Network Working Group
Internet-Draft
Updates: 2622, 4012 (if approved)
Intended status: Informational
Expires: 25 August 2025

S. Romijn
Reliably Coded
J. Bensley
Inter.link GmbH

21 February 2025

Registry scoped members for RPSL set objects
draft-romijn-grow-rpsl-registry-scoped-members-01

Abstract

This document updates RFC2622 and RFC4012 by specifying src-members, a new attribute on as-set and route-set objects in the Routing Policy Specification Language (RPSL). This attribute allows a specific registry to be defined for each member in a set, avoiding problematic ambiguity when resolving set members. A new validation rule allows gradual upgrades and backwards compatibility.

Table of Contents

1. Introduction 2
1.1. Requirements Language 4
2. The src-members attribute 4
2.1. as-set class 4
2.2. route-set class 4
2.3. Resolving members through src-members 5
2.3.1. Resolving example 5
3. Relation to (mp-)members 7
3.1. Additional validation 7
3.2. Automatic generation of (mp-)members 8
3.3. Multiple references to the same primary key 9
4. IANA Considerations 9
5. Security Considerations 9
6. Normative References 9
Authors' Addresses 10

1. Introduction

The Routing Policy Specification Language (RPSL) [RFC2622] defines the as-set and route-set objects, extended in [RFC4012]. These are among the most common objects in the Internet Routing Registry (IRR) system. These sets can either reference a direct member of the set (such as an AS number, IP prefix, etc.), or additional sets which themselves have their own direct members and/or reference yet more sets, ad infinitum. In both cases, this referencing uses the members and mp-members attributes [RFC4012]. Server and client software can follow these references to resolve a set down to its members, a set of prefixes or ASes.

A set may refer to another set by including the primary key in its (mp-)members attribute. The referred set may be in the same or in another IRR registry. It is not possible to specify the IRR registry of the referred set. This can lead to primary key collisions when resolving a set:

1. There are multiple significant IRR registries.

2. Sets often reference objects in registries other than the registry the set itself is stored in.

3. There is no guaranteed uniqueness of object primary keys amongst the different registries.

4. Hence, multiple objects may exist in the IRR system with the same primary key, making references to them ambiguous.

5. Many IRR servers will mirror data from multiple IRR registries, meaning that even within a single server, there are usually collisions.

The ambiguity encountered when resolving set members can result in either an incorrect RPSL object being chosen, because an object with the same primary key was retrieved from the wrong IRR registry, or the required RPSL object (which does exist) is not found, because the resolving process didn't try to retrieve the object from the correct IRR registry. "Incorrect" and "wrong" in this context meaning: not as intended or expected by the operator.

Including members from the incorrect RPSL object can result in the computation of unintentional routing policy information, which is then deployed to network infrastructure. That could cause route leaking, or worse, aid in route hijacking. This has been seen multiple times on the public Internet.

If intended policies are not included, because the object was not found, prefixes that should be accepted, are not, and the prefixes are not reachable.

With either case, routing policy information may end up missing and connectivity may be disrupted.

There is no current way to prevent such ambiguity during set member resolution, both for operators who create the legitimate objects and those who try to resolve them.

Two previous enhancements to reduce set name collisions have been standardized. However, the problem persists:

* [RFC2622] Section 5.1 defines hierarchical set names, such as AS65000:AS-EXAMPLE which may also have additional authorization requirements for the referred aut-num. However, this authorization only works within a single IRR registry, and doesn't allow the correct external IRR to be specified, if the object in question is not local to the IRR registry storing the referring set.

* [RFC2725] Section 9.6 defines external repository (IRR) references. This allows for the correct IRR registry to be specified for a set member object by using the SOURCE:: notation however, this syntax isn't supported in the members field of set objects.

To solve this, this document adds src-members to as-set and route-set objects, using a IRR registry name prefix with a double colon. For example: "RIPE::AS-EXAMPLE", to refer specifically to an object "AS-EXAMPLE" in the IRR registry "RIPE". This format is already widely used informally by operators, including in platforms such as PeeringDB. Continued availability of existing (mp-)members attributes together with new validation rules, ensures backwards compatibility.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. The src-members attribute

2.1. as-set class

The new src-members attribute on as-set is similar to the members attribute from [RFC2622], except that the AS set name MUST be prefixed with a registry name and a double colon.

