ZebOS-XP®
Network Platform

Version 1.4
Extended Performance

Segment Routing
Developer Guide

December 2015

IP Infusion Inc. Proprietary

© 2015 IP Infusion Inc. All Rights Reserved.

This documentation is subject to change without notice. The software described in this document and this documentation are furnished under a license agreement or nondisclosure agreement. The software and documentation may be used or copied only in accordance with the terms of the applicable agreement. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or any means electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's internal use without the written permission of IP Infusion Inc.

IP Infusion Inc.
3965 Freedom Circle, Suite 200
Santa Clara, CA 95054
+1 408-400-1900
http://www.ipinfusion.com/

For support, questions, or comments via E-mail, contact:
support@ipinfusion.com

Trademarks:
IP Infusion, OcNOS, VirNOS, ZebM, ZebOS, and ZebOS-XP are trademarks or registered trademarks of IP Infusion. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Contents

© 2015 IP Infusion Inc. Proprietary

© 2015 IP Infusion Inc. Proprietary

Preface

This guide describes the ZebOS-XP application programming interface (API) for segment routing.

Audience

This guide is intended for developers who write code to customize and extend segment routing.

Conventions

Table P-1 shows the conventions used in this guide.

Table P-1: Conventions

Convention	Description
Italics	Emphasized terms; titles of books
Note:	Special instructions, suggestions, or warnings
`monospaced type`	Code elements such as commands, functions, parameters, files, and directories

Contents

This document contains these chapters:

- Chapter 1, *Introduction*
- Chapter 2, *Segment Routing Command API*

Related Documents

The following guides are related to this document:

- *Segment Routing Command Reference*
- *Segment Routing Configuration Guide*
- *Network Services Module Developer Guide*
- *Network Services Module Command Reference*
- *Installation Guide*
- *Architecture Guide*

Note: All ZebOS-XP technical manuals are available to licensed customers at http://www.ipinfusion.com/support/document_list.

Support

For support-related questions, contact support@ipinfusion.com.

Comments

If you have comments, or need to report a problem with the content, contact techpubs@ipinfusion.com.

© 2015 IP Infusion Inc. Proprietary

CHAPTER 1 Introduction

Source routing is a technique where the sender of a packet can partially or completely specify the route that a packet should take through the network. Segment routing is a form of source routing where nodes and links are represented as segments. The path that a particular packet needs to traverse is represented by one or more segments. The list of segments is inserted into the packet itself and each segment in the path represents a particular node or an adjacency through which the packet needs to pass. The ZebOS-XP implementation of segment routing is based on draft-ietf-spring-segment-routing-02.

ZebOS-XP uses prefix segments which forward a packet along the shortest path to reach the prefix. Prefix segments are global and all the nodes in the segment routing domain advertise the forwarding entry for the prefix segment. When a prefix is for a loopback interface that identifies a node, it is called a node segment.

Segment routing uses the MPLS data plane for forwarding. Segments are identified by an MPLS label. An ordered list of segments is encoded as a stack of labels. The segment to process is on the top of the stack. Upon completion of a segment, the related label is popped from the stack.

Segment routing does not require any additional control plane protocol and is implemented by extending an existing interior gateway protocol such as OSPF or ISIS. Segment routing replaces MPLS control plane protocols such as LDP or RSVP.

In ZebOS-XP, MPLS clients such as LDP and RSVP create FEC-to-NHLFE and Incoming Label Map (FTN/ILM) entries by signaling within the MPLS domain. After this, the entries are installed into the MPLS RIB hosted by NSM.

The segment routing framework reuses the existing MPLS framework with OSPF acting as an MPLS client. OSPF with segment routing extensions exchanges the segment information within the segment routing domain. These segments are converted to MPLS FTN/ILM entries using a library. After this, the entries are installed into the same MPLS RIB hosted by NSM.

In segment routing, the path states are maintained only at the ingress node and the path to follow is pushed into the packet itself. The transit and egress nodes do not maintain state for each path traversing through them. The configuration overhead is less than traditional MPLS.

Segment Routing Global Block

The Segment Routing Global Block (SRGB) is a local property of a segment routing node. In the context of MPLS, it is a set of “local labels” for global segments. ZebOS-XP uses the same local label range for all the segment routing nodes for SRGB.

Segment Identifiers

Segments are identified by a Segment Identifier (SID) which is an unsigned 32-bit integer. Because the MPLS data plane is used, the segments are identified by a 20-bit integer, leaving the 12 left-most bits of the SID unused. A SID has an absolute value (label) allocated for the segment. Because the SRGB is the same across the entire domain, all nodes identify the segment with the same absolute value.

