Project Number:	**IST-2001-32603**
Project Title:	**6NET**
CEC Deliverable Number:	**32603/ACOnet/DS/3.3.1/??**
Contractual Date of Delivery to the CEC:	**29 February 2004**
Actual Date of Delivery to the CEC:	**19 May 2004**
Title of Deliverable:	**Contribute and report on discussions on IPv6 support in RIPE database**
Work package contributing to Deliverable:	**WP3**
Type of Deliverable*:	**R**
Deliverable Security Class**:	**PU**
Editors:	**Simon Leinen**
Contributors:	**Carlos Friaças, Wilfried Wöber, Bruno Ciscato, Rob Evans**
Reviewers:	**João Nuno Ferreira**

* Type: P - Prototype, R - Report, D - Demonstrator, O - Other

** Security Class: PU- Public, PP – Restricted to other programme participants (including the Commission), RE – Restricted to a group defined by the consortium (including the Commission), CO – Confidential, only for members of the consortium (including the Commission)

Abstract: This document describes the upcoming new version of the Routing Policy Specification Language, RPSLng, in particular how it adds support for address families beyond IPv4 unicast, so that it can be used to document IPv6 routing policy. This deliverable updates an earlier version with reports from the standardisation process as well as from experiments with a prototype RPSLng registry and RPSLng-aware tools.

Keywords: Routing Policy, Routing Registry, RIPE DataBase, RPSL, RPSLng, RAToolSet, IRRToolSet, Autonomous System

Executive Summary

We explain how the Routing Policy Specification Language (RPSL) is used by Internet Service Providers (ISPs) to document parts of their routing policies in Routing Registries, and to generate network configuration from these data, in particular configuration for the Border Gateway Protocol, version 4 (BGP-4).

RPSLng is a revision of RPSL that is currently being considered for standardisation in the IETF and RIPE communities. The major addition with respect to RPSL is support for address families other than IPv4 unicast. Since RPSLng includes support for IPv6, its deployment and use is an important step towards the integration of IPv6 into the operation of the Internet. We document some design considerations for RPSLng, as well as the current status of discussion in the standardisation effort.

Where the former version of this deliverable (D3.3.1v1) suggested to do some "sandbox tests" with the new language, we now have the possibility to do some realistic testing using RPSLng-aware test registry and software that has been made available by the RIPE NCC. We describe our experiences, both positive and negative, with this prototype environment, and give some real-world example policies at different levels of detail.

Table of Contents

Introduction

The Internet is the interconnection of many independently managed networks, with relatively loose coordination between the operators of those individual networks. Today, a single routing protocol, the Border Gateway Protocol, version 4 (BGP-4) [BGP4], is used to exchange routing (reachability) information between these "autonomous systems" (we're being liberal with the terminology here). The flow of BGP routes is constrained by policies. These policies are defined by operators to control the flow of traffic through their networks based on business objectives in the larger sense.

Operators have the possibility to document their routing policies in a globally distributed set of repositories collectively called the *Internet Routing Registry*. The format of individual entries in those registries has to be standardised to be useful throughout the Internet. To that end a standard language has been defined: the Routing Policy Specification Language - [RFC2622].

The information in the Routing Registry can be used for various purposes, including automatic generation of partial router configuration; in particular the route filters that implement BGP-4 policy. Policy information can be documented in the registry at various levels of detail. Most Internet service providers only publish enough of their external routing policy to permit the construction of useful route filters. Several very large ISPs don't document their routes in the Routing Registry at all, although for the RIPE region, the coverage of the registry data is quite good [NEMECIS].

1. Relationship to other deliverables

The present document does **not** attempt to supersede the first version of this deliverable (D3.3.1v1), but rather complements it. D3.3.1v1 includes an introduction to inter-domain routing on the Internet and the routing registry in general, as well as historic background on the development of the routing registry language. In the text you are reading now, the main focus is on recent developments related to development and standardisation of RPSLng.