Attribute:
   src-members

Value:
   list of ([as-number] or [registry-name]::[as-set-name])

Type:
   optional, multi-valued

2.2. route-set class

The new src-members attribute on route-set is similar to the mp-members attribute from [RFC4012], except that any references to set names MUST be prefixed with a registry name and a double colon.

Attribute:
   src-members

Value:
list of ([ipv4-address-prefix-range] or [ipv6-address-prefix-range] or [registry-name]::[route-set-name] or [registry-name]::[as-set-name] or [as-number] or [registry-name]::[route-set-name][range-operator])

Type:
optional, multi-valued

2.3. Resolving members through src-members

IRR software that resolves the members of a set MUST have support for objects with and without src-members. These objects may be encountered if they were created or updated before adoption of src-members, or the objects have not been updated since.

The resolving process is: 1. The resolver MUST include all members listed in the src-members attribute, if any. To find the referenced sets, the resolver MUST match on both the IRR registry name and the set's primary key. If the IRR registry is unknown to the resolver, no set can match the reference. 1. The resolver MUST include all members listed in the members and mp-members attributes, when their primary key was not already listed in src-members. If there are multiple sets with a primary key known to the resolver, the behavior is not defined by this document as this was a previously existing problem.

Note that the restriction to a specific IRR registry name is only used to select the correct IRR registry to retrieve the referred object and its attributes. During recursive resolving, if that set has references to further sets, those MUST be retrieved from a potentially different registry. This could be either the registry specified in the src-members attribute, if present, or the existing source selection algorithm the IRR server currently uses when resolving using (mp-)members. In other words, the restriction of the lookup to a specific IRR registry does not cascade.

2.3.1. Resolving example

```
route-set: RS-FIRST
members: RS-SECOND
mp-members: RS-LEGACY
src-members: RIPE::RS-SECOND
source: EXAMPLE

route-set: RS-SECOND
members: RS-THIRD
source: RIPE

route-set: RS-SECOND
members: AS65002
source: OTHER

route-set: RS-THIRD
members: AS65000
source: OTHER

route-set: RS-LEGACY
members: AS65001
source: OTHER
```

Figure 1: Example objects for recursive lookups and attribute interactions

To perform a recursive lookup of RS-FIRST, the IRR software will: 1. Look up RS-FIRST. 1. Determine that the members of RS-FIRST are RS-SECOND (RIPE registry only) and RS-LEGACY (any registry). The mention of RS-SECOND in the members attribute is not included, as RS-SECOND is already listed in src-members. 1. Look up RS-SECOND in RIPE, and RS-LEGACY in any registry. 1. Determine that the members of RS-LEGACY are AS65001, and the members of RS-SECOND are RS-THIRD. 1. Look up RS-THIRD in any registry. 1. Determine that the members of RS-THIRD are AS65000.

If all mentioned registries are enabled, RS-FIRST would resolve to AS65000 and AS65001.

The RS-SECOND object in the OTHER registry is never looked up, and AS65002 is not included. This would happen even if there was no RS-SECOND object found in RIPE.

3. Relation to (mp-)members

Existing IRR software will not be aware of the new src-members attribute and instead refer to (mp-)members. This is also why the existing attributes are not modified - this existing software would consider e.g. RIPE::AS-EXAMPLE as the full primary key of a set, and fail to look up the reference as intended.

Existing IRR objects may also not be updated with src-members for some time, as this cannot be done automatically. Or, they may be partially updated as for large sets, finding the intended IRR registry references may take some time. Deployment in both software and objects will be a gradual process, however, even partial deployment will reduce the potential for issues from reference mixups.

In order to keep support for existing IRR software, the contents of src-members must match (mp-)members as close as possible, which the IRR server will ensure.

3.1. Additional validation

When an authoritative IRR registry processes a set object with a src-members attribute, it MUST validate that all references in src-members, with the registry names removed, are also listed in members or mp-members. All values MUST be combined, regardless if they were listed in one attribute, or in multiple repetitions of the attribute.

This ensures that the new src-members can be used, providing the benefits for updated resolver software, while still having a consistent (mp-)members available for older software.

