Spring Working Group
Internet-Draft
Intended status: Standards Track
Expires: 20 September 2025

Z. Ali
Cisco Systems, Inc.
C. Li
Huawei Technologies
L. Jalil
Verizon
A. Bogdanov
British Telecom
F. Clad
S. Sidor
Cisco Systems, Inc.
19 March 2025

SR Policy Programming RPC
draft-ali-spring-sr-policy-programming-rpc-02

Abstract

The document specifies a Remote Procedure Call (RPC) for programming ephemeral states associated with an SR Policy.

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.

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 20 September 2025.

Table of Contents

1. Terminology

Headend node: Packet flows are steered into an SR Policy on a node where it is instantiated called a headend node [RFC9256].

SR: Segment Routing.

SID: Segment Identifier.

SRv6: Segment Routing over IPv6 data plane.

2. Introduction

Segment Routing (SR) [RFC8402] allows a node to steer a packet flow along any path. A Segment Routing Policy (SR Policy) [RFC8402] is an ordered list of segments that represent a source-routed policy. The headend node is said to steer a flow into an SR Policy. The packets steered into an SR Policy have an ordered list of segments associated with that SR Policy written into them. Segment Routing Policy Architecture [RFC9256] details the concepts of SR Policy and steering into an SR Policy.

There are use cases where a controller needs to install states associated with an SR policy in real-time. These use cases require programming ephemeral states at the SRTE headend node. Tactical traffic engineering, where an SR Policy is installed to divert traffic from a congested IGP shortest path for a short time, is an example of such an application.

A controller may configure ephemeral states associated with an SR Policy using PCEP [RFC5440] and BGP-TE [I.D-draft-ietf-idr-sr-policy-safi]. However, using the PCEP or BGP-TE protocol requires the controller to implement the PCEP or BGP-TE protocol stacks, which require laborious encoding, decoding, and serialization of data streams.

A controller may configure states associated with an SR Policy using Netconf. For this purpose, the controller uses the Yang model defined in [I.D-draft-ietf-spring-sr-policy-yang]. However, installing a configuration requires a configuration lock at the router. An SR Policy configuration may be serialized behind other configurations being applied at the headend node. That makes the configuration route slow and unpredictable for real-time applications like tactical traffic engineering.

The draft specifies a reference model that can be used to create, update, delete, and report ephemeral states associated with an SR Policy and report SR policy (reporting is to be added in a future revision). In this fashion, the API is self-contained for both SR policy programming and reporting. The aim is to ease the programming of SR Policies from an SDN controller and leverage benefits from the RPC framework like gRPC [GRPC].

3. RPC Data Model Overview

The RPC data model is based the SR policy data model defined in SR Policy Architecture [RFC9256]:

* An SR Policy is identified by the <Headend, Color, Endpoint> tuple as per section 2.1 of [RFC9256]. An SR Policy is associated with one or more candidate paths.

* A candidate path is identified in the context of a single SR Policy. The tuple <Protocol-Origin, Originator, Discriminator> uniquely identifies a candidate path within that context. The definition of Protocol-Origin, Originator, Discriminator fields are based on [RFC9256]. The draft introduces RPC as one of the Protocol-Origin of a candidate path described in Section 2.3 of [RFC9256]. The Preference of the candidate path is defined as per Section 2.7 of [RFC9256]. A candidate path is expressed as a segment list or a set of segment lists.

* A segment list represents a specific source-routed path to send traffic from the headend to the endpoint of the corresponding SR Policy. The segment list itself can be specified using different segment-descriptor types defined in [RFC9256]. If a candidate path is associated with a set of segment lists, each segment list is associated with weight for weighted load balancing.

* The Binding SID (BSID) and other fields are based on [RFC9256].

The draft does not modify any procedures or data model defined in [RFC9256].

4. RPC Semantics Overview

The SR Policy Service RPC is used to create/update/delete/ report an SR-TE policy based on the specified parameters (reporting is to be added in a future revision of the document).

The server of the RPC is the SR policy headend node. The client of the RPC is an entity that wants to program an SR Policy at the head-end, e.g., an SDN controller.

The RPC attributes are exchanged using the SR Policy Open Message. A set of policies to be acted on is communicated in the SR Policy Request Message.

The unit of signaling is the SR policy. Specifically, the SR Policy Request Message contains all candidate-paths. Candidate-paths not included in the update request are removed by the server. Policy attributes (e.g., Binding SID), that are not specified for each candidate-path are applied to all candidate-paths.