The external routing policies for the 6NET core are documented in Deliverable D1.3, *Operational procedures to be followed by organisations connected to 6NET*. These policies are described in prose for several different types of interconnected networks: National Research and Education Networks (NRENs), the GÉANT backbone, the Euro6IX IST project network, other international research networks, and networks of other entities. The rules proposed in D1.3 could be translated into RPSLng, and documented in public routing registries both for the 6NET Autonomous System (AS6680) and for the interconnected networks' ASes.

In WP6, *IPv6 network management architecture and tools*, Deliverables D6.2.3, D6.2.4, and D6.3.3 all discuss IPv6 support in tools and procedures for network operations. RPSLng support will be important for those network management tools that access and interpret routing registry data.

2. Background

2.1. Routing Technology for the Internet

There are no significant developments regarding for inter-domain routing technology to report. BGP-4 is still by far the dominant inter-domain routing protocol on the Internet. It is continually being enhanced with smaller new capabilities such as cooperative route filtering, graceful restart, or security mechanisms such as the generalized "TTL security hack" described in [RFC3682].

The Multiprotocol Extensions to BGP (MP-BGP) [RFC2858] are widely implemented and deployed today, mainly for IPv4 multicast, IPv6 unicast, and BGP/MPLS VPNs [RFC2547].

The need for data that can help network administrators managing "border" network elements (elements that exchange packets with other administrative domains) is still a reality today. Only with considerable amounts of data regarding topology the rapid/automatic path optimization is possible.

In real-world operations, some Internet Domains (Autonomous Systems) are not making their policies public, or fail to keep them up-to-date in the registry, however many others are relying on the published available data to find the best way for their packets.

The hierarchy paradigm described in D3.3.1v1 still applies: Managing one domain in today's Internet is directly tied to the way other domains are managed by third party entities.

2.2. History (Abridged)

For the earlier history of the Routing Registry and its Routing Policy Specification Language, see the previous version of this deliverable, D3.3.1v1.

The common rules used in the Internet Routing Registry today are specified in a set of IETF RFCs:

- RFC2622: Routing Policy Specifiaction Language (RPSL)
- RFC2725: Routing Policy System Security
- RFC2650: Using RPSL in Practice

RPSL as defined in RFC2622 only supports IPv4 unicast, which is implicitly assumed as the single address family in all policy specifications. This restriction had been noticed during the definition of RPSL, but the consensus in the community at the time was to try to get support for the IPv4 unicast address family agreed as fast as possible, and not to get distracted with e.g. multicast issues. The approach proposed was to investigate extensions to RPSL later on.

So this is exactly the focus of the "RPSLng" effort: To add support for the other IP address families and sub-address families, namely IPv6 unicast, IPv4 multicast and IPv6 multicast.

The project to extend RPSL with multi-protocol support is pretty complex, because there are competing goals which require careful analysis, discussion and building a consensus, about the best approach:

- keeping RPSL (the language) as concise and orthogonal as possible

- for the benefit of existing tools and scripts, investigate possibilities to have extensions being "backward compatible"
- minimize the amount of effort required to extend existing tools to support the proposed extensions
- and last but not least -
  should "RPSLng" be encompassing support for all address families (IPv4 unicast and multicast, IPv6 unicast and multicast), and tunnels - or would the incremental approach to only support IPv6 unicast be more realistic?

Quite a while ago (2001), activities have started to define RPSLng, and a global mailing list has been created to focus and support that activity. The messages exchanged on that list can be found on the web at the `rpslng@ripe.net` list archive.

January 2002: `draft-parent-multiprotocol-rpsl-00.txt`

An initial draft proposal was submitted by Florent Parent in January 2002 as `draft-parent-multiprotocol-rpsl-00.txt`. This proposal added `afi` and `safi` qualifiers to many RPSL attributes, but didn't introduce any new classes or attributes. The `import` and `export` attributes of `aut-num` objects would be extended to allow these qualifiers, and `route` objects to allow specification of an `ipv4` or `ipv6` address family using the `afi` qualifier.

May-December 2002: `draft-damas-rpslng-00.txt`

An alternative proposal was initially brought forward by João Luís Silva Damas in May 2002, describing how additional address families *could* be introduced. This proposal tries to keep the existing language elements untouched, in order to avoid breaking existing tools. For example, rather than extending the syntax of the existing `import` and `export` attributes as in `draft-parent`, `draft-damas` introduces new `mp-import` and `mp-export` attributes, where `afi` and `safi` qualifiers are used.

