Abstract
This document describes the RIPE Database REST API. The purpose of this document is to facilitate discussion and serve as input into a standards process in this area, currently being discussed on the WHOIS-based Extensible Internet Registration Data Service (WEIRDS) mailing list ([https://www.ietf.org/mailman/listinfo/weirds](https://www.ietf.org/mailman/listinfo/weirds)).

Status of this Memo
This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at [http://datatracker.ietf.org/drafts/current/](http://datatracker.ietf.org/drafts/current/). Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress." This Internet-Draft will expire on December 11, 2011.

Copyright Notice
Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents ([http://trustee.ietf.org/license-info](http://trustee.ietf.org/license-info)) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ......................................................... 3
   1.1. Why REST? .................................................... 3
2. Some features of the Query Services ............................. 5
3. Some sample invocation of the query services ................. 6
4. Some details on the CRUD Services .............................. 6
5. Internationalization Considerations ............................ 7
6. IANA Considerations ............................................... 7
7. Security Considerations .......................................... 8
8. References ........................................................... 8
   8.1. Normative References ........................................ 8
   8.2. Informative References ...................................... 8
Author’s Address ......................................................... 8
This document describes a pilot service for querying and displaying data in the RIPE database. The service is implemented using the HTTP protocol [RFC2616] and conforms to the architectural constraints of REST [REST]. In this document, we explain our reasons for new interfaces (and limits of existing ones) as well as high level functionalities of the API. We did not include technical details of our implementation, which can be found as a separate document and is available at:
http://labs.ripe.net/Members/bfiorell/api-documentation.

1. Why REST?

The RIPE Database has a large number of power users: organisations and individuals that make or maintain software which depends on the RIPE Database. The way these specialized clients interact with the RIPE Database is dictated by historical and technical aspects of RPSL and the Whois protocol.

RPSL ([RFC2622]) is the routing policy specification language, but for historical and convenience reasons it actually specifies much more than routing policies, there are many RFCs related to the RPSL (see [RFC2622], [RFC2650], [RFC4012]) and in practice the language describes more than 20 different types of RPSL objects, it is more a formalisation of the way policy records have been stored and exchanged in the existing Whois systems since to the late 80’s instead of a high level domain language specification.

The result is that the policy specification language and its extensions are tightly coupled with the way policies are stored in the existing Whois systems and vice versa, so any new system that is implementing the language ends up reproducing a Whois system.

The qualities of RPSL and Whois are well known. In this section we highlight their limits by describing the way a client would interact with the two Whois interfaces of query and update.

First scenario, a client needs to extract sensible information from one or more Internet Registries like the RIPE Database, the steps are:

- prepare one or more queries and invoke a command line Whois client program or open a TCP socket connection to one or more Whois servers,
- parse and post-process the RPSL responses, facing many intricacies like continuation lines, end of line comments, comma separated
values, attribute values that can reference to different types of RPSL objects depending on various conditions, etc., if object references need to be resolved, extract referenced primary keys, execute other queries, parse again RPSL to identify the right referenced objects; this step can be recursive, merge results from multiple queries, maybe queries from multiple internet registries that may adopt different RPSL flavours, extract subsets of objects, by applying various criteria, extract subsets of attributes from set of objects, by applying various criteria.

The other standard scenarios is that of update: in case of update or delete, fetch the object, this operation may require several of the steps described in the previous workflow, in case of a new object, build RPSL text, maybe starting converting business data into RPSL text, decorate the object with authorisation credentials, adding password hashes to the RPSL text or signing the RPSL text with a PGP key, send the text data via mail to the Mail Update interface or via HTTP post to the Syncupdates interface, in case of error, parse a human readable text response if specific error conditions must be handled in a special way.