IRR registry software is RECOMMENDED to make the src-members attribute mandatory on all new as-set/route-set objects, and MAY make it required when modifying existing objects.

Example of a valid object:

```
route-set: RS-EXAMPLE
members: 192.0.2.0/24
mp-members: 2001:db8::/32
mp-members: RS-OTHER
src-members: 192.0.2.0/24, RIPE::RS-OTHER, 2001:db8::/32
source: EXAMPLE
```

Figure 2: Valid object

Example of an invalid object:

```
route-set: RS-EXAMPLE
members: 192.0.2.0/24
mp-members: 2001:db8::/36
mp-members: RS-OTHER, RS-MPMBRONLY
src-members: 192.0.2.0/24, RIPE::RS-OTHER
src-members: NTTCOM::RS-SRCMBRONLY, 2001:db8::/32
source: EXAMPLE
```

Figure 3: Invalid object: inconsistent inclusion of NTTCOM::RS-SRCMBRONLY, and 200:db8::/36 vs /32. Note: the inclusion RS-MPMBRONLY only in mp- members is permitted.

3.2. Automatic generation of (mp-)members

Managing multiple copies of the same records is tedious for users. Therefore, IRR registry software is RECOMMENDED to automatically fill (mp-)members, if not specified by the user, based on src-members, in authoritative objects, when the user creates or updates the object.

Specifically, for authoritative IRR registries:

* It is RECOMMENDED that when creating/updating a route-set object with a src-members attribute, but without both a members and mp-members attribute, the software fills the mp-members attribute automatically with the contents of src-members, with the IRR registry prefix removed from references.

* It is RECOMMENDED that when creating/updating an as-set object with a src-members attribute, but without a members attribute, the software fills the members attribute automatically with the contents of src-members, with the IRR registry prefix removed from references.

The registry MAY return an informational message to the user about the modifications. The objects MUST NOT be modified if already submitted with any members or mp-members attribute, though the validation rules noted earlier (Section 3.1) MUST still be applied. Non-authoritative servers MUST NOT generate members or mp-members automatically.

IRR registry software MUST NOT attempt to automatically derive src-members from (mp-)members, as this cannot be done reliably.

3.3. Multiple references to the same primary key

Adding a IRR registry scope to each reference syntactically allows a new behavior: having multiple references to the same RPSL primary key. This is not permitted, and IRR registry software MUST reject this:

```
src-members: RIPE::AS-OTHER, ARIN::AS-OTHER
```

Figure 4: Invalid object fragment using multiple registry prefixes with the same RPSL primary key

The IRR registry software MUST verify that, without their registry prefix, all references from src-members are unique.

This reduces ambiguity regarding backwards compatibility with (mp-)members described earlier. If allowed, the attribute src-members: RIPE::AS-OTHER, ARIN::AS-OTHER would refer to two different sets, whereas the translation mp-members: AS-OTHER only refers to one set.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

This document removes a potential security issue where routing policy could be manipulated by maliciously creating set objects, which could be used in favor of legitimate objects.

While not a new issue, references between set objects can be circular, and software MUST detect such cases while resolving. It is RECOMMENDED to also limit the depth or size of their resolving to prevent excessive resource use.

6. Normative References

[RFC2622] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, DOI 10.17487/RFC2622, June 1999, <https://www.rfc-editor.org/rfc/rfc2622>.

[RFC2725] Villamizar, C., Alaettinoglu, C., Meyer, D., and S. Murphy, "Routing Policy System Security", RFC 2725, DOI 10.17487/RFC2725, December 1999, <https://www.rfc-editor.org/rfc/rfc2725>.

[RFC4012] Blunk, L., Damas, J., Parent, F., and A. Robachevsky, "Routing Policy Specification Language next generation (RPSLng)", RFC 4012, DOI 10.17487/RFC4012, March 2005, <https://www.rfc-editor.org/rfc/rfc4012>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Authors' Addresses

Sasha Romijn
Reliably Coded
Amsterdam
Netherlands
Email: sasha@reliablycoded.nl

James Bensley
Inter.link GmbH
Boxhagener Str. 80
10245 Berlin
Germany
Email: james@inter.link