This proposal was then written up by João Damas, Florent Parent and Andrei Robachevksi and submitted as Internet-Draft `draft-damas-rpslng-00.txt` in December 2002. This draft also added syntax to describe tunnels. Other modifications include:

- The `afi` qualifier, which can refer to the following address families:
  - `ipv4`
  - `ipv4.unicast` (equivalent to `ipv4`)
  - `ipv4.multicast`
  - `ipv6`
  - `ipv6.unicast` (equivalent to `ipv6`)
  - `ipv6.multicast`
- New attributes `mp-import`, `mp-export` and `mp-default` in the `aut-num` object class. These attributes can be used to specify AFI-specific policy. Also, IPv6 addresses can be used in `<router-expression>`s and `<filter>`s in `mp-import` and `mp-export`.

- Additional values of `BGP4+` and `MBGP` for `<protocol>` specifications.
- A new `route6` class, similar to `route` but keyed by an IPv6 address prefix.
- A new `mp-members` attribute in the `route-set` class.
- A new `mp-filter` attribute in the `filter-set` class.
- A new `mp-peering` attribute in the `peering-set` class.
- A new `mp-peer` attribute in the `inet-rtr` class.
- A new `interface` attribute in the `inet-rtr`. This is a generalisation of the IPv4-only `ifaddr` attribute. Besides allowing for different address families, it has some syntax to describe tunnels by remote endpoint and encapsulation type.
- A new `mp-members` attribute in the `rtr-set` class.

May 2003-today: `draft-blunk-rpslng-XX.txt`

Larry Blunk took over editorship of the Internet-Draft in March 2003 and submitted `draft-blunk-rpslng-00.txt` in May 2003. This revision added IPv6 `next-hop` addresses. The two new protocol values formerly proposed (`BGP4+` and `MBGP`) were subsumed by `MPBGP`. This version also clarifies how the RPSL security extensions [RFC2725] apply to RPSLng objects.

In July 2003, `draft-blunk-rpslng-01.txt` was issued to clarify the text in several places. The syntax was simplified by removing the `afi` from all attributes except `mp-import`, `mp-export` and `mp-default`.

That version was submitted to the IETF for publication as an RFC at the Proposed Standard level. Because this was an individual submission, i.e. not a product of an IETF working group, and IETF-wide "Last Call" was issued on August 27, opening a one-month public review period.

During this IETF Last Call, there was considerable discussion about the document. An extensive review by Pekka Savola pointed out quite a few omissions, and raised some questions about the best way to transition from the current RPSL registry to an RPSLng one with the least inconvenience for its users. This echoed earlier complaints from Mark Prior, who disliked the introduction of new attributes and classes, arguing that one should simply be able to use the new address families in existing attributes, with automatic downgrading by the software when the data is accessed from a pre-RPSLng client.

Although some respondents on the `rpslng` mailing list denied that there were any problems, the IESG decided that the issues that were raised made a revised document necessary.

On November 20, 2003, `draft-blunk-rpslng-02.txt` was issued. It contained some clarifications in the form of additional explanatory text, and resolved the editorial problems pointed out by Pekka. The only technical change was that the set of tunnel encapsulations was reduced to just GRE and IP-in-IP.

In February 2004, `draft-blunk-rpslng-03.txt` came out. It extended the set of AFIs to abbreviate specifications of congruent policies over multiple address families. The unqualified AFIs

`ipv4` and `ipv6` now designate the union of the respective `.unicast` and `.multicast` SAFIs. The new set of AFIs is:

- `ipv4.unicast`
- `ipv4.multicast`
- `ipv4` (equivalent to `ipv4.unicast, ipv4.multicast`)
- `ipv6.unicast`
- `ipv6.multicast`
- `ipv6` (equivalent to `ipv6.unicast, ipv6.multicast`)
- `any` (equivalent to `ipv4, ipv6`)
- `any.unicast` (equivalent to `ipv4.unicast, ipv6.unicast`)

Additionally, the `afi` qualifier is now optional in `mp-import`, `mp-export` and `mp-default`. Absence of an `afi` qualifier in an `mp-` attribute designates that the policy is valid for all four protocol families (IPv4/IPv6 unicast/multicast).

