will try to find an exact match for that range.

If an exact match is found, it is returned. If not, a lookup for the smallest less specific range is performed and this is returned.

Queries using the example database shown on Appendix B:

whois 193.0.0.0

returns object #4

whois 193.0.0.0-193.0.2.255

returns object #3

whois 193.0.2.0-193.0.4.255

returns object #3

3.2 More and less specific queries

Some times the exact match is not the desired information. In that case there is a set of flags that modify the response of the whois server.

This set of 4 flags (-M,-m,-L and -l) provide two generic type of queries known as more and less specific queries.

3.2.1 More specific queries

This refers to queries triggered by the use of the *-m* and *-M* flags. These queries will return information about ranges of IP addresses that are fully contained in the user supplied range and contain less addresses.

The *-M* flag requests that **instead** of returning the **exact match**, the server should return **all** the **subranges** completely contained within the user provided range no matter what their size is.

The *-m* is the more specific equivalent of the *-l* flag. It requests that **instead** of returning the **exact match**, the server should return **subranges** that are fully contained within the user provided range. But instead of reporting all existing subranges it will **only return the biggest ranges contained in the user range**. These are usually called **one level more specifics**.

Queries using the example database shown on Appendix B:

whois -m 193.0.0.0/19

returns objects #4 and #5

whois -m 193.0.0.0-193.0.2.255

returns object #4

whois -M 193.0.0.0/8

returns objects #1, #2, #3, #4 and #5

3.2.2 Less specific queries

These refer to queries triggered by the use of the *-l* and *-L* flags and that will return information about ranges of IP addresses that fully contain the user supplied range and may contain more addresses.

The *-L* flag requests that the server return the **exact match**, if any, and all the objects containing information about IP ranges that are **bigger** than the user supplied range and **fully contain** it.

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

Previous versions of the RIPE whois server did not support the kind of behaviour requested by the *-l* flag. It has been added in version 3.0 to support tools like those in the RAtoolset [2].

Queries using the example database shown on Appendix B:

whois -l 193.0.0.0

returns object #4

whois -l 193.0.0.0/25

returns object #4

whois -l 193.0.0.0-193.0.1.255 or whois -l 193.0.0.0/23

returns object #3

whois -L 193.0.0.0

returns objects #1, #2, #3 and #4

whois -L 193.0.0.0-193.0.1.255

returns objects #1, #2, #3 and #4

whois -L 193.0.0.0-193.0.2.255

returns objects #1, #2 and #3

4 Inverse queries

This section describes queries performed on inverse keys as defined in the RIPE Database objects templates. For a complete listing of these templates, please refer to [1].

Inverse query	argument for -i whois flag	Key type
admin-c	ac/admin-c	nic handle or name
tech-c	tc/tech-c	nic handle or name
zone-c	zc/zone-c	nic handle or name
author	ah/author	nic handle or name
notify	ny/notify	e-mail address
cross-nfy	cn/cross-nfy	nic handle
mnt-by	mb/mnt-by	Maintainer name
cross-mnt	ct/cross-mnt	Maintainer name
origin	or/origin	AS number
local-as	la/local-as	AS number
sub-dom	sd/sub-dom	domain name
rev-srv	rz/rev-srv	domain name or IPv[46] address
nserver	ns/nserver	domain name or IPv4 address
member-of	mo/member-of	as set, route set, rtr-set

Table 2: Inverse queries

5 Query support for tools. RAtoolset

The RAToolSet[2] is a suite of routing policy analysis tools maintained by the Information Sciences Institute at USC. Some of the tools in this set access Routing Registry servers via a specialzed whois interface in order to perform their work.

Support for some of the queries required has not been present in the RIPE whois server up to now. Version 3.0 of the RIPE whois server includes support for these query types.

This section describes the additions to the RIPE whois user interface that allow it to support the RAToolset.

The required queries are:

- Return the prefixes of all route objects with a specified origin. Return only the *route* attributes, not the full object
- Return the prefixes of all route objects referenced in a given route-set. Return only the *route* attributes, not the full object
- Return all the members (autnums or as-sets) of a specified as-set. Return only the *members* attributes, not the full object
- Optionally, include support for expansion of the previous query, if the returned value contain references to as-sets, so that the end result contains only a list of autnums. The RIPE database does not support this and it is up to the client to perform the expansion. The RAToolset currently does the expansion itself.

- Return route objects that exactly match a specified prefix.
- Return route objects that exactly match a specified prefix (as above), but return only the *route* attributes.

