Network Working Group
Internet-Draft
Intended status: Informational
Expires: 17 October 2026

J. Bensley
Inter.link GmbH

15 April 2026

Explicitly excluding objects from RPSL sets
draft-bensley-rpsl-exclude-members-00

Abstract

This document updates [RFC2622] and [RFC4012] by defining the excl-members attribute on as-set and route-set classes in the Routing Policy Specification Language (RPSL). The existing RPSL syntax only supports the implicit inclusion of everything contained within an as-set or route-set. The newly defined attribute allows operators to overcome this limitation of the existing syntax by enabling them to explicitly exclude specific members from that implicit inclusion.

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 https://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 17 October 2026.

Table of Contents

1. Introduction

The Routing Policy Specification Language (RPSL) [RFC2622] defines the as-set and route-set classes. These sets can reference direct members, such as an AS number or IP prefix, or additional sets, which in turn can reference further sets recursively. Server and client software can follow these references to resolve all members of a set. Once all references have been resolved, the result is a set of prefixes or AS numbers.

1.1. Existing Methods of Inclusion

The existing RPSL syntax allows members of an as-set or route-set to be specified in multiple ways:

1. [RFC2622] defines the members attribute.

    1. Section 5.1 of [RFC2622] defines that, for an as-set class, this attribute stores one or more primary keys, each referencing an aut-num or an as-set object.

    2. Sections 5.2 and 5.3 of [RFC2622] define that, for a route-set class, this attribute stores one or more primary keys, each referencing a route-set object (optionally with a range operator appended), an aut-num, or an as-set. Alternatively, the members attribute on a route-set may store an IPv4 address prefix range directly, that is, not an RPSL primary key that points to a route object. In this case, the address prefix range is used to identify matching route objects. That address prefix range may optionally have a range operator appended.

2. Section 4.2 of [RFC4012] defines the mp-members attribute for the route-set class. This attribute may store one or more primary keys, each referencing a route-set object (optionally with a range operator appended), or an IPv4/IPv6 address prefix range directly. Although not explicitly stated in [RFC4012], implementations of the mp-members attribute have generally followed the [RFC2622] members behaviour and also allow aut-num and as-set primary keys.

3. [RFC2622] defines the mbrs-by-ref and member-of attributes.

    1. Section 5.1 of [RFC2622] defines that, for an as-set, these attributes allow the inclusion of aut-nums in the as-set if and only if the criteria defined in the RFC linking both attributes together is met.

    2. Section 5.2 of [RFC2622] defines that, for a route-set, these attributes allow the inclusion of routes in the route-set if and only if the criteria defined in the RFC linking both attributes together is met.

4. Section 3 of [RFC4012] defines the router6 class along with the member-of attribute on that class and, as a result, allows the inclusion of route6 objects in a route-set if and only if the criteria relating to the mbrs-by-ref and member-of attributes defined in Section 5.2 of [RFC2622] is met.

When using the (mp-)members attribute to include an as-set or route-set (hereinafter the "included set") within an another as-set or route-set (hereinafter the "including set"), all members of the included set are included in the including set. This is not limited to the members directly nested inside the included set, but extends to all members recursively included throughout the RPSL hierarchy of the included set. This implicit recursive inclusion logic is hereinafter referred to as "greedy inclusion logic".

In the figure below, the as-set AS-EXAMPLE-1 only includes one member but, as a result of that single inclusion, AS-EXAMPLE-1 contains the aut-nums AS65001, AS65002, and AS65003:

```
as-set: AS-EXAMPLE-1
members: AS-EXAMPLE-2

as-set: AS-EXAMPLE-2
members: AS65001, AS65002, AS-EXAMPLE-3

as-set: AS-EXAMPLE-3
members: AS65003
```

Figure 1: An example of greedy inclusion logic. A three level hierarchy is recursed even though AS-EXAMPLE-1 only includes one additional as-set

The same inclusion logic applies when a route-set references another route-set or as-set in the members attribute; everything within the included set, including all recursively nested members, is implicitly included in the including set.

Similarly, greedy inclusion logic also applies to prefixes: the (mp-)members attribute of a route-set includes any route or route6 objects which match the IPv4 or IPv6 address prefix range and any optional range operator.

