

# Routing

This chapter provides information on configuring an enhanced, or extended, service. The product administration guides provide examples and procedures for configuring basic services on the system. You should select the configuration example that best meets your service model, and configure the required elements for that model before using the procedures described below.

This chapter includes the following sections:


- Routing Policies, on page 1
- Static Routing, on page 3
- OSPF Routing, on page 4
- OSPFv3 Routing, on page 7
- Equal Cost Multiple Path (ECMP), on page 8
- BGP-4 Routing, on page 8
- Bidirectional Forwarding Detection, on page 18
- Viewing Routing Information, on page 27


# Routing Policies

This section describes how to configure the elements needed to define routing policies. Routing policies modify and redirect routes to and from the system to satisfy specific network deployment requirements.

Use the following building blocks to configure routing policies:

- **Route Access Lists** – The basic building block of a routing policy. Route access lists filter routes based on a range of IP addresses.
- **IP Prefix Lists** – A more advanced element of a routing policy. An IP Prefix list filters routes based on IP prefixes.
- **AS Path Access Lists** – A basic building block used for Border Gateway Protocol (BGP) routing. These lists filter Autonomous System (AS) paths.
- **Route Maps** – Route-maps provide detailed control over routes during route selection or route advertisement by a routing protocol, and in route redistribution between routing protocols. For this level of control you use IP Prefix Lists, Route Access Lists and AS Path Access Lists to specify IP addresses, address ranges, and Autonomous System paths.

# Creating IP Prefix Lists

Use the following configuration example to create IP Prefix Lists:

```
config
   context context_name
      ip prefix-list name list_name { deny | permit } network_address/net_mask
```

Notes:

- Set the IP prefix list to deny, permit or match any prefix.
- IPv4 dotted-decimal and IPv6 colon-separated-hexadecimal addresses are supported.
- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# Creating Route Access Lists

Use the following procedure to create a Route Access List:

```
config
   context context_name
     route-access-list { extended identifier } { deny | permit } [ ip
address ip_address ]
     route-access-list named list_name { deny | permit } { ip_address/mask |
any } [ exact-match ]
route-access-list
standard identifier { permit | deny ) { ip_address
wildcard_mask | any |network_address }
```

Notes:

- A maximum of 64 access lists are supported per context.
- A maximum of 16 entries can defined for each route-access-list.
- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# Creating AS Path Access Lists

Use the following procedure to create an AS Path Access List:

```
config
   context context_name
     ip as-path access-list list_name [ { deny | permit } reg_expr ]
```

Notes:

- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# Creating Route Maps

Use the following configuration example to create a Route Map:

```
config
   context context_name
      route-map map_name { deny | permit } seq_number
```

Notes:

- Use the **match** and **set** commands in Route Map Configuration mode to configure the route map. Refer to the *Command Line Interface Reference* for more information on these commands.
- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

## Sample Configuration

The example below shows a configuration that creates two route access lists, applies them to a route map, and uses that route map for a BGP router neighbor.

```
config
   context isp1
      route-access-list named RACLin1a permit 88.151.1.0/30
      route-access-list named RACLin1b permit 88.151.1.4/30
      route-access-list named RACLany permit any
      route-map RMnet1 deny 100
         match ip address route-access-list RACLin1a
         #exit
         route-map RMnet1 deny 200
         match ip address route-access-list RACLin1b
         #exit
      route-map RMnet1 permit 1000
         match ip address route-access-list RACLany
         #exit
      router bgp 1
         neighbor 152.20.1.99 as-path 101
         neighbor 152.20.1.99 route-map RMnet1 in
```

# Static Routing

The system supports static network route configuration on a per context basis. Define network routes by specifying the:

- IP address and mask for the route
- Name of the interface in the current context that the route must use
- Next hop IP address

**Important** On the , static routes with IPv6 prefix lengths less than /12 and between the range of /64 and /128 are not supported.

# Adding Static Routes to a Context

To add static routes to a context configuration, you must know the names of the interfaces that are configured in the current context. Use the **show ip interface** command to list the interfaces in the current context (Exec mode).

Information for all interfaces configured in the current context is displayed as shown in the following example.

```
[local]host_name# show ip interface
Intf Name: Egress 1
Description:
IP State: Up (Bound to slot/port untagged ifIndex 402718721)
IP Address: 192.168.231.5
Subnet Mask: 255.255.255.0
Bcast Address: 192.168.231.255
MTU: 1500
Resoln Type: ARP          ARP timeout: 3600 secs
L3 monitor LC-port switchover: Disabled
Number of Secondary Addresses: 0
Total interface count: 1
```

The first line of information for each interface lists the interface name for the current context as shown in the example output. In this example, there is one interface with the name *Egress 1*.

```
config
   context context_name
      ip route { ip_address [ ip_mask ] | ip_addr_mask_combo } { next-hop
next_hop_address | egress_name [ precedence precedence [ cost cost ]
```

Notes:

- You can configure a maximum of 1,200 static routes per context. Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# Deleting Static Routes From a Context

Use the following configuration example to remove static routes from a context's configuration:

```
config
   context context_name
       no ip route { ip_address ip_mask | ip_addr_mask_combo } next_hop_address
egress_name [ precedence precedence ] [ cost cost ]
```

Notes

- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# OSPF Routing

This section gives an overview of Open Shortest Path First (OSPF) routing and its implementation in the system. It also describes how to enable the base OSPF functionality and lists the commands that are available for more complex configurations.

You must purchase and install a license key before you can use this feature. Contact your Cisco account representative for more information on licenses.

**Important** During system task recovery, it is possible for a dynamically-learned forwarding entry to incorrectly remain in the system forwarding table if that forwarding entry has been removed from the dynamic routing protocol during the recovery.

**Important** On the , OSPF routes with IPv6 prefix lengths less than /12 and between the range of /64 and /128 are not supported.

# OSPF Version 2 Overview

OSPF is a link-state routing protocol that employs an interior gateway protocol (IGP) to route IP packets using the shortest path first based solely on the destination IP address in the IP packet header. OSPF routed IP packets are not encapsulated in any additional protocol headers as they transit the network.

An Autonomous System (AS), or Domain, is defined as a group of networks within a common routing infrastructure.

OSPF is a dynamic routing protocol that quickly detects topological changes in the AS (such as router interface failures) and calculates new loop-free routes after a period of convergence. This period of convergence is short and involves a minimum of routing traffic.