Forwarding Example

Figure 1-1 shows an example of segment routing where a packet is forwarded based on the shortest path from node A to node D.

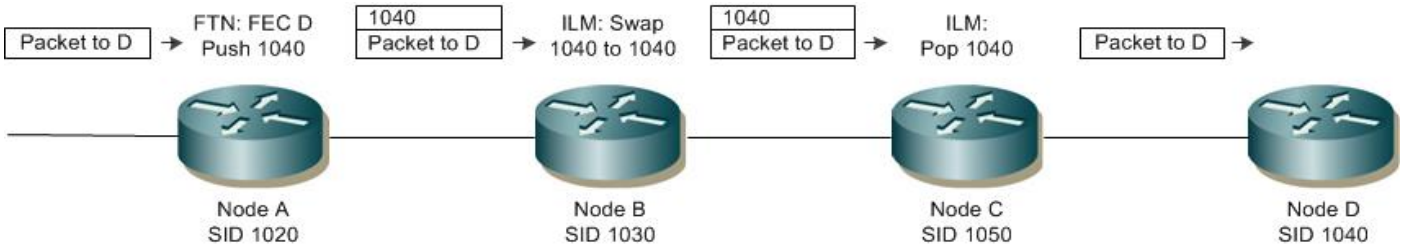

Figure 1-1: Segment routing example

Node D is identified as node segment 1040 which is advertised to the entire network. All the nodes install a MPLS forwarding entry (FTN/ILM) for node segment 1040. The next hop is based on the OSPF route. After the packet enters the segment routing domain:

1. Node A pushes label 1040 on the packet.
2. Node B swaps the same label 1040.
3. Node C does a ILM pop as it performs penultimate hop popping (PHP) for label 1040 which identifies the next node (D).
4. The packet arrives at node D.

© 2015 IP Infusion Inc. Proprietary

CHAPTER 2 Segment Routing Command API

The chapter describes the functions for segment routing.

These functions are called by the commands in the *Segment Routing Command Reference Guide*.

Function	Description
nsm_sr_api_add_prefix_sid	Adds a prefix SID to the primary address of an interface
nsm_sr_api_delete_prefix_sid	Deletes a prefix SID from the primary address of an interface
nsm_sr_api_disable	Disables SR globally on the device
nsm_sr_api_enable	Enables SR globally on the device
sr_static_edge_node_set	Creates static entries in the FTN and ILM tables in the MPLS data plane
sr_static_edge_node_unset	Deletes static entries in the FTN and ILM tables in the MPLS data plane
sr_static_route_set_ipv4	Creates a static segment routing entry
sr_static_route_unset_ipv4	Deletes a static segment routing entry

nsm_sr_api_add_prefix_sid

This function adds a prefix segment identifier (SID) to the primary address of an interface. A SID corresponds to an MPLS label.

The `prefix-sid value` command calls this function.

Syntax

```
int
nsm_sr_api_add_prefix_sid (u_int32_t vr_id, char *ifname, u_int32_t sid_value)
```

Input Parameters

`vr_id`	Virtual router identifier <0-255>
`ifname`	Interface name
`sid_value`	Segment identifier

Output Parameters

None

Return Values

NSM_SUCCESS when the function succeeds

NSM_API_SET_ERR_VR_NOT_EXIST when `vr_id` does not exist

NSM_ERR_INTERNAL when there is an internal error

NSM_SR_ERR_SR_NOT_ENABLED when segment routing is not enabled

NSM_SR_ERR_INVALID_INTERFACE when `ifname` does not exist or is layer 2 only

NSM_SR_ERR_PREFIX_ALREADY_CONFIGURED when `sid_value` is already set for `ifname`

NSM_SR_ERR_LABEL_NOT_AVAILABLE when `sid_value` cannot be reserved

NSM_SR_ERR_NO_MATCHING_IP_ADDRESS when the IP address of the interface is a secondary address or is the loopback address 127.0.0.1

nsm_sr_api_delete_prefix_sid

This function deletes a prefix segment identifier (SID) from the primary address of an interface. A SID corresponds to an MPLS label.

The `no prefix-sid value` command calls this function.