1.2. Existing Methods of Exclusion

The filter-set class and filter attribute are defined in Section 5.4 of [RFC2622]. The mp-filter attribute was later defined in Section 2.5.2 of [RFC4012]. Together these attributes provide a method for declaring, in the IRR ecosystem, the prefixes that a network will _not_ accept.

The (mp-)filter attribute may be used to exclude route or route6 objects that have been included by the greedy inclusion logic of the (mp-)members attribute of a route-set. In practice, this is achieved by first including all route or route6 objects which match the IPv4

or IPv6 address prefix range and any optional range operator in the (mp-)members attribute of a route-set, and then removing from that result any route or route6 objects which match the IPv4 or IPv6 address prefix range and any optional range operator in the (mp-)filter attribute of the filter-set.

For as-sets and route-sets that use the mbrs-by-ref and member-of attributes, both attributes must contain corresponding values. This reduces the greediness of the inclusion logic when compared with the (mp-)members attribute alone, because the including set _and_ the included set MUST both reference each other, whereas the (mp-)members attribute alone only requires the including set to reference the included set. It is also possible to reduce the greediness further by changing the value of the mbrs-by-ref attribute from ANY to a list of specific values, and/or by removing the as-set or route-set primary key from the member-of attribute of an aut-num or route/ route6 object.

There is currently no method to exclude either an aut-num, an as-set, or a route-set that was included by the greedy inclusion logic of the (mp-)members attribute of an as-set or route-set object. This is the missing functionality in the RPSL syntax that this document aims to address.

1.3. The Need for Additional Exclusion Control

Using the existing RPSL syntax, a local network operator who manages an as-set or route-set object is only in control of which members are directly included in their set. The RPSL syntax provides no control over the members included beyond this first level of inclusion. This can lead to problems for other networks connecting to the local network when using the set for route filtering purposes.

When using the local network’s set to build a routing filter for connecting to the local network, another network may decide to exclude certain members from the local network’s set. This decision needs to be made on the basis of context known to the other network. For example, the other network decides that IXP LANs generally MUST NOT be reachable via the public Internet and therefore filters the presence of any IXP route server sets from the local network’s set. There are various reasons why the other network operator MAY exclude members from the local network’s set; however, the context regarding which sets to exclude is based on knowledge and policy specific to the other network. These reasons are out of scope for this document.

After excluding any members that the other network does not want to include, the other network uses the remainder of the local network’s set to build a routing filter. However, when the other network is

performing member exclusions on the local network's set, it does not know based on IRR data alone, the relationship of the local network to the members of it's set. For example, nested many levels down inside the local network's set are networks for which the local network MUST NOT be a connectivity provider. This could be because there is no commercial agreement in place that includes connectivity SLAs and the included set requires such SLAs, or because the including set simply does not have the capacity to meet the traffic demands of the included set. The other network cannot determine the commercial or operational relationship between the local network and these nested entries within the local network's set based purely on the information available in the IRR DB. The local network operator needs a method to provide this context to the other network operator so that the other network's existing process for excluding set members can be extended with this additional context. This is the scope of this document: providing a set owner with the ability to signal exclusion context to consumers of the set.

1.4. Problems with the Existing Syntax

The existing greedy inclusion logic of the (mp-)members attribute of as-sets and route-sets, coupled with the inability to alter this logic, means that there is no method for a local network operator to provide context to other network operators about set member exclusions. Due to this inability to provide such context, other networks encounter various problems when using the local network's set to build routing filters. A non-exhaustive list of example undesirable outcomes is provided below.

1. The owner of an including set adds a new member, but the included set is not connected with, or related to, the including set. This allows the including set owner to make routing advertisements they are not authorised to make, either intentionally (commonly known as a route hijack) or accidentally (commonly known as a route leak). Other networks lack this context and will generate routing policy based on the unauthorised inclusion.

   This can happen anywhere in the set hierarchy; the unauthorised inclusion may be nested many levels down within the set used by the other network, making it difficult to remove the included set due to the absence of any direct relationship between the network making the inclusion and the network using the as-set or route-set to build a routing filter.

2. A member is added to a set which creates a loop when the set is resolved (set A contains set B which contains set A). This can lead to IRR-derived prefix or AS-path filters either massively expanding in size or simply becoming unresolvable.

