Proposed REG-RWS Extensions

June 10, 2015

Andrew Newton, Chief Engineer, ARIN

Background

This document describes proposed extensions to ARIN's Registration RESTful Web Service (REG-RWS). This service is the primary means used by network operators to programmatically interact with ARIN's registration system. Information on REG-RWS can be found here.

These extensions are intended to make development of software using REG-RWS easier in the areas of child network reassignment or reallocation reporting (commonly known as SWiP) and management of Resource Public Key Infrastructure (RPKI). By making these improvements ARIN customers can more easily maintain their assignment and allocation data, improving ease and reporting accuracy.

This proposal is based on discussions from the ARIN Technical Discussions Mailing List (arin-tech-discuss).

Problem Statement

SWiPs

REG-RWS has all the necessary methods and payloads to conduct SWiP activity. However, there are two significant issues with the current SWiP methods and payloads:

1. The network, recepient organization/customer, and point of contact (POC) methods are separate actions. While this separation yields much greater flexibility than is currently offered by the email templates, they are far more difficult to use for the majority of simple use cases.
2. The current method used to determine if an operator's network space is currently reassigned or reallocated is cumbersome, requiring clients to request reassignment reports, wait for them to be generated, and then parse them. (Note: there is no automatic means for doing this with email templates as

well). Additionally, the current methods require operators to correlate natural keys such as IP addresses to ARIN registration handles.

To mitigate these difficult to implement scenarios, this proposal recommends the implementation of methods for the following: 1. "one-shot" SWiP functions (similar to and modeled on the functionality of the email templates). 2. Methods for checking reassignment or reallocation space based on network addresses. 3. A method for bulk update of network registrations.

RPKI

REG-RWS currently has one RPKI method for Route Origin Authorization (ROA) in the hosted RPKI. While this method makes it easy to create initial ROAs, the subsequent management of ROAs is difficult because there is no means to determine which ROAs exist nor which resources are covered by the RPKI.

Additionally, ROA creation is not necessarily straightforward in networks with reallocations as a covering ROA may accidentally invalidate routes to reallocated networks.

To mitigate these shortcomings, this proposal recommends the implementation of the following methods:

1. A method to list active ROAs.
2. A method to remove active ROAs.
3. A method to list resources covered under RPKI.
4. A method to suggest a ROA which does not cover reallocated networks.

Namespace

All XML extensions in this document will use the XML namespace `http://www.arin.net/regrws/extensions/v1`.

Methods and Payloads

SWiP Payload

The SWiP payload is a new payload used in the new SWiP methods. It is a container for XML types currently defined in REG-RWS.

The SWiP payload has two forms: 1) one containing a network, an organization, and organization POCs, and 2) one containing a network and a customer. Each form creates a network and related organizations/POCs and/or customers.

The basic structure of a SWiP payload containing a network, an organization, and the organization's POCs is as follows:

```
<ext1:swip
  xmlns:ext1="http://www.arin.net/regrws/extensions/v1"
  xmlns="http://www.arin.net/regrws/core/v1">

  <net>
    ...
  </net>

  <org>
    ...
  </org>

  <ext1:pocs>
    <ext1:poc function="XX">
      <poc>
        ...
      </poc>
    </ext1:poc>
  </ext1:pocs>

</ext1:swip>
```

The `<net>`, `<org>`, and `<poc>` elements are defined in the `http://www.arin.net/regrws/core/v1` namespace. The `<ext1:poc>` element is used to contain a `<poc>` element and assign the POC a function ('AB', 'AD', 'N' or 'T' for abuse, administrative, NOC, and technical) to the organization defined in `<org>`.

When used within a SWiP payload, the `<net>` element must not have the following elements or they must be empty:

1. registrationDate
2. orgHandle
3. handle
4. customerHandle
5. pocLinks

Additionally, there should only be one `<netblock>` within the `<net>` element, and it must not contain the `<type>` or `<description>` elements or they must be empty.