Syntax

```
int
nsm_sr_api_delete_prefix_sid (u_int32_t vr_id, char *ifname,
                              u_int32_t sid_value)
```

Input Parameters

`vr_id`	Virtual router identifier <0-255>
`ifname`	Interface name
`sid_value`	Segment identifier

Output Parameters

None

Return Values

NSM_SUCCESS when the function succeeds

NSM_API_SET_ERR_VR_NOT_EXIST when `vr_id` does not exist

NSM_ERR_INTERNAL when there is an internal error

NSM_SR_ERR_INVALID_INTERFACE when `ifname` does not exist

NSM_SR_ERR_COMMAND_NOT_CONFIGURED when an interface configured with `sid_value` cannot be found

nsm_sr_api_disable

This function disables segment routing (SR) globally on the device, de-initializing the Segment Routing Global Block (SRGB) and notifying SR-enabled protocols.

The `no segment-routing mpls` command calls this function.

Syntax

```
int
nsm_sr_api_disable (u_int32_t vr_id,
                    u_int8_t *fwd_proto,
```

 © 2015 IP Infusion Inc. Proprietary

```
                u_int8_t *igp_proto)
```

Input Parameters

`vr_id`	Virtual router identifier <0-255>
`fwd_proto`	Protocol used in forwarding plane for SR (“MPLS”)
`igp_proto`	IGP protocol used for segment signalling (“ospf”)

Output Parameters

None

Return Values

NSM_SUCCESS when the function succeeds

NSM_API_SET_ERR_VR_NOT_EXIST when `vr_id` does not exist

NSM_ERR_INTERNAL when there is an internal error

NSM_SR_ERR_COMMAND_NOT_CONFIGURED when SR is not enabled

nsm_sr_api_enable

This function enables segment routing (SR) globally on the device, initializing the Segment Routing Global Block (SRGB) and notifying SR-enabled protocols.

The `segment-routing mpls` command calls this function.

Syntax

```
int
nsm_sr_api_enable (u_int32_t vr_id,
                   u_int8_t *fwd_proto,
                   u_int8_t *igp_proto)
```

Input Parameters

`vr_id`	Virtual router identifier <0-255>
`fwd_proto`	Protocol used in forwarding plane for SR (“MPLS”)
`igp_proto`	IGP protocol used for segment signalling (“ospf”)

Output Parameters

None

Return Values

NSM_SUCCESS when the function succeeds

NSM_API_SET_ERR_VR_NOT_EXIST when `vr_id` does not exist

NSM_ERR_INTERNAL when there is an internal error

NSM_SR_ERR_ALREADY_ENABLED when SR is already enabled

NSM_SR_ERR_SRGB when the SRGB cannot be initialized

sr_static_edge_node_set

This function creates static entries in the FTN and ILM tables in the MPLS data plane.

Note: After this function executes, the segment node is considered an edge node (first/last node of a segment routing capability cloud).

The `segment-routing-static edge-node` command calls this function.

Syntax

```
int
sr_static_edge_node_set (u_int32_t vr_id,
                         char *vrf_name)
```

Input Parameters

`vr_id`	Virtual router identifier <0-255>
`vrf_name`	Virtual routing/forwarding name

Output Parameters

None

Return Values

RIB_API_SET_SUCCESS when the function succeeds

RIB_API_SET_ERROR when the RIB cannot be found or replacing the static route fails

RIB_API_SET_ERR_VRF_NOT_EXIST when `vrf_name` does not exist

RIB_API_SET_ERR_SR_DISABLED when segment routing is not enabled

RIB_API_SET_ERR_COMMAND_ALREADY_CONFIGURED when this node is already an edge node

RIB_API_SET_ERR_CONFIG_INVALID when the configuration is invalid

sr_static_edge_node_unset

This function deletes static entries in the FTN and ILM tables in the MPLS data plane.

The `no segment-routing-static edge-node` command calls this function.

Syntax

```
int
sr_static_edge_node_unset (u_int32_t vr_id,
                           char *vrf_name)
```

Input Parameters

`vr_id`	Virtual router identifier <0-255>
`vrf_name`	Virtual routing/forwarding name

Output Parameters

None

© 2015 IP Infusion Inc. Proprietary

Return Values

RIB_API_SET_SUCCESS when the function succeeds

RIB_API_SET_ERROR when the RIB cannot be found

RIB_API_SET_ERR_VRF_NOT_EXIST when `vrf_name` does not exist

RIB_API_SET_ERR_SR_DISABLED when segment routing is not enabled

RIB_API_SET_ERR_COMMAND_NOT_CONFIGURED when this node is not an edge node

RIB_API_SET_ERR_CONFIG_INVALID when the configuration is invalid

sr_static_route_set_ipv4

This function creates a static segment routing entry.

The `segment-routing-static` command calls this function.