There are currently several servers that provide access to Routing Registry data through a whois interface. The syntax of the whois query they require to perform the previous set of queries differs from server to server.

Below we explain the new flags introduced in RIPE whois server v3.0 to support the above mentioned functionality.

Flag	Effect
-K	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	Request the server to never return the exact match for a prefix but instead return the closest less specific prefix.
-x	Requests that an exact match on a prefix is performed. Without this flag, if an exact match is not found, the closest less specific prefix is returned.

Table 3: Flags introduced to support the RAToolset in RIPE whois server v3.0

5.1 Example output illustrating the RAToolset support flags

5.1.1 -K flag

Query: Return all prefixes originating from AS #####

-r -K -Trt -ior AS#####

Output:

route: x.x.x.x/y
origin: AS#####

Query: Return as-set AS-#### primary key and corresponding members

-r -K -Tas AS-####

Output:

as-set: AS-####
members: ASxxx, ASyyy,...

Query: Return list of aut-nums that are members-of as-set AS-####

-r -K -Taut-num -imo AS-####

Output:

aut-num: ASxxx

aut-num: ASyyy

5.1.2 -x flag

Query: Normal query (default lookup behaviour)

-r -Troute 193.0.0.0/25

Output:

Returns object #8 from the example database

Query: Request only exact matches

-r -x -Troute 193.0.0.0/25

Output:

Does not return any data (the example database has no /25 object for this prefix).

5.1.3 -l flag

Query: Normal query (default behaviuor)

-r -Troute 193.0.0.0/24

Output:

Returns object #8 from the example database

Query: Normal query (default behaviuor)

-r -Troute 193.0.0.0/25

Output:

Again, returns object #8 from the example database

Query: Request one level less specific

-r -Troute -l 193.0.0.0/24

Output:

Returns object #9 from the example database

Query: Request one level less specific

-r -Troute -l 193.0.0.0/25

Output:

Returns object #8 from the example database

5.2 Persistent connections and keeping state

In order to make more efficient use of the server, a query can request a persistent connection, one that will not be closed by the server after the reply to the whois query has been sent to the client.

This avoids having to setup and tear down a new TCP connection for every query.

The client can request this by sending the -k flag.

During this type of connection the server will keep state for some internal variables.

Currently state is only kept for the list of sources to query.

To exit a persistent connection, send the -k flag as the only query argument to the server or an empty query (\n).

Function	RIPE DB v3.0	current RIPE DB [1]
Persistent connection (keep alive)	-k	-k
Disconnect from persistent connection	-k or \n	-k or \n
Get routes with specified origin. Return only *route* and *origin* attributes	-r -K -Trt -ior *AS#####*	-r -Trt -ior *AS#####* (see [1])
Get prefixes from route objects in a specific route-set	-r -K -Trt -imember-of *RS-####*	not available
Return members of as-set. Only the aut-num or as-set names, not the full objects, are returned	-r -K -Tas[2]*AS-####* **and** -r -K -T autnum -i member-of *AS-####*	-r -Tam *AS-####* (see [1])
Get a full aut-num object	-r -Tan *AS#####*	-r -Tan *AS#####*
Get a full as-set object	-r -Tas[2]*AS-####*	-r -Tam *AS-####*
Get a full route-set (community) object	-r -Trs[2]*RS-####*	-r -Tcm *COMMUNITY*
Get a full maintainer object	-r -Tmt *MNTNER*	-r -Tmt *MNTNER*
Client identification (for server logging purposes)	-V*xxx*	-V*xxx*
Get server version	-q version	not available
Turn off recursive searches	-r	-r
Look for routes. Full object. Exact match [3]	-r -x -Trt *prefix*	-r -Trt *prefix*. If no exact match is found the closest less specific is dispayed
Look for routes. Return only *route* and *origin* attributes of exact matches [3]	-r -x -K -Trt *prefix*	If no exact match is found the closest less specific is dispayed
Look for one level less specific routes. Do not return exact matches. Return full object	-r -l -T rt *prefix*	Will return the exact match if found, otherwise will return closest less specific
Look for all level less specific routes. Return full object	-r -Trt -L *prefix*	-r -Trt -L *prefix*

Function	RIPE DB v3.0	current RIPE DB[1]
Look for all level more specific routes. Return full object	-r -Trt -M *prefix*	-r -Trt -M *prefix*
Request to perform search on all sources	-a	-a
Specify sources to lookup	-s *source*[,*source*,...]	-s *source*[,*source*,...]
Query for current set of sources used in lookups	-q sources	not available
Object delimiter	\n	\n
End of query delimiter	\n\n	\n\n

Table 4: Differences between RIPE DB v3.0 and previous versions in order to support RAToolset type queries.