In April 2004, `draft-blunk-rpslng-04.txt` followed with some clarifications of the "structured" syntax for `mp-import` and `mp-export`.

Dimitrios Kalogeras reported more issues with complex policy specifications involving multiple `refine` and/or `except` stances, which led to the current `draft-blunk-rpslng-05.txt` [RPSLNG] in May. It also clarifies that a trailing semicolon is **not** necessary in the "unstructured" syntax, which is by far prevalent in the routing registry today.

3. RPSLng Tools Available

3.1. Pilot RPSLng Registry

Since May 2003, the RIPE NCC has been operating an experimental registry that understands RPSLng syntax.

It consists of a whois server on host `rpslng.ripe.net`, port 53001, and an e-mail update robot reachable under <rpslng-auto@ripe.net>.

This registry provides a convenient way to experiment with RPSLng. Operators can register RPSLng-enhanced versions of their normal routing registry objects with no impact on the production registry.

The fact that the pilot registry uses not just an obscure server name, but also a non-standard port number (53001) makes its use somewhat awkward, in particular with common whois clients that don't support user-selectable server ports.

3.2. Software

3.2.1. IRRToolSet

In July 2003, the RIPE NCC announced the availability of a new version (4.8.0) of the IRRToolSet (formerly known as RAToolSet), which includes RPSLng support. Version 4.8.1 was released in October 2003. It can be downloaded from `ftp://ftp.ripe.net/tools/IRRToolSet/`.

Since the release, several people have reported problems with IRRToolSet, in particular the following:

1. IRRToolSet is written in C++ for the most part. As of version 4.8.1, the source still isn't compatible with ISO C++, the dialect implemented by most modern C++ compilers including GCC 3. The only relatively easy way to compile this was to use a pre-ISO C++ compiler, such as GCC 2.95 or older.

2. IRRToolSet hasn't tracked recent changes to the RPSLng Internet-Draft, but still implements `draft-blunk-rpslng-01.txt`. It also contains what is believed to be a bug, namedly that it requires trailing semicolons in the unstructured form of `mp-import` and `mp-export` attributes. This may seem minor, but it should be noted that the vast majority of import/export attributes in the database today are of this form, and have no trailing semicolon. So a straightforward translation to RPSLng will lead to confusing and frustrating error messages when IRRToolSet is used.

So far, these reports haven't led to any improvements to the IRRToolSet as distributed by the RIPE NCC. However, Glen Turner from AARNet has prepared an extensive set of patches that make the system compile under GCC 3.3.2. These patches can be found on:

`http://www.aarnet.edu.au/~gdt/patch/irrtoolset/`.

3.2.2. RIPE database server

An RPSLng-capable prototype version of the RIPE database server was made available at the same time as IRRToolSet 4.8.0, in July 2003. It can be retrieved from:

ftp://ftp.ripe.net/ripe/dbase/software/ripe-dbase-RPSLNG.0.1.tar.gz.

As of May 2004, these changes haven't been merged into recent versions of the RIPE database server.

3.2.3. IRRd

IRRd is a registry database server maintained by MERIT, and in particular by Larry Blunk, the current editor of the RPSLng Internet-Draft. The software can be downloaded from:

http://www.irrd.net/.

RPSLng support has been added in IRRd 2.2 Beta in December 2003. The program seems to be updated relatively frequently, and can be expected to track changes to the draft. This should be seen as an attractive alternative to the RIPE whois server for private experiments with RPSLng.

3.3. State of Experimentation in Pilot Registry

In order to assess the amount of testing that has been done using RIPE's experimental RPSLng registry, we prepared a simple script that counts the `aut-num` objects with `mp-import` and `mp-export` attributes in them. The Perl source of the script can be found in an appendix.