Syntax

```
int
sr_static_route_set_ipv4 (u_int32_t vr_id,
                          char *vrf_name,
                          char *prefix_str,
                          char *nh_addr,
                          char *oifname,
                          u_int32_t sid_value,
                          bool_t nh_route)
```

Input Parameters

`vr_id`	Virtual router identifier <0-255>
`vrf_name`	Virtual routing/forwarding name
`prefix_str`	Prefix or FEC with mask
`nh_addr`	Next hop IPv4 address
`oifname`	Outgoing interface name
`sid_value`	Segment identifier (SID) for `prefix_str` <1024000-1048319>
`nh_route`	`PAL_TRUE` if the prefix belongs to the next hop; `PAL_FALSE` otherwise

Output Parameters

None

Return Values

SR_SUCCESS when the function succeeds

RIB_API_SET_ERROR when the RIB cannot be found

RIB_API_SET_ERR_VRF_NOT_EXIST when `vrf_name` does not exist

RIB_API_SET_ERR_SR_DISABLED when segment routing is not enabled

RIB_API_SET_ERR_NO_SUCH_INTERFACE when `oifname` does not exist

RIB_API_SET_ERR_WRONG_INTERFACE when `oifname` is layer 2 only

RIB_API_SET_ERR_VRF_NOT_BOUND when `vrf_name` is not associated with `oifname`

RIB_API_SET_ERR_VRF_BOUND when `vrf_name` is null, but `oifname` is associated with a virtual routing/forwarding name

RIB_API_SET_ERR_MALFORMED_ADDRESS when `prefix_str` is not a valid IPv4 address

RIB_API_SET_ERR_INCONSISTENT_ADDRESS_MASK when the mask portion of `prefix_str` is not valid

RIB_API_SET_ERR_MALFORMED_GATEWAY when `nh_addr` is not a valid IPv4 address

RIB_API_SET_ERR_INVALID_SID when `sid_value` is not valid

RIB_API_SET_ERROR when there was an error adding the route to the RIB

RIB_API_SET_ERR_SID_ALREADY_MAPPED when `sid_value` is already used for a static segment routing entry

RIB_API_SET_ERR_FAILED when installing FTN/ILM entries fails

sr_static_route_unset_ipv4

This function deletes a static segment routing entry.

The `no segment-routing-static` command calls this function.

Syntax

```
int
sr_static_route_unset_ipv4 (u_int32_t vr_id,
                            char *vrf_name,
                            char *prefix_str,
                            char *nh_addr,
                            char *oifname,
                            u_int32_t sid_value,
                            bool_t nh_route)
```

Input Parameters

`vr_id`	Virtual router identifier <0-255>
`vrf_name`	Virtual routing/forwarding name
`prefix_str`	Prefix or FEC with mask
`nh_addr`	Next hop IPv4 address
`oifname`	Outgoing interface name
`sid_value`	Segment identifier (SID) for `prefix_str` <1024000-1048319>
`nh_route`	`PAL_TRUE` if the prefix belongs to the next hop; `PAL_FALSE` otherwise

Output Parameters

None

Return Values

SR_SUCCESS when the function succeeds

RIB_API_SET_ERROR when the RIB cannot be found or deleting the static route

RIB_API_SET_ERR_VRF_NOT_EXIST when `vrf_name` does not exist

RIB_API_SET_ERR_SR_DISABLED when segment routing is not enabled

© 2015 IP Infusion Inc. Proprietary

RIB_API_SET_ERR_NO_SUCH_INTERFACE when `oifname` does not exist

RIB_API_SET_ERR_WRONG_INTERFACE when `oifname` is layer 2 only

RIB_API_SET_ERR_VRF_NOT_BOUND when `vrf_name` is not associated with `oifname`

RIB_API_SET_ERR_VRF_BOUND when `vrf_name` is null, but `oifname` is associated with a virtual routing/forwarding name

RIB_API_SET_ERR_MALFORMED_ADDRESS when `prefix_str` is not a valid IPv4 address

RIB_API_SET_ERR_INCONSISTENT_ADDRESS_MASK when the mask portion of `prefix_str` is not valid

RIB_API_SET_ERR_MALFORMED_GATEWAY when `nh_addr` is not a valid IPv4 address

RIB_API_SET_ERR_NO_MATCHING_ROUTE when `prefix_str`, `nh_addr`, and `sid_value` do not point to a static segment route

© 2015 IP Infusion Inc. Proprietary

© 2015 IP Infusion Inc. Proprietary

Index

© 2015 IP Infusion Inc. Proprietary

© 2015 IP Infusion Inc. Proprietary