3. A member is added to a set which is intended to contain a network's downstreams, but the included set relates to a peer or upstream, not a downstream. According to the IRR data, the operator of the including set has now become a transit provider for the operator of the included set. This can also lead to IRR-derived prefix or AS-path filters massively expanding in scope, to a point at which the filters are effectively nullified and provide no protection. This can happen because one or more included sets, when recursively resolved, expand to the entire global BGP routing table (or a significant portion thereof), or because the filters become so large that they can no longer be programmed into network devices.

4. A member is added to a set which relates to a network whose actions violate a law, a geo-political agreement, sanctions, or the connectivity terms and conditions of a peer or upstream of the including set operator. All peers and upstreams of the network owning the including set could now forward traffic for the included set towards the network using the including set. If the including set is operated by a direct peer, two peers may be able to discuss an alternative approach between themselves. If the included set is many levels down within the set hierarchy of a peer, the peer likely cannot help due to having no relationship with the including set. The operator may have to implement manual workarounds to avoid routing traffic for the included set, which may be difficult to maintain and error-prone.

5. A member is added to a set with whom a peer or upstream of the including set operator already has a direct relation. A regulatory requirement may restrict the peer or upstream from exchanging traffic with the operator of the included set via the including set operator, or via any third-party operator. Again, such a scenario may require manual workarounds.

This document updates the RPSL definitions in [RFC2622] and [RFC4012] by introducing the excl-members attribute.

This allows the operator of the including set to exclude aut-nums, as-sets, and route-sets from the included set, or to exclude the included set entirely from being included in the including set. This allows operators to have finer-grained control over the members of their sets and to avoid the undesired outcomes described above.

1.5. 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.

1.6. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

The terms for peering relationships used in this document are taken from [draft-ietf-grow-routing-ops-terms], specifically:

* Upstream
* Downstream
* Peer
* Transit provider

2. The excl-members Attribute

The excl-members attribute is defined by this document for the as-set class and route-set class.

This attribute builds on the RPSL modifications introduced by the src-members attribute defined in [draft-romijn-grow-rpsl-registry-scoped-members]. Therefore, readers should be familiar with the src-members attribute and its motivation before reading the rest of this document.

2.1. The as-set Class

The new excl-members attribute on the as-set class uses almost exactly the same syntax as the existing members attribute from Section 5.1 of [RFC2622], in that one or more RPSL primary keys of an aut-num or as-set may be specified. The only difference is that when an as-set is specified in excl-members, the as-set primary key MUST be prefixed with a registry name and a double colon, for example SOURCE::. This requirement is intended to ensure that the correct object is being excluded because of the existing ambiguity of as-set primary keys in the IRR ecosystem, as documented in

[draft-romijn-grow-rpsl-registry-scoped-members].

Attribute:
excl-members

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

Type:
optional, multi-valued

2.2. The route-set Class

The new excl-members attribute on the route-set class uses similar syntax to the existing members attribute from Sections 5.2 and 5.3 of [RFC2622], in that one or more RPSL primary keys of an aut-num, an as-set, or a route-set may be specified. What is different is that excl-members does not accept IPv4 or IPv6 address prefix ranges because they can already be filtered using a filter-set. If an as-set or route-set is specified in excl-members, the set primary key MUST be prefixed with a registry name and a double colon, for example SOURCE::. This requirement is intended to ensure that the correct object is being excluded because of the existing ambiguity of as-set and route-set primary keys in the IRR ecosystem, as documented in [draft-romijn-grow-rpsl-registry-scoped-members].

Attribute:
excl-members

Value:
list of ([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. Attribute Validation

When an authoritative IRR registry processes the creation or update of an as-set or route-set object with the excl-members attribute present, it MUST validate the contents of the attribute.

2.3.1. Registry Scoped Keys Only

All primary keys in excl-members MUST have a registry scope provided. By requiring registry-scoped primary keys in the excl-members attribute, it becomes possible to have multiple references to the same RPSL primary key after stripping the registry prefix. This is not permitted, and IRR registry software MUST reject this:

```
excl-members: RIPE::AS-EXAMPLE, ARIN::AS-EXAMPLE
```

Figure 2: 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 primary keys in excl-members are unique. If allowed, the attribute excl-members: RIPE::AS-EXAMPLE, ARIN::AS-EXAMPLE would refer to two different set objects, whereas the (mp-)members attribute can only contain one instance of AS-EXAMPLE, which creates ambiguity regarding which set the exclusion refers to when sets exist in multiple registries with the same primary key.