The server supports two modes of operations - transactional and persistent:

* Transactional Mode: The server preserves all policies/CPs instantiated by this client, event after the client disconnects. The benefit of this mode of operation is that the client does not need to maintain a persistent connection with all the headend nodes where it has instantiated SR Policies. It is the clients responsibility to delete the Policies when they are no longer needed.

* Persistent Mode: The client and server maintains a persistent RPC session, i.e., controller continues to keep the RPC session established with SR policy head-end node. The server marks the SR policies as stale on RPC session disconnect and deletes the SR polices if the client does not reconnect and replay the SR Policies within a specified time. The benefit of this mode is that there is synchronization between the SR Policy state by the headend with the controller and this reduces the chances of their being stale state on the headend.

5. RPC Messages Overview

The service uses the following messages to programming ephemeral states associated with an SR Policy.

5.1. OPEN Message

The OPEN Message exchanges the RPC attributes. The following attributes are exchanged using the OPEN Message:

* RPC Mode of Operation: Transactional or Persistent, as described in previous section.

* Clean-up Timer: The clean-up timer attribute is only applicable to the persistent mode of the RPC. The server deletes the SR polices if the client does not reconnect and replay the SR Policies within the time interval specified by the clean-up timer value.

* Client Identifier: The server tracks policy ownership by RPC client by a client identifier. The primary reason, being the ability to restrict a given SR Policy to be managed by only 1 client.

5.2. POLICY_REQUESTS Message

A set of policies to be acted on is communicated in the POLICY REQUESTS Message

Each policy request is uniquely identified by a sequence ID and a policy key:

* The policy key is < Headend, Color, Endpoint > tuple [RFC9256].

* The sequence number serves the purpose of allowing the client to identify which request the server is responding to when multiple requests for the same policy are sent.

An operation is specified for each policy request. The following policy operations are supported:

* CREATE/ UPDATE: The server creates and installs the specified policies. Policies that already exist are updated by the service.

* DELETE: The server delates the specified policies.

5.3. START_OF_REPLAY Message

The START_OF_REPLAY message is applicable when client and server maintains a persistent RPC session (the persistence mode). The client is expected to send a START OF REPLAY message after re-connect. The client is expected to replay programming the SR policies previously programmed at the server.

5.4. END_OF_REPLAY Message

The END_OF_REPLAY message is applicable when client and server maintains a persistent RPC session (the persistence mode). The client is expected to send an End of Replay message after replaying policies after re-connect. It indicates that client has finished replaying all policies, which will then trigger cleanup of any policies not reclaimed by the client.

6. RPC

6.1. INPUT Parameters

Set of the SR Policy Messages:

* OPEN Message.

* POLICY_REQUEST Message.

* START_OF_REPLAY Message.

* END_OF_REPLAY Message.

The OPEN Message, START_OF_REPLAY and END_OF_REPLAY contents are as summarized in the previous section.

The POLICY_REQUESTS Message supports the following operations:

* CREATE_UPDATE: Create operation for specified Policy. If Policy already exist, it will be updated.

* DELETE: Delete operation for specified Policy. Deletion of non-existent policy will be responded with specific error code.

The POLICY_REQUESTS Message contains a set of policies to be acted, where each policy is identified by the following attributes:

* POLICY_KEY: The policy key is < Headend, Color, Endpoint > tuple [RFC9256]:

  - Headend: IPv4 or IPv6 Router-ID of the headend node.

  - Color: Color is an unsigned non-zero 32-bit integer value.

  - Endpoint: IPv4 or IPv6 address of the policy endpoint.

* SEQUENCE_NUMBER: ID to associate the request with. The sequence number serves the purpose of allowing the controller to identify which request the network element is responding to when multiple requests for the same policy are sent. The sequence number is incremented by the controller and should be scoped per policy, meaning there is an independent sequence number for each policy. Server includes the sequence number of the request it is responding to, in its reply. The server is not enforcing/ validating or re-ordering requests based on sequence number specified

* SR_POLICY_ATTRIBUTES: Policy attributes consist of the following fields:

  - TRANSIT_ELIGIBLE: Is Policy eligible for Transit as defined in [I-D.draft-ietf-idr-bgp-ls-sr-policy].

  - SR_DATAPLANE: Segment Routing Data Plane consists of the following options: SR-MPLS Data Plane or SRv6 Data Plane.