In a link-state routing protocol, each router maintains a database, referred to as the link-state database, that describes the Autonomous System's topology. Each participating router has an identical database. Each entry in this database is a particular router's local state (for example, the router's usable interfaces and reachable neighbors). The router distributes its local state throughout the AS by flooding.

All routers run the same algorithm in parallel. From the link-state database, each router constructs a tree of shortest paths with itself as root to each destination in the AS. Externally derived routing information appears on the tree as leaves. The cost of a route is described by a single dimensionless metric.

OSPF allows sets of networks to be grouped together. Such a grouping is called an area. The topology of this area is hidden from the rest of the AS, which enables a significant reduction in routing traffic. Also, routing within the area is determined only by the area's own topology, lending the area protection from bad routing data. An area is a generalization of an IP subnetted network.

OSPF enables the flexible configuration of IP subnets so that each route distributed by OSPF has a destination and mask. Two different subnets of the same IP network number may have different sizes (that is, different masks). This is commonly referred to as variable-length subnetting. A packet is routed to the best (longest or most specific) match. Host routes are considered to be subnets whose masks are "all ones" (0xffffffff).

OSPF traffic can be authenticated or non-authenticated, or can use no authentication, simple/clear text passwords, or MD5-based passwords. This means that only trusted routers can participate in the AS routing. You can specify a variety of authentication schemes and, in fact, you can configure separate authentication schemes for each IP subnet.

Externally derived routing data (for example, routes learned from an exterior protocol such as BGP) is advertised throughout the AS. This externally derived data is kept separate from the OSPF ink state data.

Each external route can also be tagged by the advertising router, enabling the passing of additional information between routers on the boundary of the AS.

OSPF uses a link-state algorithm to build and calculate the shortest path to all known destinations.

# Basic OSPFv2 Configuration

This section describes how to implement basic OSPF routing.

## Enabling OSPF Routing For a Specific Context

Use the following configuration example to enable OSPF Routing for a specific context:

```
config
  context context_name
    router ospf
    end
```

Notes:

- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

## Enabling OSPF Over a Specific Interface

After you enable OSPF, specify the networks on which it will run. Use the following command to enable OSPF:

```
network network_ip_address/network_mask area { area_id | area_ip_address }
```



**Important** The default cost for OSPF on the system is 10. To change the cost, refer to the **ip ospf cost** command in the *Ethernet Interface Configuration Mode Commands* chapter of the *Command Line Interface Reference*.

Notes:

- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

## Redistributing Routes Into OSPF (Optional)

Redistributing routes into OSPF means any routes from another protocol that meet specified a specified criterion, such as route type, metric, or rule within a route-map, are redistributed using the OSPFv2 protocol to all OSPF areas. This is an optional configuration.

```
config
  context context_name
    router ospf
      redistribute { connected   | static }
      end
```

Notes:

- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

## Confirming OSPF Configuration Parameters

To confirm the OSPF router configuration, use the following command and look for the section labeled **router ospf** in the screen output:

```
show config context ctxt_name [ verbose ]
```

# OSPFv3 Routing

This section gives an overview of Open Shortest Path First Version 3 (OSPFv3) routing and its implementation in the system. It also describes how to enable the base OSPFv3 functionality and lists the commands that are available for more complex configurations.



**Important** On the , OSPFv3 routes with IPv6 prefix lengths less than /12 and between the range of /64 and /128 are not supported.

# OSPFv3 Overview

Much of OSPF version 3 is the same as OSPF version 2. OSPFv3 expands on OSPF version 2 to provide support for IPv6 routing prefixes and the larger size of IPv6 addresses. OSPFv3 dynamically learns and advertises (redistributes) IPv6 routes within an OSPFv3 routing domain

In OSPFv3, a routing process does not need to be explicitly created. Enabling OSPFv3 on an interface will cause a routing process and its associated configuration to be created.

# Basic OSPFv3 Configuration

This section describes how to implement basic OSPF routing.

## Enabling OSPFv3 Routing For a Specific Context

Use the following configuration example to enable OSPF Routing for a specific context:

```
config
  context context_name
    router ospfv3
    end
```

Notes:

- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

## Enabling OSPFv6 Over a Specific Interface

After you enable OSPFv3 specify the area in which it will run. Use the following command to enable OSPFv3:

```
area { area_id | area_ip_address } [ default-cost dflt-cost ] [ stub stub-area ]
[ virtual-link vl-neighbor-ipv4address ]
```



**Important** The default cost for OSPFv3 on the system is 10. To change the cost, refer to the **ipv6 ospf cost** command in the *Ethernet Interface Configuration Mode Commands* chapter of the *Command Line Interface Reference*.

Notes:

- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

### Redistributing Routes Into OSPFv3 (Optional)

Redistributing routes into OSPFv3 means any routes from another protocol that meet specified a specified criterion, such as route type, metric, or rule within a route-map, are redistributed using the OSPFv3 protocol to all OSPF areas. This is an optional configuration.

```
config
  context context_name
    router ospf3
      redistribute { connected | static }
      end
```

Notes:

- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

## Confirming OSPFv3 Configuration Parameters

To confirm the OSPF router configuration, use the following command and look for the section labeled **router ipv6 ospf** in the screen output:

```
show config context ctxt_name [ verbose ]
```

# Equal Cost Multiple Path (ECMP)

The system supports ECMP for routing protocols. ECMP distributes traffic across multiple routes that have the same cost to lessen the burden on any one route.

ECMP can be used in conjunction with most routing protocols, since it is a per-hop decision that is limited to a single router. It potentially offers substantial increases in bandwidth by load-balancing traffic over multiple paths

The following command configures the maximum number of equal cost paths that can be submitted by a routing protocol:

```
config
  context context_name
    ip routing maximum-paths [ max_num ]
```

Notes:

- *max_num* is an integer from 1 through 32.
- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# BGP-4 Routing

The Border Gateway Protocol 4 (BGP-4) routing protocol is supported through a BGP router process that is implemented at the context level.

Border Gateway Protocol (BGP) is an inter-AS routing protocol. An Autonomous System (AS) is a set of routers under a single technical administration that use an interior gateway protocol and common metrics to route packets within the AS. The set of routers uses an exterior gateway protocol to route packets to other autonomous systems.

BGP runs over TCP. This eliminates the need for the BGP protocol to implement explicit update fragmentation, retransmission, acknowledgement, and sequencing information. Any authentication scheme used by TCP may be used in addition to BGP's own authentication mechanisms.