When used within a SWiP payload, the `<org>` element must not have the following elements or they must

be empty:

1. registrationDate
2. handle

The `<org>` element may have a `<pocLinks>` child element, but it is not necessary if the SWiP payload contains `<poc>` elemeents. If a `<pocLinks>` element is given and `<poc>` elements are also given, the organization will be linked to new POCs created specified by the `<poc>` elements and will also be linked to the POCs specified by the `<pocLinks>` element.

If the `<org>` element does not contain a `<pocLinks>` child element, the SWiP payload must specify POCs with the `<swip:pocs>` element. When used within a SWiP payload, the `<poc>` element must not have the `<registrationDate>` and `<handle>` elements or they must be empty.

The basic structure of a SWiP payload containing a network and a customer is as follows:

```
<ext1:swip
  xmlns:ext1="http://www.arin.net/regrws/extensions/v1"
  xmlns="http://www.arin.net/regrws/core/v1">

  <net>
    ...
  </net>

  <customer>
    ...
  </customer>

</ext1:swip>
```

The `<customer>` element is defined in `http://www.arin.net/regrws/core/v1`. When used in a SWiP payload, is must not have the `<registrationDate>` or `<handle>` elements or they must be empty.

SWiP Methods

Thw SWiP methods use the SWiP payload to approximate all the functions of the SWiP email templates in one action. That is the SWiP methods will create child networks, create organizations and related POCs (if necessary), and/or create customer records.

The SWiP methods are:

1. PUT /rest/swip/reassign?apikey=APIKEY

2. PUT /rest/swip/reallocate?apikey=APIKEY

If successful, these methods will return a `<ticketedRequest>` payload as defined by the current /net/XXXX/reassign and /net/XXXXX/reallocate methods.

Network Registrations Payload

The Network Registrations payload is a container for SWiP payloads and expresses network registration types and hierarchies.

The basic structure of a Network Registrations payload is as follows:

```
<ext1:networkRegistrations
  xmlns:ext1="http://www.arin.net/regrws/extensions/v1"
  xmlns="http://www.arin.net/regrws/core/v1">

  <ext1:allocation>
    <ext1:swip>
      ...
    </ext1:swip>
    <ext1:assignment>
      <ext1:swip>
        ...
      </ext1:swip>
    </ext1:assignment>
    <ext1:assignment>
      <ext1:swip>
        ...
      </ext1:swip>
    </ext1:assignment>
  </ext1:allocation>

  <ext1:assignment>
    <ext1:swip>
      ...
    </ext1:swip>
  </ext1:assignment>

</ext1:networkRegistrations>
```

This payload is used by methods for finding networks and bulk updating registration data. It is made up of three elements: * `<ext1:assignment>` - describes a network assignment * `<ext1:allocation>` - describes a netowrk allocation * `<ext1:unregistered>` - describes unregistered network space.

Inside of each of these elements, the first child element is a `<ext1:swip>` element (as defined above) describing the network registration and associated registration information.

The `<ext1:allocation>` element may also have `<ext1:assignment>` and `<ext1:allocation>` child elements. These child elements further define an allocation.

The `<ext1:unregistered>` element will contain a `<ext1:swip>` element with minimal subelements and information. The SWiP information will have no customer or organization information, and the `<net>` element will only have network address information indicating the unregistered space.

Find Networks Method

To determine if network space has been reallocated or reassigned, the following method is to be used:

- GET /rest/net/XXXX/YY?apikey=APIKEY

where XXXX is an IP address and YY is a CIDR network length. Because this method could potentially be very compute intensive, it will return a Ticketed Request payload already defined in Reg-RWS. The Ticketed Request payload will contain a Network Registrations payload if the request can be fulfilled immediately. Otherwise it will contain a Ticket payload to be used for retreiving the Network Registrations payload once the ticket has been processed.

Queries outside of the address space directly allocated or assigned will yield an error.

Update Network Registrations Method

The Update Network Registrations method takes a Network Registrations payload and has the following form:

- PUT /rest/nets?apikey=APIKEY

The method will replace all sub-network registration information for all direct allocations for all organizations associated with the API Key. The Network Registrations payload need not have `<ext1:unregistered>` elements.