Similarly, the IRR software MUST NOT allow the registry scopes in the excl-members attribute and the src-members attribute to be mixed when both attributes are populated on the same set object and, after removing the registry scope, they reference the same primary key.

```
member: AS-EXAMPLE
src-members: ARIN::AS-EXAMPLE
excl-members: RIPE::AS-EXAMPLE
```

Figure 3: Invalid object fragment using different registry prefixes with the same RPSL primary key across attributes

If allowed, because of the presence of the src-members attribute, ARIN::AS-EXAMPLE would be included instead of AS-EXAMPLE (src-members takes precedence over members), and the excl-members attribute value RIPE::AS-EXAMPLE would not match the src-members value.

2.3.2. Any Primary Key and Registry Scope

The IRR software MUST NOT require that the primary key of an entry in the excl-members attribute is also a direct member of the object being created or updated. The excl-members attribute is used to exclude objects anywhere in the recursive set hierarchy, starting from the point of definition and moving downwards within the hierarchy. This is because the object to be excluded might have been included by a member of a member of a member, and so on.

When creating or updating an object with the excl-members attribute, authoritative IRR software MUST NOT require the registry scope that precedes the object primary key to be one the IRR software recognises as valid. An authoritative IRR server may have its content mirrored to resolver IRR servers, which may have visibility of many more registries.

2.4. Joint vs. Split Attributes

The excl-members attribute could have been designed as two separate attributes, that is, excl-members and excl-mp-members. This would provide finer-grained control by only excluding a primary key found in a members attribute, and only excluding a primary key found in an mp-members attribute, respectively. However, this creates additional problems:

1. If the primary key of an object that needs to be excluded exists in the set hierarchy in a members attribute, but is only excluded if found in an mp-members attribute (for example, it is only referenced in excl-mp-members), the object referenced is not successfully excluded. The problem is that the wrong exclude attribute has been used. This problem does not exist with a single combined exclude attribute.

2. The syntax of the mp-members attribute allows it to contain all types of value supported by the members attribute, plus additional value types not supported by the members attribute; syntactically, mp-members supports a superset of members. A primary-key type which is valid in both attributes, such as a route-set, could be included via the members attribute or the mp-members attribute in the set hierarchy. To ensure that this object is excluded, its primary key would need to be added to both exclude attributes. This creates the problem that the two attributes would need to store all values that are valid in the members attribute. This would render an excl-members attribute redundant because everything is duplicated in an excl-mp-members attribute. This problem does not exist with a single combined exclude attribute.

3. Split exclude attributes allow intentional evasion of an exclusion by examining the set hierarchy, observing that a primary key is excluded only if found in one of the (mp-)members attributes, and then intentionally including it in the other attribute. This problem does not exist with a combined exclusion attribute.

Due to the problems listed above, the excl-members attribute has been designed as a single attribute.

3. Exclusion Logic

3.1. The as-set Class

When the excl-members attribute is populated on an as-set object, the primary keys stored in the attribute reference aut-nums or as-sets that a resolving IRR server MUST NOT resolve when recursively resolving that object's members.

1. This exclusion applies to the members attribute of the as-set object on which the excl-members attribute is populated, and to the members attribute of all recursively resolved as-sets within that set. Because the RPSL primary keys stored in the excl-members attribute are prefixed with a registry scope, the primary keys in the members attribute MUST be checked against all keys in the excl-members attribute with the registry scope removed.

2. This exclusion applies to the src-members attribute (as defined in [draft-romijn-grow-rpsl-registry-scoped-members]) of the as-set object on which the excl-members attribute is populated, and to the src-members attribute of all recursively resolved as-sets within that set. In this case, the registry-scoped RPSL primary keys in src-members MUST match a registry-scoped key in excl-members exactly, without the registry scope being removed from either of the two keys being compared.

3. If both members and src-members are defined on an as-set object, and the same key exists in both attributes after removing the registry scope from the src-members entry, the key from src-members, which is prefixed with a registry scope, MUST be compared against all entries in excl-members with their registry scopes present. Matching keys in src-members take precedence over matching keys in members.