BGP routers exchange network reachability information with other BGP routers. This information builds a picture of AS connectivity from which routes are filtered and AS-level policy decisions are enforced.

BGP-4 provides classless inter-domain routing. This includes support for advertising an IP prefix and eliminates the concept of network class within BGP. BGP-4 also allows the aggregation of routes, including the aggregation of AS paths.



**Important** On the , BGP routes with IPv6 prefix lengths less than /12 and between the range of /64 and /128 are not supported.

# Overview of BGP Support

Mobile devices communicate to the Internet through Home Agents (HAs). HAs assign IP addresses to the mobile node from a configured pool of addresses. These addresses are also advertised to Internet routers through an IP routing protocol to ensure dynamic routing. The BGP-4 protocol is used as a monitoring mechanism between an HA and Internet router with routing to support Interchassis Session Recovery (ICSR). (Refer to *Interchassis Session Recovery* for more information.)

The objective of BGP-4 protocol support is to satisfy routing requirements and monitor communications with Internet routers. BGP-4 may trigger an active to standby switchover to keep subscriber services from being interrupted.

The following BGP-4 features are supported:

- Exterior Border Gateway Protocol (EBGP) multi-hop
- Route Filtering for inbound and outbound routes
- Route redistribution and route-maps
- Support for BGP communities and extended communities in route maps
- Local preference for IPv4 and IPv6 (IBGP peers)

IP pool routes and loopback routes are advertised in the BGP domain in the following ways:

- Through BGP Configuration Mode **redistribution** commands, all or some of the connected routes are redistributed into the BGP domain. (IP pool and loopback routes are present in the IP routing table as connected routes.) The **network routemap** command provides the flexibility to change many BGP attributes.
- Through the BGP Configuration Mode **network** commands, connected routes are explicitly configured for advertisement into the BGP domain. The **network routemap** command provides the flexibility to change many BGP attributes. Refer to the *BGP Configuration Mode Commands* chapter of the *Command Line Interface Reference* for details on these commands.



**Important** If a BGP task restarts because of a processing card failure, a migration, a crash, or the removal of a processing card, all peering session and route information is lost.

# Configuring BGP

This section describes how to configure and enable basic BGP routing support in the system.

```
config
   context context_name
      router bgp AS_number
         neighbor ip_address remote-as AS_num
```

Notes:

- A maximum of 64 BGP peers are supported per context.
- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# Redistributing Routes Into BGP (Optional)

Redistributing routes into BGP simply means that any routes from another protocol that meet a specified criterion, such as a route type, or a rule within a route-map, are redistributed through the BGP protocol to all BGP areas. This is an optional configuration.

```
config
   context context_name
      router bgp as_number
         redistribute bgp { bgp | connected | static } [ metric  metric_value
 ] [ metric-type { 1 | 2 } ] [ route-map  route_map_name ]
```

Notes:

- The redistribution options are connected, ospf, rip, or static. Refer to the *Border Gateway Protocol Configuration Mode Commands* chapter of the *Command Line Interface Reference* for details on the **redistribute** command.
- A maximum of 64 route-maps are supported per context.
- Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# BGP Communities and Extended Communities

Route filtering based on a BGP community or extended community (route target) is configured via CLI Route Map Configuration mode commands.

# BGP Communities

## Configuring a BGP Community

A BGP community is a group of destinations that share some common attribute. Each destination can belong to multiple communities. Autonomous system administrators define to which communities a destination belongs.

You configure a BGP community via a Context Configuration mode command.

```
config
   context context_name
       ip community-list { named named_list | standard  identifier } { deny |
permit } { internet | local-AS | no-advertise | no-export | value
AS-community_number AS-community_number AS-community_number ...}
        { internet | local-AS | no-advertise | no-export | value
AS-community_number AS-community_number AS-community_number ...}
        { internet | local-AS | no-advertise | no-export | value
AS-community_number AS-community_number AS-community_number ...}
```

You can permit or deny the following BGP community destinations.

- **internet** – Advertise this route to the internet community, and any router that belongs to it.
- **local-AS** – Use in confederation scenarios to prevent sending packets outside the local autonomous system (AS).
- **no-advertise** – Do not advertise this route to any BGP peer, internal or external.
- **no-export** – Do not advertise to external BGP (eBGP) peers. Keep this route within an AS.
- **value** *AS-community_number* – Specifies a community string in AS:NN format, where AS = 2-byte AS-community hexadecimal number and NN = 2-byte hexadecimal number (1 to 11 characters).

You can enter multiple destinations and AS community numbers for each community. For additional information, see the *Command Line Interface Reference*.

Multiple community-list entries can be attached to a community-list by adding multiple permit or deny clauses for various community strings. Up to 64 community-lists can be configured in a context.

## Setting the Community Attribute

You set the BGP community attribute via a **set community** command in a route map.

```
config
   context context_name
       route-map map_name { deny | permit } sequence_number
          set community [additive]{ internet | local-AS | no-advertise |
no-export | none | value AS-community_number AS-community_number AS-community_number
 ...}
         { internet | local-AS | no-advertise | no-export | none | value
 AS-community_number AS-community_number AS-community_number ... }
         { internet | local-AS | no-advertise | no-export | none | value
 AS-community_number AS-community_number AS-community_number ... }
```

The **additive** option allows you to enter multiple destinations and AS community numbers. For additional information, see the *Command Line Interface Reference*.

### Filtering via a BGP Community

To filter routes based on a BGP community, you configure a **match** clause in a route map. The command sequence follows below.

```
config
   context context_name
       route-map map_name { deny | permit } sequence_number
          match community { named  named_list | standard  identifier }
```

## BGP Extended Communities

### Configuring a BGP Extended Community (Route Target)

A BGP extended community defines a route target. MPLS VPNs use a 64-bit Extended Community attribute called a Route Target (RT). An RT enables distribution of reachability information to the correct information table.

You configure a BGP extended community via a Context Configuration mode command.

```
config
   context context_name
       ip extcommunity-list { named named_list | standard  identifier } { deny
 | permit } rt rt_number rt_number rt_number ...
```

*rt_number* specifies a Route Target as a string in AS:NN format, where AS = 2-byte AS-community hexadecimal number and NN = 2-byte hexadecimal number (1 to 11 characters). You can add multiple route numbers to an IP extcommunity list.