Arguments to flags in the RIPE whois server can be in either full or short hand notation (i.e. rt=route, an=aut-num, as=as-set, rs=rs-set, mo=member-of, mt=mntner).

6 Getting all the members of set objects

In RPSL[3] there are two ways in which an object can be a member of a set object.

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[4] (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. That is, 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 of the maintainers in the *mbrs-by-ref* attribute of the set object.

7 More/less specific lookups for in-addr.arpa and ip6.int domains

Version 3.0 of the RIPE Database supports the use of the *-m*, *-M*, *-l* and *-L* flags for lookups on reverse delegation domains.

Their use is similar to that described for IP numbers except that lookups have to be on 8 bit boundaries for IPv4 lookups and 4 bit boundaries for IPv6 lookups.

[1]The current RIPE DB always returns full objects. It is not possible to return only a few wanted attributes.

[2]For backwards compatibility you can specify cm in place of rs and am in place of as

[3]Return something only if an exact match is found

8 Near real time mirroring

Near real time mirroring is a mechanism that enables whois servers, other than the main one, to have an up to date copy of all the data in the main server by getting modifications as they are processed by the main server.

From time to time (usually around 15 minutes) the remote server connects to the main server and requests all the modifications that have been processed since the last connection. When all the data is retrieved the connection is closed, until it is time for a new connection.

The RIPE whois server generates a sequence number every time it does a write operation in the database. For the purpose of generating this serial numbers, the server describes all modifications to the database in terms of two atomic operations: deletion and addition.

A modification of an existing object is considered to be a deletion followed by a creation of a new object with the modified data.

As a result of this, when creating or deleting an object one new serial number is generated but when modifying an existing object, two serial numbers are generated (one for the deletion operation and another for the creation operation).

When sending data to a mirror server, the main server will send one of two strings (ADD or DELETE) followed by two newline characters and the correspoding object (either the object as it was before deletion or the object as it should be added).

Near real time mirroring is implemeted with the *-g* flag.

The arguments to this flag are:

-g <source>:<Protocol version>:<first>:<last>

where

source is the string that identifies the Database from which updates are requested (eg. RIPE)

Protocol version must currently be 1.

first is the lowest serial number requested

last is the last serial number requested or the word *LAST* which tells the main server to send all updates up to the last one.

If the values of either *first* or *last* are outside the values of the serial numbers available at the main server an error is returned.

% ERROR: 4: Invalid range: serial(s) *beggining-end* don't exist

where the *beggining* and *end* give the range of unavailable serial numbers.

At the beginning of the data stream the main server will send the following string

%START Version: *Protocol version source first-last*

After the last piece of data is sent to the mirror server, the main server will send the string

%END RIPE

to signal the end of transmision.

9 Referral mechanism for domains

The referral mechanism for domain objects is one of the most recently added functionalities of the RIPE whois server.

The RIPE Database is not an authoritative source of data domain names registered by TLD administrators.

The referral mechanism provides a way for TLD administrators to instruct the whois server to reply to user by fetching data from the TLDs database rather than from local data.

It's implementation consists of two different parts:

Domain name stripping

When no domain object is found in the database with the name specified in the query, the domain name is stripped towards higher level domains (xxx.yyy.zzz becoming yyy.zzz) and the lookup is repeated until a domain object is found or search string becomes empty.

For backwards compatibility, if the domain object found by domain stripping does not contain a refer attribute, then the query result is displayed just as it was in the previous software version:

% No entries found for the selected source(s).

The refer attribute

The *refer* gives domain name administrators the possibility to point the whois server to an authoritative server for information about the domain name.

The syntax for this attribute is

refer: <type of server> <remote server> <TCP port>

The possible values for *type of server* are *SIMPLE*, *INTERNIC*, *RIPE* and *CLIENTADDRESS*.

remote server is the host name or IP address of the TLD's database.

TCP port is the TCP port of the whois server running on *remote server*.

If a query to the whois server results in the retrieval of a local object that contains a *refer* attribute the server will connect to *TCP port* on *remote server* and issue a whois query for the requested domain name.

Whatever is returned is then sent to the user. This mechanism is **disabled** by including the *-R* flag in the original query.

The *CLIENTADDRESS* type will send information to the referral server about the IP address of the real querier.