The figure below shows IRR data in its raw and unresolved state. In the figure, RIPE::AS-EXAMPLE-4 is not defined; it may not exist, or the resolving IRR system may not contain data from the RIPE registry:

```
as-set: AS-EXAMPLE-1
members: AS-EXAMPLE-2, AS65001
source: ARIN

as-set: AS-EXAMPLE-2
members: AS65002, AS-EXAMPLE-3
excl-members: RIPE::AS-EXAMPLE-4, AS65005, AS65002
source: RIPE

as-set: AS-EXAMPLE-3
members: AS65003, AS65005, AS-EXAMPLE-4
src-members: RIPE::AS-EXAMPLE-4
source: RIPE

as-set: AS-EXAMPLE-4
members: AS65004
source: ARIN
```

Figure 4: An example as-set hierarchy, in its unresolved state

The figure below shows the result from a resolving IRR server after the members of set AS-EXAMPLE-1 have been resolved and the logic relating to excl-members has been applied:

```
as-set: AS-EXAMPLE-1
members: AS65001, AS65003
```

Figure 5: AS-EXAMPLE-1 in its resolved state with exclusions applied

* It can be seen that excl-members took effect on the object on which it was defined, not just its descendants. This is shown by AS65002 not being included in the final result because AS65002 is both a member _and_ excl-members of AS-EXAMPLE-2.

* AS-EXAMPLE-4 is excluded even though AS-EXAMPLE-4 is defined in ARIN and is a member of AS-EXAMPLE-3. This happens because a src-members attribute is defined on AS-EXAMPLE-3, and it takes precedence over the members attribute. This means RIPE::AS-EXAMPLE-4 would be included, but it is excluded by the excl-members attribute one level higher in the tree, on AS-EXAMPLE-2.

3.2. The route-set Class

When the excl-members attribute is populated on a route-set object, the primary keys stored in the attribute reference aut-nums, as-sets, or route-sets that a resolving IRR server MUST NOT resolve when recursively resolving the members of that route-set object.

1. This exclusion applies to the (mp-)members attributes of the route-set object on which the excl-members attribute is populated, and to the (mp-)members attributes of all recursively resolved route-sets and as-sets within that route-set. Because the RPSL primary keys stored in the excl-members attribute are prefixed with a registry scope, the primary keys in the (mp-)members attributes MUST be checked against all keys in excl-members with the registry scope removed.

2. This exclusion applies to the src-members attribute (as defined in [draft-romijn-grow-rpsl-registry-scoped-members]) of the route-set object on which the excl-members attribute is populated, and to the src-members attribute of all recursively resolved route-sets and as-sets within that route-set. In this case, the registry-scoped RPSL primary keys in src-members MUST match a registry-scoped key in excl-members exactly, without the registry scope being removed from either of the two keys being compared.

3. If both (mp-)members and src-members are defined on a route-set object, and the same key exists in both attributes after removing the registry scope from the src-members entry, the key from src-members, which is prefixed with a registry scope, MUST be compared against all entries in excl-members with their registry scopes present. Matching keys in src-members take precedence over matching keys in (mp-)members.

The figure below shows IRR data in its raw and unresolved state:

```
route-set: RS-EXAMPLE-1
members: 192.0.2.0/25, RS-EXAMPLE-2
source: ARIN

route-set: RS-EXAMPLE-2
mp-members: 2001:db8::/33
mp-members: RS-EXAMPLE-3, RS-EXAMPLE-4
src-members: RIPE::RS-EXAMPLE-3, RIPE::RS-EXAMPLE-4
excl-members: RIPE::RS-EXAMPLE-4
source: RIPE

route-set: RS-EXAMPLE-3
members: 192.0.2.128/25, RS-EXAMPLE-4
source: RIPE

route-set: RS-EXAMPLE-4
members: 2001:db8:8000::/33
source: ARIN
```

Figure 6: An example route-set hierarchy, in its unresolved state

The figure below shows the result from a resolving IRR server after the members of set RS-EXAMPLE-1 have been resolved and the logic relating to excl-members has been applied:

```
route-set: RS-EXAMPLE-1
members: 192.0.2.0/25, 2001:db8::/33, 192.0.2.128/25
```

Figure 7: RS-EXAMPLE-1 in its resolved state with exclusions applied

* It can be seen that excl-members took effect on the object on which it was defined, not just its descendants. This is shown by 2001:db8:8000::/33 not being included in the final result because RS-EXAMPLE-4 is both a member _and_ excl-members of RS-EXAMPLE-2.