Multiple extended community-list entries can be attached to an extended community-list by adding multiple permit or deny clauses for various extended community strings. Up to 64 extended community-lists can be configured in a context.

### Setting the Extended Community Attribute

You set the BGP extended community attribute via a **set extcommunity** command in a route map.

```
config
   context context_name
       route-map map_name { deny | permit } sequence_number
         set extcommunity rt rt_number rt_number rt_number ...
```

*rt_number* specifies a Route Target as a string in AS:NN format, where AS = 2-byte AS-community hexadecimal number and NN = 2-byte hexadecimal number (1 to 11 characters). You can add multiple route numbers to an IP extcommunity list.

### Filtering via a BGP Extended Community

To filter routes based on a BGP extended community (route target), you configure a **match** clause in a route map. The command sequence follows below.

```
config
   context context_name
       route-map map_name { deny | permit }
         [no] match extcommunity { named  named_list | standard  identifier }
```

## BGP Local Preference

The BGP local preference attribute is sent by a BGP speaker only to IBGP peers. It is set in a route map via the following command sequence:

```
config
  context context_name
      route-map map_name { deny | permit }
         set local-preference pref_number
```

There is no **match** clause corresponding to local preference in the route-map because local-preference is directly used in the route selection algorithm.

# ICSR and SRP Groups

BGP is employed with Interchassis Session Recovery (ICSR) configurations linked via Service Redundancy Protocol (SRP). By default an ICSR failover is triggered when all BGP peers within a context are down.

Optionally, you can configure SRP peer groups within a context. ICSR failover would then occur if all peers within a group fail. This option is useful in deployments in which a combination of IPv4 and IPv6 peers are spread across multiple paired VLANs, and IPv4 or IPv6 connectivity is lost by all members of a peer group.

For additional information refer to *Interchassis Session Recovery* in this guide and the description of the **monitor bgp**, **monitor diameter** and **monitor authentication-probe** commands in the *Service Redundancy Protocol Configuration Mode Commands* chapter of the *Command Line Interface Reference*.

# Advertising BGP Routes from a Standby ICSR Chassis

An SRP Configuration mode command enables advertising BGP routes from an ICSR chassis in standby state. This command and its keywords allow an operator to take advantage of faster network convergence accrued from deploying BGP Prefix Independent Convergence (PIC) in the Optical Transport Network Generation Next (OTNGN).

BGP PIC is intended to improve network convergence which will safely allow for setting aggressive ICSR failure detection timers.

```
configure
   context context_name
      service-redundancy-protocol
       advertise-routes-in-standby-state [ hold-off-time hold-off-time ] [
reset-bfd-nbrs bfd-down-time ]
       end
```

Notes:

- **hold-off-time** *hold-off-time* delays advertising the BGP routes until the timer expires. Specify *hold-off-time* in seconds as an integer from 1 to 300.
- After resetting BFD, **reset-bfd-nbrs** *bfd-down-time* keeps the BFD sessions down for the configured number of milliseconds to improve network convergence. Specify *bfd-down-time* as an integer from 50 to 120000.

# Configurable BGP Route Advertisement Interval for ICSR

By default, the MinRtAdvInterval is set for each peer with a value of 5 seconds for an iBGP peer and 30 seconds for an eBGP peer. An operator can use the **neighbor** *identifier* **advertisement-interval** command to globally change the default interval.

The BGP advertisement-interval can also be separately set for each address family. If configured, this value over-rides the peer's default advertisement-interval for that address-family only. BGP will send route update-message for each AFI/SAFI based on the advertisement-interval configured for that AFI/SAFI. If no AFI/SAFI advertisement-interval is configured, the peer-based default advertisement-interval is used.

In ICSR configurations, this feature can be used to speed route advertisements and improve network convergence times.

The **timers bgp icsr-aggr-advertisement-interval** command is available in both the BGP Address-Family (VPNv4/VPNv6) Configuration and BGP Address-Family (VRF) Configuration modes.

```
configure
   context context_name
      router bgp as_number
         address-family { ipv4 | ipv6 | vpnv4 | vpnv6 }
            timers bgp icsr-aggr-advertisement-interval seconds
```

Notes:

- *seconds* – sets the number of seconds as an integer from 0 to 30. Default: 0.

# BGP CLI Configuration Commands

The following table lists the BGP Configuration mode CLI commands that support the configuration of various BGP parameters. For additional information, refer to the *BGP Configuration Mode Commands* chapter of the *Command Line Interface Reference*

```
configure
   context context_name
      router bgp as_number
```

**Note** The SRP AS-Path prepending feature does not support 32-bit AS numbers.

***Table 1: BGP Configuration Mode CLI Commands***

| bgp Command | Description |
|---|---|
| **accept-zero-as-rd** | Configures to accept VPN prefixes with Route Distinguisher (RD) value having Administrator Subfield, which is an AS number 0. |
| **address-family { ipv4 \| ipv6 }** | Enters the IPv4 or IPv6 Address Family configuration mode. |
| **address-family { vpnv4 \| vpnv6 }** | Enters the VPNv4 or VPNv6 Address Family configuration mode. |