Because this method could potentially be very compute intensive, it will return a Ticketed Request payload (as defined in Reg-RWS). The Ticketed Request payload will contain a Network Registrations payload if the request can be fulfilled immediately. Otherwise it will contain a Ticket payload to be used for retreiving the Network Registrations payload once the ticket has been processed. This Network Registration payload will reflect the registration data as if a Find Networks query had been executed.

Find Unregistered Network

While the Find Networks Method returns unregistered network space for which an operator can determine network space that can be allocated/assigned, in the simple use case of finding unregistered space under a direct allocation the Find Unregistered Network method may be easier to use. This method has the following form:

- GET /rest/unregisteredNet/XXXX/YY;size=ZZ?apikey=APIKEY

where XXXX is an IP address and YY is a CIDR network length. The size parameter determines the size of the network being sought. This method finds unregistered network space of the size ZZ directly under the direct allocation that encloses XXXX/YY. This method returns a `<net>` object in the form that can be found in `<ext1:unregistered>`.

The RPKI Resources Payload

The RPKI Resources payload enumerates IP network and autonomous system resources certified in the RPKI by a network operator. Resources certified in the RPKI appear in RPKI Resource Certificiates (RCs). IP network resources may also be used in RPKI Route Origin Authorization (ROA) objects.

This payload has the following form:

```
<rpkiResources xmlns="http://www.arin.net/regrws/extensions/v1">

  <autonomousSystemNumbers>
    <autonomousSystemNumber>...</autonomousSystemNumber>
    ...
  </autonomousSystemNumbers>

  <networks>
    <network handle="XXXXX">
      <cidr>XXXX/YY</cidr>
      <cidr>XXXX/YY</cidr>
      ...
    </network>
    ...
  </networks>

</rpkiResources>
```

The RPKI Resources Method

The following method:

- GET /rpki/ORGHANDLE/resources

will return an RPKI Resources payload listing the RPKI certified resources for the organization identified by ORGHANDLE.

The ROAs Payload

The ROAs payload contains ROA information. It has the following form:

```
<roas xmlns="http://www.arin.net/regrws/extensions/v1">

  <roa>
    <id>XXXXXXXXXXXXXXXXXXX</id>
    <name>ipsum lorem</name>
    <validityStartDate>mm-dd-yyyy</validityStartDate>
    <validityEndDate>mm-dd-yyyy</validityEndDate>
    <originAS>ZZ</originAS>
    <prefixes>
      <prefix>
        <startAddress>XXXXXXX</startAddress>
        <cidrLength>YY</cidrLength>
        <maxLength>LL</maxLength>
      </prefix>
      ...
    </prefixes>
  </roa>
  ...

</roas>
```

The child elements of `<roa>` are the fields that would appear in the ROA, with the exception of the `<id>` element. The `<id>` element is a synthesized identifier of the ROA.

The ROAs Method

The following method:

- GET /rest/roas/ORGHANDLE;resourceClass=RESOURCECLASS?apikey=APIKEY

will list all the active ROAs for an organization identified by ORGHANDLE for the resource class given by RESOURCECLASS. Valid values of RESOURCECLASS are:

- AR (representing ARIN)

- AP (representing APNIC)
- RN (representing RIPE)
- LN (representing LACNIC)
- AF (representing AFRINIC)

If no RESOURCECLASS is given (the paramter is omitted), then ROAs from all resource classes are given.

This method will return a ROAs payload.

The Delete ROA Method

The following method:

- DELETE /rest/roa/XXXXXXX?apikey=APIKEY

where XXXXXX is the identifier as found in the ROAs payload will delete the ROA identified by XXXXXXX. This method returns a ROAs payload reflecting the deleted ROA.

The Suggest ROA Method

The following method:

- GET /rest/suggestedRoa/ORGHANDLE;resourceClass=RESOURCECLASS?apikey=APIKEY

will return a ROA payload (as specified by the `http://www.arin.net/regrws/rpki/v1` namespace. The `<roaData>` element will contain a suggested ROA for that organization in the RESOURCECLASS specified. The prefixes in the ROA will not contain reallocated prefixes, as is best current practice. As best current practice may change in the future, resources covered in the suggestion may change as well.

The `<signature>` element will be empty. Clients must sign the `<roaData>` element and submit the ROA paylaod with the appropriate GET method.