* Even though RS-EXAMPLE-4 is excluded by RS-EXAMPLE-2, it was also included by RS-EXAMPLE-3, but 2001:db8:8000::/33 is still excluded. This shows that the exclusion logic applies from the point in the hierarchy where it is defined, all the way down, taking precedence over any subsequent includes.

* RS-EXAMPLE-4 is excluded even though RS-EXAMPLE-4 is defined in ARIN and RIPE::RS-EXAMPLE-4 is specified in excl-members on RS-EXAMPLE-2. This is because the RS-EXAMPLE-4 entry in the (mp-)members attribute of RS-EXAMPLE-3 is ambiguous due to the absence of a src-members attribute on RS-EXAMPLE-3. This means that the excl-members value RIPE::RS-EXAMPLE-4 has to be checked against the members attribute on RS-EXAMPLE-3 with the registry scope removed, which results in a match when checking for primary keys to be excluded from recursive resolution.

3.3. Cumulative Excludes

As as-set and route-set objects are recursively resolved and excl-members attributes are encountered, the RPSL primary keys to be excluded need to be tracked by the software implementation.

At any point in the hierarchy where excl-members is encountered, all (mp-)members and src-members attributes within the same branch of the hierarchy are subject only to the excl-members that have been encountered along this branch. The cumulative nature of the excl-members attribute requires an approach which is inline with a depth-first search. However, the resolving software may use any algorithm to resolve the set hierarchy, which means that multiple lists of RPSL keys to exclude MAY have to be maintained, one for each branch of the hierarchy.

The exact software implementation details are outside the scope of this document. This section aims to demonstrate that the exclusion list is cumulative, but not as simple as a single global list.

Consider the following figure which shows as-set objects in their unresolved state:

```
as-set: AS-EXAMPLE-1
members: AS-EXAMPLE-2, AS-EXAMPLE-3
excl-members: RIPE::AS-EXAMPLE-4

as-set: AS-EXAMPLE-2
members: AS-EXAMPLE-4
excl-members: RIPE::AS-EXAMPLE-5

as-set: AS-EXAMPLE-4
members: AS65004

as-set: AS-EXAMPLE-3
members: AS-EXAMPLE-5
excl-members: AS65006

as-set: AS-EXAMPLE-5
members: AS65005
```

Figure 8: An example as-set hierarchy, in its unresolved state

The following figure shows the resolved members of as-set AS-EXAMPLE-1:

```
as-set: AS-EXAMPLE-1
members: AS65005
```

Figure 9: AS-EXAMPLE-1 in its resolved state with exclusions applied

1. The resolving process starts by resolving the members of AS-EXAMPLE-1.

2. If a depth-first search approach is taken by the IRR software and AS-EXAMPLE-2 is resolved before AS-EXAMPLE-3; AS-EXAMPLE-4 is not included due to the excl-members attribute defined on AS-EXAMPLE-1. The exclusion is applied from the point of definition onwards; the resolving process inherits the currently defined list of exclusions (RIPE::AS-EXAMPLE-4) when it moves on to resolve AS-EXAMPLE-2.

3. AS-EXAMPLE-2 defined a new excl-members attribute with the value RIPE::AS-EXAMPLE-5; however, there is nothing left to resolve in AS-EXAMPLE-2, so this exclusion has no effect.

4. Continuing the depth-first search approach, the IRR software returns to AS-EXAMPLE-1, uses the exclusion list _as it existed_ while resolving AS-EXAMPLE-1 (containing only RIPE::AS-EXAMPLE-4), and then resolves AS-EXAMPLE-3.

5. AS-EXAMPLE-3 includes AS-EXAMPLE-5. This is not excluded even though the IRR software has encountered an excl-members attribute which contains the value RIPE::AS-EXAMPLE-5. This is because that excl-members attribute was found on a different branch of the set hierarchy.

6. Continuing the resolution process, resolving AS-EXAMPLE-5 returns AS65005 only. The exclusion of AS65006 defined on AS-EXAMPLE-3 was applied to the resolution of AS-EXAMPLE-5 in addition to the exclusion of RIPE::AS-EXAMPLE-4; however, no members or src-members attributes were found on AS-EXAMPLE-5 with these values.