Here are the results of an execution of this script performed on 11 May 2004:

```
15 route6 objects found.
AS#      mp-i mp-e    i    e  changed origin-routes
AS559      10    9   38   37 20040511 3FFE:2000::/24, 2001:620::/32
AS1001      1    -    1    - 20030718 1001::/35
AS1103      -    -  104  103 20030417 2001:610::/32
AS1897      -    -   19   19 20030401 2001:B28::/32
AS1930     28   35    -    - 20031211 2001:690::/32
AS2914      -    -    -    - 20030904 2001:418:2002::/48
AS3243      -    -   27   23 20030218 2001:8a0::/32
AS4589   1327 1327    -    - 20040409 2001:6f8::/32
AS5408      5    -    -    - 20040428 2001:648::/35, 2001:648::/32
AS12353     -    -   16   16 20030106 2001:818::/32
AS12859     -    -  146  146 20030507 2001:7b8::/32
AS16245     3    3   27   25 20030702 2001:1448::/32
AS20965     2    -   38   38 20040203 2001:0798::/32
AS25137     -    -    5    5 20030404 2001:b18::/32
```

It can be seen that there were just six non-ficticious `aut-num` objects that actually had any multi-protocol policy in them. This suggests that relatively few operators have practical experience with the proposed RPSLng specification.

Yet there recently has been useful feedback to the `rpslng` mailing list, including by 6NET members, pointing out errors and omissions in the specification as well as in the available tools. This will lead to another new revision of the RPSLng draft soon.

4. Examples

Here are a few examples of actual multi-protocol policies expressed in RPSLng.