  - SR_MPLS_BINDING_SID: MPLS Binding SID defined as per section 6.1 of [RFC9256]. If not specified, network element is expected to generate dynamic BSID based on policy data plane. It consists of following fields: LABEL

- SR_SRV6_BINDING_SIDs: SRv6 Binding SIDs attributes for the policy. SR_SRV6_BINDING_SID can be EXPLICIT (explicitly specified) or DYNAMIC (dynamically allocated). Each SR_SRV6_BINDING_SID consists of following fields:

  - o EXPLICIT

    - + SRv6_SID_INFO: Used to specify explicit BSID allocation of the policy. It consists of:

      - * SID_ADDRESS: SID value.

      - * BEHAVIOR: Behavior to be associated for the BSID.

      - * SID_STRUCTURE.

      - * SRV6_BINDING_SID: Used for dynamic allocation of the BSID by network element. It consists of following fields: behavior (BEHAVIOR to be associated for the dynamic BSID).

  - o DYNAMIC

    - + BEHAVIOR : Used for dynamic allocation of the BSID by network element. It consists of following fields: behavior (BEHAVIOR to be associated for the dynamic BSID).

- PROFILE_ID: ID of the profile with which policy can be associated with a non-zero value. The Profile ID concept is described as Policy Association Group in RFC 9005. 0 value means unset. A profile represents a set of configuration knobs specifying policy or policy candidate-path (CP) attributes. The profile-ID feature allows usage of vendor/implementation specific functionality per policy without requiring explicit support by the controller.

- SR_CANDIDATE_PATHS: A set of Segment Routing Candidate Path in the context of an SR Policy, as defined in section 2.2 of [RFC9256]. It consists of following fields:

  - o SR_CANDIDATE_PATH_KEY: Identifier of an SR Candidate Path in the context of an SR Policy, as defined in section 2.6 of [RFC9256]. Candidate Path Key consists of following fields:

  + ORIGINATOR : Originator of the candidate path, as defined in [section 2.4 of [RFC9256]. It consists of ASN (Autonomous System Number) and NODE_ID (originator Node Address)

  + DISCRIMINATOR: DISCRIMINATOR of the candidate path, as defined in section 2.5 of [RFC9256D].

- NAME: Candidate path name. String of printable ASCII with max length of 256 characters. If unset, candidate-path name will be generated by the network element. Name with invalid length or unacceptable characters will be rejected

- PREFERENCE: The Preference of the candidate path is used to select the best candidate path for an SR Policy. If not specified, 100 is used as a default.

- SR_EXPLICIT_CP: Candidate path with explicitly defined a set of segment-lists. Conforms with [RPC9256]. It consists of following fields:

  + SR_SEGMENT_LISTS: A set of weighted segment lists in the explicit candidate path. A SR_SEGMENT_LIST consists of following fields: SR_SEGMENTS (Ordered list of segments), WEIGHT (Load balancing weight factor), PATH_COST (Total path cost, i.e., the accumulated metric values along the path).

  + METRIC_TYPE: Optimization metric type used for accumulating metric value (specified for each segment-list). Values are defined in BGP-LS SR Policy Metric Type registry under Border Gateway Protocol - Link State (BGP-LS) Parameters.

6.2. OUTPUT Parameters

Set of the POLICY_RES Messages, each representing response to the POLICY_REQUEST Message.

The POLICY_RES indicates result of the operation. In case of multiple requests received for policy identified by specific POLICY_KEY, always at least single response is sent. This is to confirm processing of last request.

The POLICY_RES consists of the following attributes:

* SEQUENCE_NUMBER: The SEQUENCE_NUMBER copies from corresponding the POLICY_REQUEST Message.

* POLICY_KEY: The SR Policy identifier (POLICY_KEY) copied from the corresponding POLICY_REQUEST Message.

* STATUS: Indicates the result of the operation (SUCCESS or FAILURE). This is not operational state of policy.

7. Security Considerations

TBA

8. IANA Considerations

TBA

9. References

9.1. 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/info/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/info/rfc8174>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

9.2. Informative References

[GRPC] The gRPC authors, "gRPC", March 2018.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

Appendix A. Acknowledgements

The authors would like to thank Ketan Talaulikar, Rajesh M Venkateswaran, Adithya Reddy Sesani for the contribution and the review comments.

Authors' Addresses

Zafar Ali
Cisco Systems, Inc.
Email: zali@cisco.com

Cheng Li
Huawei Technologies
Email: c.l@huawei.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

Alex Bogdanov
British Telecom
Email: alex.bogdanov@bt.com

Francois Clad
Cisco Systems, Inc.
Email: fclad@cisco.com

Samuel Sidor
Cisco Systems, Inc.
Email: ssidor@cisco.com