| bgp Command | Description |
|---|---|
| **bgp graceful-restart { restart-time** *rest_time* \| **stalepath-time** *stale_time* \| **update-delay** *delay* | Defines the BGP-specific parameters regarding graceful restarts. |
| **description** *text* | Allows you to enter descriptive text for this configuration. |
| **distance { admin** *distance* **prefix** *prefix_addr* **[ route-access-list** *list_name* **] \| bgp external** *ebgp_dist* **internal** *ibgp_dist* local *local_dist* **}** | Defines the administrative distance for routes. The administrative distance is the default priority for a specific route or type route. |
| **enforce-first-as** | Enforces the first AS for Exterior Border Gateway Protocol (eBGP) routes. |
| **ip vrf** *vrf_name* | Adds a preconfigured IP VRF context instance to the BGP ASN and configures the BGP attributes and related parameters to the VRF. |
| **maximum-paths { ebgp** *max_num* \| **ibgp** *max_num* **}** | Enables forwarding packets over multiple paths and specifies the maximum number of external BGP (eBGP) or internal BGP (iBGP) paths between neighbors. |
| **neighbor** *ip_address* **{ activate \| advertisement-interval** *adv_time* **\| capability graceful-restart \| default-originate [ route-map** *map_name* **] \| distribute-list** *dist_list***{ in \| out } \| ebgp-multihop [ max-hop number ] \| encrypted password** *encrypted_password* **\| fall-over bfd [ multihop ] \| filter-list filt_list { in \| out } \| max-prefix** *max_num* **[ threshold** *thresh_percent* **] [ warning-only ] \| next-hop-self \| password** *password* **\| remoteas** *AS_num* **\| remove-private-AS \| restart-time** *rest_time* **\| route-map** *map_name* **{ in \| out } \| send-community { both \| extended \| standard } \| shutdown \| srp-activated-soft-clear \| timers { [ connect-interval** *conn_time* **] \| [ keepalive-interval** *keep_time* **holdtimeinterval** *hold_time* **] } \| update-source** *ip_address* **\| weight** *value* **}** | Configures BGP routers that interconnect to non-broadcast networks. Note that a remote AS number must be specified for a neighbor before other parameters can be configured.<br>**Note:** The **advertisement-interval** must be explicitly configured for an address-family so that it can take effect for that address-family. By default it will be applicable only for the IPv4 address-family. Specify the address family via the **address-family** command. You can then set the neighbor advertisement-interval in the address family configuration mode. |
| **network** *ip_address/mask* **[ route-map** *map_name* **]** | Specifies a network to announce via BGP. |
| **redistribute { connected \| ospf \| rip \| static } [ route-map** *map_name* **]** | Redistributes routes via BGP from another protocol to BGP neighbors. |
| **router-id** *ip_address* | Overrides the configured router identifier and causes BGP peers to reset. |
| **scan-time** *time* | Configures the BGP background scanner interval in seconds. BGP monitors the next hop of the installed routes to verify next-hop reachability and to select, install, and validate the BGP best path. loops. |
| **timers bgp keepalive-interval** *interval* **holdtime-interval** *time* **[ min-peer-holdtimeinterval** *time* **]** | Configures BGP routing timers. |

# Confirming BGP Configuration Parameters

To confirm the BGP router configuration, use the following command and look for the section labeled **router bgp** in the screen output:

**show config context** *ctxt_name* **[ verbose ]**

# BGP Peer Limit

## Feature Summary and Revision History

### Summary Data

| | |
|---|---|
| Applicable Product(s) or Functional Area | All |
| Applicable Platform(s) | VPC - DI |
| Feature Default | Disabled - Configuration Required |
| Related Changes in This Release | Not Applicable |
| Related Documentation | • *Command Line Interface Reference*<br>• *Statistics and Counters Reference*<br>• *VPC-DI System Administration Guide* |

### Revision History

| Revision Details | Release |
|---|---|
| First introduced. | 21.8 |

## Feature Description

In the with CUPS architecture, the flexibility of BGP peering is provided across packet processing cards namely, Session Function (SF) cards, including the demux SF cards.

In deployment setups based on “contrail” model of the SDN , each packet processing card has a vRouter within the compute node. In this model, with the current flexible BGP peering scheme, the BGP configurations needs to be implemented on each of those vRouters. This poses a challenge to service providers when there are large number of SF cards in their network. The number of lines of configuration required, poses a scaling challenge.

To overcome this challenge, the BGP Peer Limit feature is introduced that restricts BGP peering to only **two** SF cards in the VPC-DI architecture. This feature mandates that the routing table has only two routes corresponding to the two SF cards, with a third route being a “blackhole” or a “null” route. To ensure that the new routes are longest-prefix-match routes, provisioning of only host-addresses only (/32 bitmask) is enforced. This drastically reduces the amount of configuration and the routing table size.

# How It Works

This feature is implemented using the **ip route kernel** command. When configured, BGP peering is restricted to only the two SF cards with the special route.

When the **blackhole** keyword is configured, it enables the kernel routing engine to block or drop packets going out of the node. This is not limited to any interface and defaults to a wildcard interface.

For information on configuring the BGP Peer Limit feature, see the "Configuring BGP Peer Limit" section.

### Limitations

- This feature support is limited only to the context level.
- There is no support provided at the VRF level.
- This feature is supported only for IPv4.

# Configuring BGP Peer Limit

The following section provides the configuration command to enable or disable the functionality.

### Configuring Packet Processing Card Routes

Use the following CLI commands to add the special (static) route to any two packet processing interfaces (SF cards) defined in the context configuration.

```
configure
   context context_name
     [ no ] ip route kernel ip_address/ip_address_mask_combo egress_intrfc_name
 cost number
      end
```

**NOTES**:

- **no**: Deletes the added routes.
- **kernel**: Allows static route in the kernel routing table options.
- **ip_address/ip_address_mask_combo**: Specifies a combined IP address subnet mask bits to indicate what IP addresses the route applies to. *ip_address_mask_combo* must be specified using CIDR notation where the IP address is specified using IPv4 dotted-decimal notation and the mask bits are a numeric value, which is the number of bits in the subnet mask.
- *egress_intrfc_name* : Specifies the name of an existing egress interface as an alphanumeric string of 1 through 79 characters.
- **cost** *number* : Defines the number of hops to the next gateway. The cost must be an integer from 0 through 255 where 255 is the most expensive. Default is 0.
- This functionality is disabled by default.

### Configuring Blackhole Route

Use the following CLI commands to block or drop packets going out of the node.

```
configure
   context context_name
```

```
    [ no ] ip route kernel ip_address/ip_address_mask_combo egress_intrfc_name
cost number blackhole
    end
```

**NOTES**:

- **no**: Deletes the added routes.
- **kernel**: Allows static route in the kernel routing table options.
- **ip_address/ip_address_mask_combo**: Specifies a combined IP address subnet mask bits to indicate what IP addresses the route applies to. *ip_address_mask_combo* must be specified using CIDR notation where the IP address is specified using IPv4 dotted-decimal notation and the mask bits are a numeric value, which is the number of bits in the subnet mask.
- *egress_intrfc_name* : Specifies the name of an existing egress interface as an alphanumeric string of 1 through 79 characters. The default is "*", that is, a wildcard interface.
- **cost** *number* : Defines the number of hops to the next gateway. The cost must be an integer from 0 through 255 where 255 is the most expensive. The default is 0.
- **blackhole**: Defines the blackhole route to install in the kernel to to block or drop packets.
- This functionality is disabled by default.

## Monitoring and Troubleshooting

This section provides information regarding the CLI command available in support of monitoring and troubleshooting the feature.

### Show Command(s) and/or Outputs

This section provides information regarding the show command and/or its output in support of this feature.