10 Access control for queries

This section describes the mechanism for controlling how many queries a particular whois client can do to the whois server. The objective is to control and prevent abusive

```
% You have reached the limit of returned contact information objects.
% This connection will be terminated now.
% This is a mechanism to prevent abusive use of contact data in the RIPE Database.
% You will not be allowed to query for more CONTACT information for a while.
% Continued attempts to return excessive amounts of contact
% information will result in permanent denial of service.
% Please do not try to use CONTACT information information in the
% RIPE Database for non-operational purposes.
% Refer to http://www.ripe.net/db/dbcopyright.html for more information.
```

Figure 1: Message sent to client when maximum number of contact information object retrieval is reached

queries of the Database done by systematic queries for contact information potentially used for non-agreed purposes (eg. spam).

Imposing restrictions on queries by whois clients must take several aspects into account.

While we want to impose restrictions on queries for contact information we do not want other kinds of queries to be affected. Even though thousands of queries for objects that contain contact information should be restricted, as there is no Internet operational use for the massive download of contact information, the whois server may receive a lot of queries for other kinds of objects from a particular host which are not a wrongful use of the database (eg. router configuration, statistical studies of IP assignments, etc).

Therefore the mechanism we want to implement should impose restrictions only in the number of person and role objects sent back to a querier.

The limitations must, of course, be not so strict that enough contact information can't be extracted from the database in the events where it is actually needed (eg. network outages, hacker attacks, etc.), since this is one of the major functions of the RIPE Database.

A Whois flags supported by the RIPE whois server v3.0

Flag	Arguments	Action
-a	NONE	Specifies that queries shall look in all available databases. Lookup order is specified in the server configuration file.
-F	NONE	Produce output using short hand notation for attribute names. Kept for compatibility. It is slower than normal queries.
-g	source:version:first:last	Support near real time mirroring
-iac (or -iadmin-c)	Name or nic handle	Return all objects whose admin-c attributes match the query arguments
-itc (or -itech-c)	Name or nic handle	Return all objects whose tech-c attributes match the query arguments
-izc (or -izone-c)	Name or nic handle	Return all objects whose zone-c attributes match the query arguments
-iah (or -iauthor)	Name or nic handle	Return all objects whose author fields match the query arguments
-iny (or -inotify)	e-mail address	Return all objects whose notify attributes match the query arguments
-imb (or -imnt-by)	Maintainer name	Return all objects whose mnt-by attributes match the query arguments
-ior (or -iorigin)	AS#####	Return all objects whose origin attribute match the query arguments
-irz (or -irev-srv)	Domain name or IP address	Return all objects whose rev-srv attributes match the query arguments
-imo (or -imember-of)	RPSL set name	Return all objects whose member-of attributes match the query arguments
-icn (or -icross-nfy)	e-mail address	Return all objects whose cross-nfy attributes match the query arguments
-ict (or -icross-mnt)	Maintainer name	Return all objects whose cross-mnt attributes match the query arguments
-ila (or -i local-as)	AS number	Return all objects whose local-as attributes match the query arguments
-ins (or -inserver)	Domain name or IP address	Return all objects whose nserver attributes match the query arguments
-isd (or -isub-dom)	Domain name	Return all objects whose sub-dom attributes match the query arguments
-ipn (or -iperson)	Name or nic handle	Equivalent to -iac,tc,zc,ah
-iro (or -irole)	Name or nic handle	Equivalent to -iac,tc,zc,ah
-K	NONE	Return only the primary keys of the objects. Exceptions are the set objects where the emphmembers attributes are also returned.

Flag	Arguments	Action
-k	NONE	Toggle connection keep-alive. By default the server will disconnect after each individual query. Use this option to issue multiple queries on the same connection.
-L	NONE	Return all level less specific route or IP objects for this query. Only applies to IP and route lookups
-l	NONE	Request the lookup to return the next level less specific prefix even if there is an exact match.
-M	NONE	Return all level more specific route or IP objects for this query. Only applies to IP and route lookups
-m	NONE	Return first level more specific route or IP objects for this query. Only applies to IP and route lookups
-q	• sources • version	Query the server for: • the current set of sources • version
-R	NONE	Toggle the use of the referral mechanism for domain name queries
-r	NONE	Switch off recursion for contact information after retrieving the objects that match the query key. Recursion is on by default.
-s	Comma separated list of sources (eg. RIPE,APNIC,ARIN)	Specifies which databases (and in which order) are to be used when performing a query.
-T	Comma separated list of object types (in short hand or expanded notation	Restrict the type of objects to lookup in this query
-t	Object type	Request a template for the specified object type
-V	string	Send information about the client to the server
-v	Object type	Request a verbose template for the specified object type
-x	NONE	Return something only if there is an exact match.