The example shows that encountered exclusions do not apply across branches of the hierarchy.

Cross-branch exclusions MUST NOT be allowed by the software implementation. If allowed, the operator of an included set would be able to influence the exclusions for the network owning the including set. The included set could add RPSL keys to their excl-members attribute, for a set included in a different branch of the including set.

4. Backwards Compatibility

Section 2 of [RFC2622] defines syntax which allows for finer-grained inclusion and exclusion of IP prefix ranges in the (mp-)members attribute of a route-set, that is, ^-, ^+, ^n, and ^n-m. It might be possible to introduce similar syntax to the existing (mp-)members attributes, which would match and include or exclude aut-num, as-set, and route-set primary keys. However, this would break backwards compatibility. Existing IRR implementations or tooling which parses IRR data, and which has not been updated to include such changes to the existing attributes, would break when encountering data from an implementation which has been updated to return data using these new syntax options.

The behaviour of RPSL-compliant software is to ignore unrecognised attributes. This means that adding the exclusion logic defined in this document, based on the contents of a new attribute, has no impact when existing IRR software implementations process an object containing the new attribute.

Although unrelated to the ability to retain backwards compatibility for parsing data, there is a breaking change introduced by this document in relation to the data returned from an IRR implementation that has implemented the defined changes. The data returned by two IRR resolvers, operating on the exact same database, where one resolver uses software that has not implemented the changes introduced in this document, would not be the same. Therefore, for consumers of the returned data, the presence of the excl-members attribute indicates that data may have been excluded if any matches to the excluded primary keys were found during the resolving process.

5. Security Considerations

This document adds the ability to specify that IRR-derived prefix and AS-path filter lists may exclude specific entries that are presently included by the existing greedy inclusion logic, which may otherwise create security issues.

It is possible that the operator of an including set includes the wrong primary key in the excl-members attribute. However, this is not a new issue; it has long been possible to include unintended primary keys in set objects. This document does not change this existing behaviour. It does limit the scope of unintended inclusion by requiring registry-scoped keys only in the excl-members attribute.

Great progress has been made recently in relation to BGP route filtering, specifically with the deployment of Route Origin Authorizations (ROAs) as defined in [RFC9582], the ongoing development of Autonomous System Provider Authorizations (ASPAs) as defined in [draft-ietf-sidrops-aspa-verification], and the publication of [RFC9234]. The method proposed in this document is intended to complement those existing developments, further enriching the operator toolkit, and not to work against them or be mutually exclusive.

6. References

6.1. Normative References

[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>.

[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>.

[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>.

[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>.

6.2. Informative References

[RFC9234] Azimov, A., Bogomazov, E., Bush, R., Patel, K., and K. Sriram, "Route Leak Prevention and Detection Using Roles in UPDATE and OPEN Messages", RFC 9234, DOI 10.17487/RFC9234, May 2022, <https://www.rfc-editor.org/rfc/rfc9234>.

[RFC9582] Snijders, J., Maddison, B., Lepinski, M., Kong, D., and S. Kent, "A Profile for Route Origin Authorizations (ROAs)", RFC 9582, DOI 10.17487/RFC9582, May 2024, <https://www.rfc-editor.org/rfc/rfc9582>.

[draft-romijn-grow-rpsl-registry-scoped-members] Romijn, S. and J. Bensley, "Registry scoped members for RPSL set objects", 25 August 2025, <https://datatracker.ietf.org/doc/draft-romijn-grow-rpsl-registry-scoped-members/>.

[draft-ietf-sidrops-aspa-verification] Azimov, A., Bogomazov, E., Bush, R., Patel, K., Snijders, J., and K. Sriram, "BGP AS_PATH Verification Based on Autonomous System Provider Authorization (ASPA) Objects", 19 October 2025, <https://datatracker.ietf.org/doc/draft-ietf-sidrops-aspa-verification/>.

[draft-ietf-grow-routing-ops-terms]
Fiebig, T. and W. Tremmel, "Currently Used Terminology in Global Routing Operations", 3 April 2026, <https://datatracker.ietf.org/doc/draft-ietf-grow-routing-ops-terms/>.

Author’s Address

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