*show ip route*

This show command CLI now includes the value for the following new field when a static route is added to any two packet processing interfaces (SF cards).

kernel-only

# Bidirectional Forwarding Detection

Bidirectional Forwarding Detection (BFD) is a network protocol used to detect faults between two forwarding engines connected by a link. BFD establishes a session between two endpoints over a particular link. If more than one link exists between two systems, multiple BFD sessions may be established to monitor each one of them. The session is established with a three-way handshake, and is torn down the same way. Authentication may be enabled on the session. A choice of simple password, MD5 or SHA1 authentication is available.



**Important** On the , BFD routes with IPv6 prefix lengths less than /12 and between the range of /64 and /128 are not supported.

# Overview of BFD Support

BFD does not have a discovery mechanism; sessions must be explicitly configured between endpoints. BFD may be used on many different underlying transport mechanisms and layers, and operates independently of all of these. Therefore, it needs to be encapsulated by whatever transport it uses.

Protocols that support some form of adjacency setup, such as OSPF or IS-IS, may also be used to bootstrap a BFD session. These protocols may then use BFD to receive faster notification of failing links than would normally be possible using the protocol's own keepalive mechanism.

In asynchronous mode, both endpoints periodically send Hello packets to each other. If a number of those packets are not received, the session is considered down.

When Echo is active, a stream of Echo packets is sent to the other endpoint which then forwards these back to the sender. Echo can be globally enabled via the **bfd-protocol** command, and/or individually enabled/disabled per interface. This function is used to test the forwarding path on the remote system.

The system supports BFD in asynchronous mode with optional Echo capability via static or BGP routing.



**Important** On an ASR 5500 one of the packet processing cards must be configured as a demux card in order for BFD to function. See the *Configuring a Demux Card* section in the *System Settings* chapter for additional information.

# Configuring BFD

This section describes how to configure and enable basic BFD routing protocol support in the system.

There are several factors affecting the configuration of BFD protocol:


- Configuring a BFD Context, on page 19
- Configuring IPv4 BFD for Static Routes, on page 20
- Configuring IPv6 BFD for Static Routes, on page 20
- Configuring BFD for Single Hop, on page 20
- Configuring Multihop BFD, on page 21
- Scaling of BFD, on page 21
- Associating BGP Neighbors with the Context, on page 21
- Associating OSPF Neighbors with the Context, on page 22
- Associating BFD Neighbor Groups with the BFD Protocol, on page 22
- Enabling BFD on OSPF Interfaces, on page 22
- Monitoring BFD Connection for ICSR, on page 22


## Configuring a BFD Context

```
config
  context context_name
    bfd-protocol
    [ bfd echo ]
    exit
```

Notes:

- Echo function can be optionally enabled for all interfaces in this context.

- 16 BFD sessions per context and 64 per chassis.

## Configuring IPv4 BFD for Static Routes

Enable BFD on an interface.

```
config
   context bfd_context_name
      interface if_name
         ip address ipv4_address ipv4_mask
         bfd interval interval_value min_rx rx_value multiplier multiplier_value
         [ bfd echo ]
      exit
```

Configure BFD static route.

```
ip route static bfd if_name ipv4_gw_address
```

Add static routes.

```
ip route ipv4_address ipv4_mask
ip route ipv4_address ipv4_mask
```

## Configuring IPv6 BFD for Static Routes

Enable BFD on an Interface

```
config
   context bfd_context_name
   interface if_name
      ipv6 address ipv6_address ipv6_mask
         bfd interval interval_value min_rx rx_value multiplier multiplier_value
         [ bfd echo ]
         exit
```

Configure BFD static route.

```
ipv6 route static bfd if_name ipv6_gw_address
```

Add static routes.

```
ipv6 route ipv6_address ipv6_mask
ipv6 route ipv6_address ipv6_mask
```



**Important** On the ASR 5500, static routes with IPv6 prefix lengths less than /12 and between the range of /64 and /128 are not supported.

## Configuring BFD for Single Hop

Enable BFD on an interface.

```
config
   context bfd_context_name
      interface if_name
         ip address ipv4_address ipv4_mask
```

```
ipv6 address ipv6_address ipv6_mask
bfd interval interval_value min_rx rx_value multiplier multiplier_value
[ bfd echo ]
exit
```

Enable BFD on a BGP Neighbor. For additional information, see Associating BGP Neighbors with the Context, on page 21.

Enable BFD on an OSPF Neighbor. For additional information, see Associating OSPF Neighbors with the Context, on page 22.



**Important** On the ASR 5500, routes with IPv6 prefix lengths less than /12 and between the range of /64 and /128 are not supported.

## Configuring Multihop BFD

Enable BFD on an interface.

```
config
  context bfd_context_name
    interface if_name
      ip address ipv4_address ipv4_mask
      ipv6 address ipv6_address ipv6_mask
      bfd interval interval_value min_rx rx_value multiplier multiplier_value
      [ bfd echo ]
      exit
```

Configure a Multihop BFD session.

```
bfd-protocol
  bfd multihop peer  destination-address interval interval-value multiplier
multiplier-value
```

Enable BFD on a BGP Neighbor. For additional information, see Associating BGP Neighbors with the Context, on page 21.

## Scaling of BFD

Configure an active BFD session using one of the above methods and use same BFD neighbor while configuring the active interface. For additional information, see Associating BFD Neighbor Groups with the BFD Protocol, on page 22.

```
bfd-protocol
  bfd nbr-group-name grp_name active-if-name if_name nexthop_address
```

Apply the same BFD results to one or more passive interfaces.

```
bfd nbr-group-name grp_name passive-if-name if_name nexthop_address
bfd nbr-group-name grp_name passive-if-name if_name nexthop_address
```

## Associating BGP Neighbors with the Context

```
config
  context context_name
```

```
router bgp AS_number
   neighbor neighbor_ip-address remote-as rem_AS_number
   neighbor neighbor_ip-address ebgp-multihop max-hop max_hops
   neighbor neighbor_ip-address update-source update-src_ip-address
   neighbor neighbor_ip-address failover bfd [ multihop ]
```

Notes:

- Repeat the sequence to add neighbors.

## Associating OSPF Neighbors with the Context

```
config
   context context_name
      router ospf
         neighbor neighbor_ip-address
```

Notes:

- Repeat the sequence to add neighbors.