4.1. Typical NREN `aut-num` object: AS559, SWITCH

```
aut-num:      AS559
as-name:      SWITCH
descr:        SWITCH, Swiss Education and Research Network
descr:        Peering requests:
descr:        Peering policy: http://www.switch.ch/network/peering_policy.html
descr:        Present at CIXP (Geneva) and TIX (Zurich)
import:       from AS513 action pref=110; accept AS513 AS789 AS777 AS1754 AS2879
AS3350 AS8659
import:       from AS553 action pref=106; accept AS-BELWUE AS-MANDA AS-RLP-NET
import:       from AS789 action pref=100; accept AS789
import:       from AS1257 action pref=100; accept AS-SWIPNET
import:       from AS1299 action pref=70; accept ANY
import:       from AS1836 action pref=100; accept AS-KQCHCUST
import:       from AS2686 action pref=100; accept AS-IGNEMEA
import:       from AS3257 action pref=100; accept AS-TISCALI
mp-import:    afi ipv6.unicast from AS3257 action pref=100; accept AS-TISCALI;
import:       from AS3291 action pref=100; accept AS-PSINETEU
import:       from AS3303 action pref=100; accept AS-SWCMGLOBAL
mp-import:    afi ipv4.multicast from AS3303 action pref=100; accept AS-
SWCMGLOBAL;
mp-import:    afi ipv6.unicast from AS3303 action pref=100; accept AS-
SWCMGLOBAL;
import:       from AS3320 action pref=100; accept AS-DTAG
import:       from AS3549 action pref=70; accept ANY
import:       from AS4589 action pref=100; accept AS-EASYNET
mp-import:    afi ipv6.unicast from AS4589 action pref=100; accept AS-EASYNET;
import:       from AS5669 action pref=100; accept AS-VIA
mp-import:    afi ipv6.unicast,ipv6.multicast from AS6680 action pref=106;
accept ANY;
import:       from AS6719 action pref=100; accept AS-EQUANT-CH
import:       from AS6730 action pref=100; accept AS-GLOBAL AS-SWISS
import:       from AS6772 action pref=100; accept AS-IMPNET
import:       from AS8220 action pref=100; accept AS-COLT
import:       from AS8235 action pref=100; accept AS8235
import:       from AS8327 action pref=100; accept AS-IXPRIME
import:       from AS8404 action pref=100; accept AS-CABLECOM
mp-import:    afi ipv6.unicast from AS8758 action pref=100; accept AS-DOLPHINS;
import:       from AS8833 action pref=100; accept AS-GPS AS-GPS-CUST
import:       from AS9044 action pref=100; accept AS-SOLNET
mp-import:    afi ipv6.unicast from AS9044 action pref=70; accept ANY;
import:       from AS9177 action pref=100; accept AS9177:AS-TSMCHAC
import:       from AS12347 action pref=100; accept AS-MAGNET
import:       from AS12350 action pref=100; accept AS-VTX
import:       from AS12429 action pref=100; accept AS-CYBERNETEU
import:       from AS12654 action pref=100; accept AS12654
import:       from AS13030 action pref=100; accept AS-INIT7
import:       from AS13237 action pref=100; accept AS-LNCTIX
import:       from AS15169 action pref=100; accept AS15169
import:       from AS15600 action pref=100; accept AS-LAN
import:       from AS15623 action pref=100; accept AS-CYBERLINK
import:       from AS16215 action pref=100; accept AS-GENOTEC
import:       from AS20932 action pref=100; accept AS-IP-MAN
mp-import:    afi ipv4.multicast from AS20932 action pref=100; accept AS-IP-MAN;
```

```
import:       from AS20965 action pref=106; accept AS-GEANTNRN AS-GEANTEXTRAS
AS-GEANTNRENPEERS
mp-import:    afi ipv4.multicast,ipv6.unicast from AS20965 action pref=106;
accept AS-GEANTNRN AS-GEANTEXTRAS AS-GEANTNRENPEERS;
import:       from AS20965 action pref=100; accept AS-INFONET
mp-import:    afi ipv4.multicast,ipv6.unicast from AS20965 action pref=100;
accept AS-INFONET;
import:       from AS21494 action pref=100; accept AS-GREEN
export:       to AS513 announce AS-SWITCH AS-BELWUE AS-MANDA AS-RLP-NET
export:       to AS553 announce AS-SWITCH AS-CERNEXT
export:       to AS789 announce AS-SWITCH
export:       to AS1257 announce AS-SWITCH
export:       to AS1299 announce AS-SWITCH AS-CERNUSA
export:       to AS1836 announce AS-SWITCH
export:       to AS2686 announce AS-SWITCH
export:       to AS3257 announce AS-SWITCH
mp-export:    afi ipv6.unicast to AS3257 announce AS-SWITCH;
export:       to AS3291 announce AS-SWITCH
export:       to AS3303 announce AS-SWITCH
mp-export:    afi ipv4.multicast to AS3303 announce AS-SWITCH;
mp-export:    afi ipv6.unicast to AS3303 announce AS-SWITCH;
export:       to AS3320 announce AS-SWITCH
export:       to AS3549 announce AS-SWITCH AS-CERNUSA
export:       to AS4589 announce AS-SWITCH
mp-export:    afi ipv6.unicast to AS4589 announce AS-SWITCH;
export:       to AS5669 announce AS-SWITCH
mp-export:    afi ipv6.unicast,ipv6.multicast to AS6680 announce AS-SWITCH;
export:       to AS6719 announce AS-SWITCH
export:       to AS6730 announce AS-SWITCH
export:       to AS6772 announce AS-SWITCH
export:       to AS8220 announce AS-SWITCH
export:       to AS8235 announce ANY
export:       to AS8327 announce AS-SWITCH
export:       to AS8404 announce AS-SWITCH
mp-export:    afi ipv6.unicast to AS8758 announce AS-SWITCH;
export:       to AS8833 announce AS-SWITCH
export:       to AS9044 announce AS-SWITCH
mp-export:    afi ipv6.unicast to AS9044 announce ANY;
export:       to AS9177 announce AS-SWITCH
export:       to AS12347 announce AS-SWITCH
export:       to AS12350 announce AS-SWITCH
export:       to AS12429 announce AS-SWITCH
export:       to AS12654 announce ANY
export:       to AS13030 announce AS-SWITCH
export:       to AS13237 announce AS-SWITCH
export:       to AS15169 announce AS-SWITCH
export:       to AS15600 announce AS-SWITCH
export:       to AS15623 announce AS-SWITCH
export:       to AS16215 announce AS-SWITCH
export:       to AS20932 announce AS-SWITCH
mp-export:    afi ipv4.multicast to AS20932 announce AS-SWITCH;
export:       to AS20965 announce AS-SWITCH AS-CERNEXT
mp-export:    afi ipv4.multicast,ipv6.unicast to AS20965 announce AS-SWITCH AS-
CERNEXT;
export:       to AS21494 announce AS-SWITCH
default:      to AS1299 action pref=10; networks ANY
default:      to AS3549 action pref=10; networks ANY
admin-c:      WH1101
tech-c:       SNOC1-RIPE
```

```
mnt-by:         SWITCH-MNT
mnt-lower:      AS559-MNT
changed:        simon@switch.ch 20040511
source:         RIPE
```

This RPSLng `aut-num` object adds IPv6 and IPv4 multicast policy in a minimally intrusive way. IPv4 unicast policy is specified using unmodified `import` and `export` RPSL attributes. The `mp-` versions are only used for the other protocols.