Table 5: Flags that can be used in whois queries to the RIPE whois server v3.0.

B Contents of the database used in the examples

```
Object 1

    inetnum:     193.0.0.0 - 193.255.255.255
    netname:     EU-ZZ-193
    descr:       European Regional Registry
    descr:       Europe
    country:     EU
    admin-c:     MK16-RIPE
    tech-c:      OPS4-RIPE
    status:      ALLOCATED UNSPECIFIED
    mnt-by:      RIPE-NCC-HM-MNT
    mnt-lower:   RIPE-NCC-HM-MNT
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 2

    inetnum:     193.0.0.0 - 193.0.255.255
    netname:     EU-ZZ-193-0
    descr:       European Regional Registry
    descr:       Europe
    country:     EU
    admin-c:     MK16-RIPE
    tech-c:      OPS4-RIPE
    status:      ALLOCATED UNSPECIFIED
    mnt-by:      RIPE-NCC-HM-MNT
    mnt-lower:   RIPE-NCC-HM-MNT
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 3

    inetnum:     193.0.0.0 - 193.0.31.255
    netname:     EU-ZZ-990305
    descr:       RIPE Network Coordination Centre
    descr:       Amsterdam, Netherlands
    country:     NL
    admin-c:     DK58
    tech-c:      OPS4-RIPE
    rev-srv:     ns.ripe.net
    rev-srv:     ns.eu.net
    status:      ALLOCATED PI
    mnt-by:      RIPE-NCC-MNT
```

```
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 4

    inetnum:     193.0.0.0 - 193.0.1.255
    netname:     RIPE-NCC
    descr:       RIPE Network Coordination Centre
    descr:       Amsterdam, Netherlands
    country:     NL
    admin-c:     DK58
    tech-c:      OPS4-RIPE
    status:      ASSIGNED PI
    remarks:     used to be two different /24 inetnum objects
    remarks:     until 19990305  (ripe-ncc & ripe-meeting)
    mnt-by:      RIPE-NCC-MNT
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 5

    inetnum:     193.0.2.0 - 193.0.3.255
    netname:     RIPE-NCC
    descr:       RIPE Network Coordination Centre
    descr:       Amsterdam, Netherlands
    country:     NL
    admin-c:     DK58
    tech-c:      OPS4-RIPE
    status:      ASSIGNED PI
    remarks:     RIPE Meetings and other non-permanent uses
    mnt-by:      RIPE-NCC-MNT
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 6

    inetnum:     193.100.0.0
    netname:     PPPPPPPPP
    descr:       QQQQQQQQQ
    country:     EU
    ...
    status:      ASSIGNED PA
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 7
```

```
    inetnum:     193.100.1.0 - 193.100.3.255
    netname:     XXXXXXXX
    descr:       YYYYYYYY
    country:     EU
    ...
    status:      ASSIGNED PI
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 8

    route: 193.0.0.0/24
    descr: a /24
    origin: as3333
    mnt-by: RIPE-NCC-MNT
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 9

    route: 193.0.0.0/23
    descr: a /24
    origin: as3333
    mnt-by: RIPE-NCC-MNT
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 10

    route: 193.0.0.0/22
    descr: a /24
    origin: as3333
    mnt-by: RIPE-NCC-MNT
    changed:     someone@ripe.net 19991231
    source:      RIPE

Object 11

    as-set:      AS-ONE
    descr:       the first as-set. NOTE: The three character string
                   AS- means this is an as-set.
    members:     AS3333 AS766 AS12345 AS-TWO
    admin-c:     DK58
    tech-c:      OPS4-RIPE
    mnt-by:      RIPE-NCC-MNT
```

```
changed:      someone@ripe.net 19991231
source:       RIPE

Object 12

as-set:       AS-TWO
descr:        the second as-set. NOTE: The three character string
                AS- means this is an as-set.
members:      AS65533 AS1
admin-c:      DK58
tech-c:       OPS4-RIPE
mnt-by:       RIPE-NCC-MNT
changed:      someone@ripe.net 19991231
source:       RIPE
```

References

[1] A schema for implementing the RIPE Database on a SQL engine.

[2] Ratoolset http://www.isi.edu/ra/RAToolSet/

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

[4] 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/docs/ripe-181.html