As shown above, such processes are modeled on a human centered workflow, they are not the optimal to build new services, extend existing ones or build tools on top of them. Concretely, this means clients have to handle too much complexity, many clients have problems at handling all the complexity of RPSL, many are too expensive to be extended and difficult to maintain. RPSL and Whois are still invaluable but it’s evident that there is a need for simpler interfaces and machine ready data formats in order to simplify the development of services and tools and increase the value of Registries by exposing new domain specific interfaces. The first step can be the identification of a layer of Service Provider Interfaces and a data schema simple enough to be agnostic of
the underlying Registry implementation.
The SPI would sit on top of a Registry Database allowing the composition of domain specific Services but also making real-time interoperation between Registries and services on multiple authoritative Databases possible to achieve.
With these forces and this vision the RIPE Database Group has been prototyping the RIPE Whois RESTful Query and CRUD API and the related XML schemas.
The choice of REST has been quite obvious, HTTP is nowadays the most accessible protocol and it represents a good architectural blueprint for stateless services, with its different HTTP methods, the resource locator protocol, the HTTPS, etc.
For the representation schema we have designed a very relaxed attribute oriented XML Schema. We only apply a structural validation via XSD, after some testing we decided to remove any form of attribute or type validation in order to reuse the same schema on different RPSL flavors.
The services support representational styles JSON, HTML and plain text, all derived via XSL transformation, with the latter two to showcase the transformation powers of XML and also to demo resource navigation via lookup references on any HTML browser.
Content negotiation can be done using HTTP headers or appending a file extension to the request URL.
The query services can be used on any RPSL based Whois server or mirror.
For example in one request it is possible to execute the same Lookup or Search request on all the Regional Internet Registries and return all the responses as a unique set of resources.
Similarly can be done for the CRUD interfaces, building adapter modules for the different update mechanisms provided by different Registries given that they adopt the same set of resource types.
Actually we implemented an adapter for the Whois Syncupdates interface in the form of an HTTP proxy.

2. Some features of the Query Services
Single resource Lookup Service, given a primary key a type and a registry it always return one and only one object. It can also be used to identify resources by URL bookmark.

Resolution of referenced resources. Given a resource, all the attribute values that represent references to other resource contain an xlink anchor that can be followed to navigate and browse networks of resources.

HATEOAS, the client can navigate through any network of resources via xlinks without requiring any stateful information to be stored on the servers, this comes as a benefit of the two previous features.

Normalisation of comma separated values, when they represent references to multiple resources.

Normalisation of continuation lines, end of line comments and other RPSL intricacies. This is still a work in progress and still not complete because is difficult to test all the side effects of this operation. The only real solution would be to deprecate continuation lines in RPSL and split them into multiple attribute value pairs.

All the Query Services are available also on HTTPS.

3. Some sample invocation of the query services
   A Lookup Service URL for an organization object: http://apps.db.ripe.net/whois/lookup/ripe/organisation/ORG-BME1-RIPE
   A Lookup Service URL for a route object: http://apps.db.ripe.net/whois/lookup/ripe/route/193.6.23.0/24/AS2547
   A Search Service URL on multiple registries: http://apps.db.ripe.net/whois/search.xml?flags=r&source=ripe&source=apnic&source=afrinic&query-string=AS2547

4. Some details on the CRUD Services
   The CRUD Services are an attempt to split the single overloaded generic update interface provided by Mail Updates and Syncupdates into separate low level interfaces, each defining a simpler contract, designed for programmatic use rather than for human interaction.
For example the existing update interfaces always require clients to provide the entire RPSL resource in exactly the same text-format as it is stored in the Whois database, even if the requested operation is just a deletion. A client will have to fetch the RPSL object from the RIPE Database doing an appropriate query, then extract only the text blob that represent the resource to be deleted and finally it will have to execute a request to the update service providing the entire text and adding delete attributes and authorisation credentials.

The new CRUD Services on the other hand provide a Delete interface that only requires a primary key, a type, a registry identifier and one or more passwords. It is the equivalent of a lookup but is executed with the HTTP Delete method. It is exposed only on HTTPS. The client will just have to send a request like: HTTP DELETE: https://lab.db.ripe.net/whois/delete/test/person/pp16-test?password=123:

The server will just respond with an HTTP Status code indicating success or failure, in case of failure different status codes will indicate different error conditions.

On top of Create, Update and Delete methods we also prototyped some attribute modification services that in one only request can implement complex update workflow.

For example with one only HTTP request will be possible to execute commands like:

- replace all the attributes of a given type with a new set of attribute,
- remove all the attributes that have value matching the given regular expression,
- add a new set of attributes after the Nth attribute of type X.

5. Internationalization Considerations
TBA

6. IANA Considerations
TBA
7. Security Considerations

TBA.

8. References

8.1. Normative References


8.2. Informative References


Author’s Address
Benedetto Fiorelli
RIPE NCC
P.O. Box 10096
Amsterdam 1001EB
The Netherlands
Phone: +31.20.535.4444
Email: bfiorell@ripe.net