The alert reader will have noticed that the `mp-import` and `mp-export` attributes have trailing semicolons, while the `import` and `export` attributes don't. This is necessary to work around a bug in the current version of IRRToolSet, which is described below.

5. Conclusions and next steps

One very basic conclusion (both from the mailing list and from face-to-face discussions in various working groups) is that it is close to impossible to design RPSLng as a formal exercise "on a desk", without regular feedback from networks actively trying to deploy - and to properly manage - internets that support network protocols other than IPv4 unicast. Thus it is essential to "test-drive" the proposed RPSL extensions, in order to find out how those language elements would fit the real world. Only then is it possible to design a reasonably high quality solution (and achieve consensus) in due time.

Looking at RPSLng standardisation today, the testing that has started happening now seems to have a very positive impact on the process. Before real testing started, the RPSLng developers mostly seemed to consider their work as done, and expected the specification to be accepted for publication as an RFC with minor changes. Operators were waiting for this apparently imminent "ratification" and weren't motivated to do any testing.

But RFC publication didn't happen as fast as expected, because there were reservations raised during IETF Last Call and during IESG evaluation. Together with the availability of a pilot server and software infrastructure, this made it possible for a few operators to start testing RPSLng after all. The result was that the specification turned out not to be as "baked" as had been thought.

Now that a dialog between those who specify RPSLng and those who are expected to use it (the operators) has started, there is increasing hope that this process can lead to a well-understood specification and, hopefully, consensus.

The proposed next steps are as follows:

- Ask the 6NET NOC to describe the core routing policy with the proposed RPSLng facilities, and to provide feedback about problems or missing functionality.
- Ask the 6NET project partners to describe their routing interactions with
    - the 6NET core,
    - and with any other IPv6 networks.
- Ask WP1 to document proposed IPv6-related routing policy with a focus on the links to other continents and NREN networks (i.e. ETRI/KR, Internet2/Abilene).
- Have WP3 investigate applicability (or not) of the proposed extensions to RPSL to document the IPv6 multicast routing configuration.
- Ask all project partners to do an investigation on tools and scripts which might be affected by changes to RPSL, or which might benefit from new functions in the routing registry.

6. Appendix A. Acronyms

AFI	Address Family Identifier
AS	Autonomous System
ASN or AS number	Autonomous System number
BGP	Border Gateway Protocol
Euro6IX	an IST project
IETF	Internet Engineering Task Force
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IRRToolSet	Internet Routing Registry Tool Set
I/IS-IS	Integrated Intermediate System to Intermediate System (routing protocol)
MP-BGP	BGP-4 with Multi-Protocol extensions according to RFC 2858
NREN	National Research and Education Network
RAToolSet	Routing Arbiter Tool Set
RFC	Request for Comment (IETF document)
ripe-999	RIPE Document number 999
RPSL	Routing Policy Specification Language
RPSLng	Routing Policy Specification Language (next version being delopped)
WP	Work Package in a project