## Associating BFD Neighbor Groups with the BFD Protocol

```
config
   context context_name
      bfd-protocol
         bfd nbr-group-name grp_name active-if-name if_name nexthop_address
         bfd nbr-group-name grp_name passive-if-name if_name nexthop_address
```

## Enabling BFD on OSPF Interfaces

### All OSPF Interfaces

```
config
   context context_name
      router ospf
         bfd-all-interfaces
```

### Specific OSPF Interface

```
config
   context context_name
      interface interface_name
         broadcast
            ip ospf bfd
```

## Monitoring BFD Connection for ICSR

For ICSR configurations, the following command sequence initiates monitoring of the connection between the primary chassis and the BFD neighbor in the specified context. If the connection drops, the standby chassis becomes active.

```
config
   context context_name
```

```
    service-redundancy-protocol
      monitor bfd context context_name { ipv4_address | ipv6_address } {
chassis-to-chassis | chassis-to-router }
```

Notes:

- *ipv4 _address | ipv6_address* defines the IP address of the BFD neighbor to be monitored, entered using IPv4 dotted-decimal or IPv6 colon-separated-hexadecimal notation
- **chassis-to-chassis** enables BFD to run between primary and backup chassis on non-SRP links.
- **chassis-to-router** enables BFD to run between chassis and router.

## Saving the Configuration

Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# Chassis-to-Chassis BFD Monitoring for ICSR

An operator can configure BFD to more quickly advertise routes during an ICSR switchover. This solution complements the feature that allows the advertising of BGP routes from a Standby ICSR chassis. The overall goal is to support more aggressive failure detection and recovery in an ICSR configuration when implementing of VoLTE.

You must configure the following features for chassis-to-chassis BFD monitoring in ICSR configurations:

- Enable Primary Chassis BFD Monitoring, on page 23.
- Set BFD to Ignore ICSR Dead Interval, on page 23.
- Configure ICSR Switchover Guard Timer, on page 24.
- Enable BFD Multihop Fall-over , on page 24.
- Enable Advertising BGP Routes from Standby ICSR Chassis, on page 26.

## Enable Primary Chassis BFD Monitoring

You must enable monitoring of the connection between the primary chassis and specified BFD neighbors. If the connection drops, the standby chassis becomes active. For more information, see Monitoring BFD Connection for ICSR, on page 22.

## Set BFD to Ignore ICSR Dead Interval

The SRP Configuration mode **bfd-mon-ignore-dead-interval** command causes the standby ICSR chassis to ignore the dead interval and remain in the standby state until all the BFD chassis-to-chassis monitors fail.

Enable this feature in association with BFD chassis-to-chassis monitoring to support more aggressive ICSR failure detection times.

```
configure
   context context_name
      service-redundancy-protocol variable
         bfd-mon-ignore-dead-interval
         end
```

## Configure ICSR Switchover Guard Timer

The SRP Configuration mode **guard timer** command configures the redundancy-guard-period and monitor-damping-period for SRP service monitoring.

Use these guard timers to ensure that local failures, such as card reboots and task restarts, do not result in ICSR events which can be disruptive.

```
configure
   context context_name
      service-redundancy-protocol variable
         guard-timer { aaa-switchover-timers { damping-period seconds |
guard-period seconds } | diameter-switchover-timers { damping-period seconds
 | guard-period seconds } | srp-redundancy-timers { aaa  { damping-period
seconds | guard-period seconds } | bgp { damping-period seconds | guard-period
 seconds } | diam  { damping-period seconds | guard-period seconds } }
         end
```

Notes:

- **aaa-switchover-timers** – sets timers that prevent back-to-back ICSR switchovers due to an AAA failure (post ICSR switchover) while the network is still converging.
  - **damping-period** – configures a delay time to trigger an ICSR switchover due to a monitoring failure within the guard-period.
  - **guard-period** – configures the local-failure-recovery network-convergence timer.
- **diameter-switchover-timers** – sets timers that prevent a back-to-back ICSR switchover due to a Diameter failure (post ICSR switchover) while the network is still converging.
  - **damping-period** – configures a delay time to trigger an ICSR switchover due to a monitoring failure within the guard-period.
  - **guard-period** – configures the local-failure-recovery network-convergence timer.
- **srp-redundancy-timers** – sets timers that prevent an ICSR switchover while the system is recovering from a local card-reboot/critical-task-restart failure.
  - **aaa** – local failure followed by AAA monitoring failure
  - **bgp** – local failure followed by BGP monitoring failure
  - **diam** – local failure followed by Diameter monitoring failure

## Enable BFD Multihop Fall-over

A **fall-over bfd multihop** *mhsess_name* keyword in the Context Configuration mode ip route and ipv6 route commands enables fall-over BFD functionality for the specified multihop session. The **fall-over bfd** option uses BFD to monitor neighbor reachability and liveliness. When enabled it will tear down the session if BFD signals a failure.

```
configure
   context context_name
      ip route { ip_address/ip_mask | ip_address ip_mask } { gateway_ip_address |
next-hop next_hop_ip_address | point-to-point | tunnel } egress_intrfc_name [
cost cost ] [ fall-over bfd multihop mhsess_name ] [ precedence precedence ] [
 vrf vrf_name [ cost value ] [ fall-over bfd multihop mhsess_name ] [ precedence
```

```
precedence ] +
      end
```

The **ip route** command now also allows you to add a static multihop BFD route.

```
      ip route static multihop bfd mhbfd_sess_name local_endpt_ipaddr
remote_endpt_ipaddr
```



**Important** SNMP traps are generated when BFD sessions go up and down (BFDSessUp and BFDSessDown).

### ip route Command

```
configure
   context context_name
      ip route { ip_address/ip_mask | ip_address ip_mask } { gateway_ip_address |
next-hop next_hop_ip_address | point-to-point | tunnel } egress_intrfc_name [
cost cost ] [ fall-over bfd multihop mhsess_name ] [ precedence precedence ] [
 vrf vrf_name [ cost value ] [ fall-over bfd multihop mhsess_name ] [ precedence
 precedence ] +
      end
```

The **ip route** command now also allows you to add a static multihop BFD route.

```
      ip route static multihop bfd mhbfd_sess_name local_endpt_ipaddr
remote_endpt_ipaddr
```

### ip routev6 Command

```
configure
   context context_name
     ipv6 route ipv6_address/prefix_length { interface name | next-hop ipv6_address
 interface name } [ cost  cost] [ fall-over bfd multihop mhsess_name ] [
precedence precedence ] [ vrf vrf_name [ cost value ] [ fall-over bfd multihop
 mhsess_name ] [ precedence precedence ]
      end
```

The **ipv6 route** command now also allows you to add a static multihop BFD route.

```
      ipv6 route static multihop bfd mhbfd_sess_name local_endpt_ipv6addr
remote_endpt_ipv6addr
```

## Adjust BFD Interval

Set the transmit interval (in milliseconds) between BFD packets to meet the convergence requirements of your network deployment.

```
configure
   context context_name
      interface interface_name broadcast
         bfd interval interval_num  min_rx milliseconds multiplier value
         end
```

Notes:

- *milliseconds* is an integer from 50 through 10000. (Default 50)

## Enable Advertising BGP Routes from Standby ICSR Chassis

For information on configuring the feature, see Advertising BGP Routes from a Standby ICSR Chassis , on page 13.