7. Appendix B. `dump-pilot.pl` Source

```
#!/usr/bin/perl -w
##
## dump-pilot.pl
##
## Simon Leinen  <simon@limmat.switch.ch>
## Created 04-May-2004
##
## Walk an RPSLng registry, counting how much RPSLng there is in
## there.
use strict;
sub get_route6();
sub get_as_policy_counts ($ );
sub whois_query_command_plain ($$);

my $whois_server = "rpslng.ripe.net";
my $whois_port = 53001;
my $cache_p = 1;               # set to 0 to disable jwhois' cache
my $routes = get_route6 ();
printf STDOUT "%d route6 objects found.\n", scalar keys %{$routes};
printf STDOUT ("AS#     mp-i mp-e    i    e  changed origin-routes\n");
foreach my $as (sort { $a <=> $b } keys %{$routes}) {
    my ($mi, $me, $md, $i, $e, $d, $changed)
    = get_as_policy_counts ($as);
    if (defined $mi) {
        printf STDOUT ("AS%-5d %4s %4s %4s %4s %8s %-s\n",
                       $as, $mi || '-', $me || '-', $i || '-', $e || '-',
                       $changed,
                       join (", ", @{$routes->{$as}}));
    }
}
1;

sub get_route6 () {
    my ($route6, $origin, %routes);
    local ($_);
    open (IN, whois_query_command_plain ("route6", "-M ::/0")."|")
        or die "jwhois: $!";
    while (<IN>) {
        if (/^route6:\s*(.*)$/) {
            $route6 = $1;
        } elsif (/^origin:\s*AS(\d+)\s*$/) {
            $origin = $1;
            push @{$routes{$origin}}, $route6;
        }
    }
    close (IN)
        or die "jwhois: $!";
    return \%routes;
}

sub get_as_policy_counts ($ ) {
    my ($as) = @_;
    my ($aut_num,
        $mp_import, $mp_export, $mp_default,
        $import, $export, $default, $changed)
        = (0, 0, 0, 0, 0, 0, 0, 0);
```

```
    local ($_);
    open (IN, whois_query_command_plain ("aut-num", "AS".$as)."|")
        or die "jwhois: $!";
    while (<IN>) {
        if (/^aut-num:/) { ++$aut_num; }
        elsif (/^mp-import:/) { ++$mp_import; }
        elsif (/^mp-export:/) { ++$mp_export; }
        elsif (/^mp-default:/) { ++$mp_default; }
        elsif (/^import:/) { ++$import; }
        elsif (/^export:/) { ++$export; }
        elsif (/^default:/) { ++$default; }
        elsif (/^changed:\s*.* ([0-9]+)/ && $1 ge $changed) { $changed = $1; }
    }
    close (IN)
        or die "jwhois: $!";
    return $aut_num ? ($mp_import, $mp_export, $mp_default,
                       $import, $export, $default,
                       $changed) : undef;
}

sub whois_query_command_plain ($$) {
    my ($type, $name) = @_;
    return "jwhois".($cache_p ? "" : " -d")." -h $whois_server -p $whois_port --
"
        ."-r -T $type $name";
}
```

8. Appendix C. References

[BGP4]

BGP-4, Rekhter, Y., and T. Li, Eds., RFC1771, March 1995

[RFC2858]

Multiprotocol Extensions for BGP-4, RFC2858 T. Bates, Y. Rekhter, R. Chandra, D. Katz, June 2000

[RFC2547]

BGP/MPLS VPNs, RFC2547 Rosen, E. and Y. Rekhter, June 2000

[RFC2622]

Routing Policy Specification Language (RPSL), Alaettinoglu, C., C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg and M. Terpstra, RFC2622, June 1999

[RFC2725]

Routing Policy System Security, Villamizar, C., C. Alaettinoglu, D. Meyer, and S. Murphy, RFC2725, December 1999

[RFC2769]

Routing Policy System Replication, Villamizar, C., C. Alaettinoglu, R. Govindan, and D. Meyer, RFC2769, February 2000

[RFC2650]

Using RPSL in Practice, Meyer, D., J. Schmitz, C. Orange, M. Prior, and C. Alaettinoglu, RFC2650, August 1999

[RPSLNG]

*Routing Policy Specification Language next generation (RPSLng),*Blunk, L., J. Damas, F. Parent, and A. Robachevksi, draft-blunk-rpslng-05.txt, May 2004 (work in progress)

[NEMECIS]

Nemecis: A tool to analyze the IRR Registries, Siganos, G. and M. Faloutsos, NANOG 30, February 2004. PDF available under http://www.nanog.org/mtg-0402/pdf/siganos.pdf

[RFC3682]

The Generalized TTL Security Mechanism (GTSM), Gill, V., J. Heasley, and D. Meyer, February 2004.