## Saving the Configuration

Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

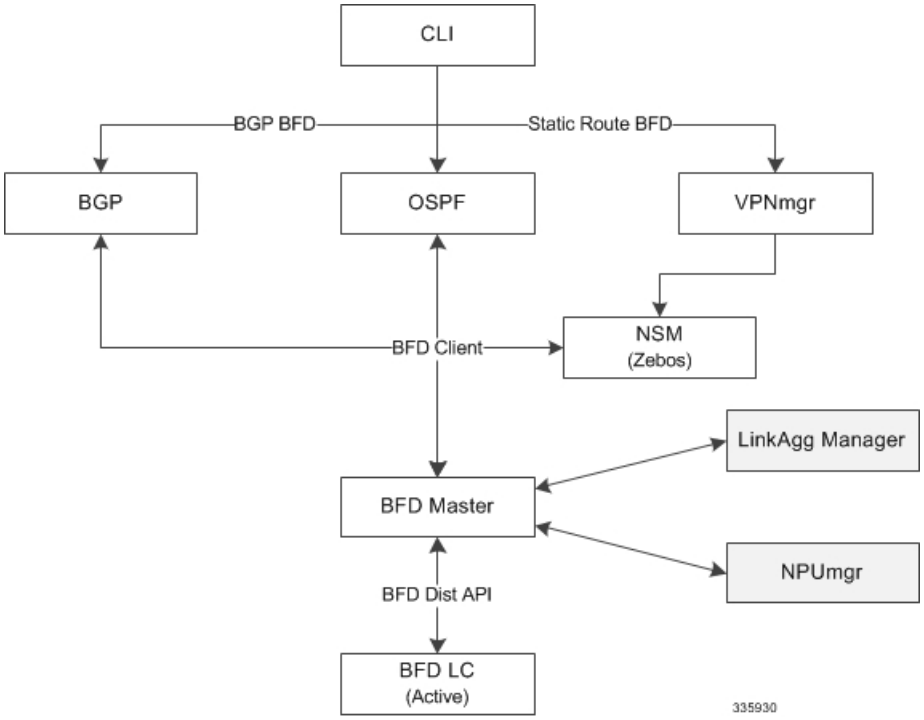
# BFD Support for Link Aggregation Member Links

Member-link based BFD detects individual link failures faster than LACP and reduces the overall session/traffic down period as a result of single member link failure.

## Overview

A BFD Configuration mode CLI command configures BFD interactions with the linkagg task. Once a session is configured, BFD creates per member link BFD sessions and starts sending packets on each of the linkagg member links. If a member link BFD session fails, StarOS notifies failures to the linkagg task.

***Figure 1: BFD Interactions***




If you define a linkagg-peer using a slot number, you may configure a linkagg-peer for a redundant LC (Line Card) slot which must also specify a slot in its member-link configuration. Likewise, if you configure a linkagg-peer without a slot, you must delete it before configuring a peer with a slot specified.

**Important** Only one IPv4 or IPv6 BFD session-based configuration is allowed per linkagg interface for compliance with RFC 7130.

## Configuring Support for BFD Linkagg Member-links

The **bfd linkagg-peer** command enables member-link BFD and configures the BFD link aggregation (linkagg) session values [RFC 7130].

```
configure
  context context_name
    bfd-protocol
      bfd linkagg-peer linkagg_group_id local-endpt-addr local-endpt_ipaddress
remote-endpt-addr remote_endpt_ipaddress interval tx_interval min_rx rx_interval
multiplier multiplier_value [ slot slot_number ]
      no bfd linkagg-peer linkagg_group_id [ slot slot_number ]
      end
```

Notes:

- *linkagg_group_id* specifies the LAG number as an integer from 1 through 255.
- **local-endpt-addr** *local-endpt_ipaddress* specifies the source address of the multihop BFD session in IPv4 or IPv6 notation.
- **remote-endpt-addr** *remote-endpt_ipaddress* specifies the remote address of the multihop BFD session in IPv4 or IPv6 notation.
- **interval** *tx_interval* specifies the transmit interval of control packets in milliseconds as an integer from 50 through 10000.
- **min_rx** *rx_interval* specifies the receive interval of control packets in milliseconds as an integer from 50 through 10000.
- **multiplier** *multiplier_value* specifies the value used to compute hold-down time as an integer from 3 through 50.
- **slot** *slot_number* for redundant active-standby link aggregation, this option specifies the card for which this configuration is intended.

## Saving the Configuration

Save your configuration as described in the *Verifying and Saving Your Configuration* chapter.

# Viewing Routing Information

To view routing information for the current context, run one of the following Exec mode commands;

- **show ip route**: Displays information for IPv4 routes in the current context.
- **show ipv6 route**: Displays information for ipv6 routes in the current context.
- **show ip static-route**: Displays information only for IPv4 static routes in the current contextospf.
- **show ip ospf**: Displays IPv4 OSPF process summary information in the current context.
- **show ipv6 ospf**: Displays IPv6 OSPFv3 process summary information in the current context.

- **show ip bgp**: Displays IPv4 BGP information.

This example shows sample output of the command, **show ip route**.

```
[local]host_name# show ip route
"*" indicates the Best or Used route.

 Destination         Nexthop          Protocol   Prec Cost Interface
*44.44.44.0/24       208.230.231.50   static     1    0    local1
*192.168.82.0/24     0.0.0.0          connected  0    0
*192.168.83.0/24     0.0.0.0          connected  0    0
 208.230.231.0/24    0.0.0.0          ospf       110  10   local1
*208.230.231.0/24    0.0.0.0          connected  0    0    local1
Total route count: 5
```