BFD Configuration Guide, 17.2.0

November 2017

Supporting AT&T Vyatta Network Operating System

Contents

Copyright Statement

The training materials and other content provided herein for assistance in training on the Vyatta vRouter may have references to Brocade as the Vyatta vRouter was formerly a Brocade product prior to AT&T's acquisition of Vyatta. Brocade remains a separate company and is not affiliated to AT&T.

About This Guide

This guide describes how to run BFD on AT&T products that run on the AT&T Vyatta Network Operating System (referred to as a virtual router, vRouter, or router in the guide).

Overview of BFD

Understanding BFD

Bidirectional Forwarding Detection (BFD) is a simple control protocol that is used to detect faults between two forwarding systems that are connected by a link. BFD is comparable to the detection components of well-known routing protocols.

A requirement of networking solutions is the rapid detection of communication failures between adjacent systems to establish alternative paths quickly. The time-to-detect failure in existing protocols is no better than one second, which is far too long for some applications, and represents data loss at gigabit rates.

Consider two systems, R1 and R2, that already share a routing protocol such as BGP on a data plane interface named `dp0s7`. If we set up a BFD session between R1 and R2, R1 and R2 begin to transmit packets periodically over each path between the two systems. The control packets adhere to a previously agreed-upon frequency. If R1 stops receiving BFD packets for a specified time, some component in that particular bidirectional path to R2 is assumed to have failed. Under some conditions, R1 and R2 may negotiate not to send periodic BFD packets to reduce overhead. Thus, allows for fast systems on a shared medium with a slow system to rapidly detect failures between the fast while allowing the slow system to participate to the best of its ability.

Figure 1: An overview of BFD

BFD works with both peers that are connected directly and peers that are multiple hops away. No automatic discovery of BFD neighbors occurs; the sessions must be explicitly configured for each new system. For more information about BFD, refer to RFC 5880, *Bidirectional Forwarding Detection (BFD)*, at https://tools.ietf.org/html/rfc5880.

> **Note:** BFD is a failure-detection mechanism only; the routing protocol is responsible for taking a failover action.

BFD operating modes

BFD can operate in any of three available modes: asynchronous, demand, and echo. The three BFD modes are described in the following paragraphs.

- Asynchronous mode—In this mode, the systems periodically send BFD control packets to one another, and if a number of those packets in a row are not received by the other system in a stipulated time, the session is declared to be down.
- Demand mode—In this mode, a system has an independent way of verifying that it has connectivity to the other system. After a BFD session is established, such a system may ask the other system to stop sending BFD control packets, except when the system detects the need to verify connectivity explicitly. In such a case, a short sequence of BFD control packets is exchanged. Demand mode may operate independently in each direction or simultaneously.
- Echo mode—In this mode, packets are transmitted and consumed by the originating system. This mode exercises the forwarding path of the remote end. Echo mode is used when more-aggressive intervals are required.

Note: AT&T supports only the asynchronous mode currently.

BFD timers

The BFD intervals for packet transmission, packet reception, and session detection are continuously negotiated; thus the intervals can change at any time. These intervals are referred to as BFD timers.

Both BFD nodes negotiate and converge on the same timer interval which is the higher of the two intervals. The detection time is independent in each direction as the multiplier can vary for each node.

The three modes of BFD have three types of BFD timers associated with them:

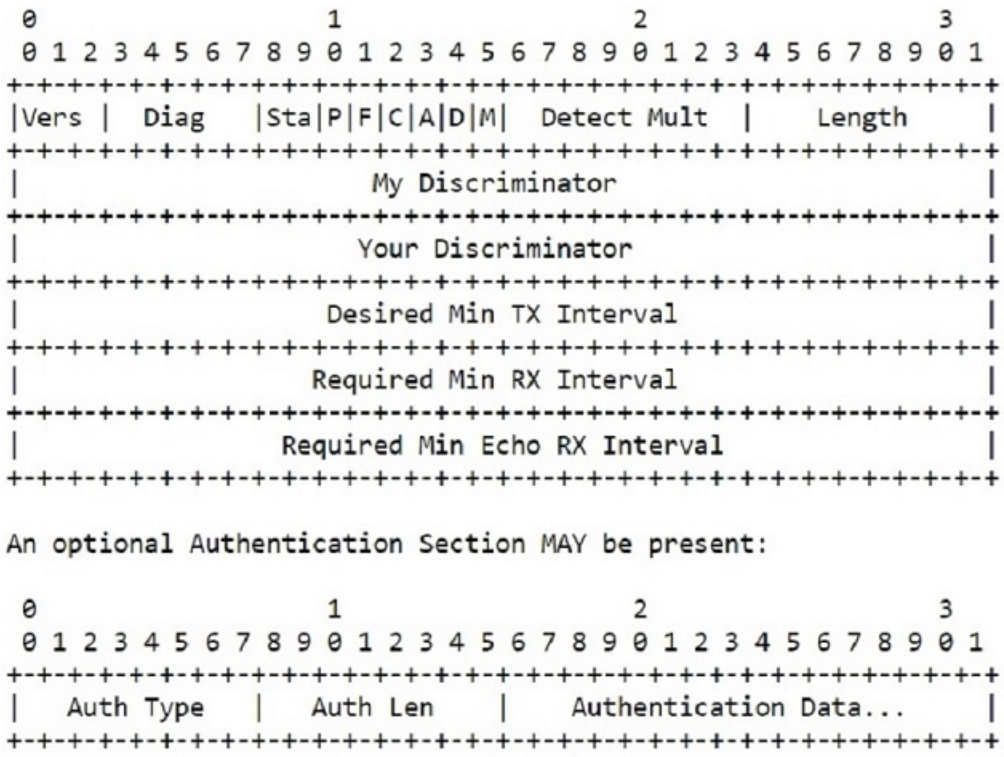
- Tx timer—Minimum time the system prefers to use for transmitting consecutive BFD packets
- Rx timer—Minimum time required by the receiving system to receive consecutive packets
- Echo Rx timer—Minimum time required by the transmitting system to receive consecutive echo packets
- Multiplier—Negotiated multiplier for the BFD session

Note: AT&T does not support echo mode currently.

Format of a BFD control packet

A BFD control packet is sent in an encapsulation and the packet has a mandatory section and an optional authentication section. The format of the authentication section, if present, depends on the type of authentication in use.

Figure 2: Format of a BFD control packet



The preceding figure displays a cross-sectional view of the BFD control packet. The packet dividers are described here:

- Vers (Version)—Version of the BFD protocol.

- Diag (Diagnostics)—Diagnostic code that indicates the reason for the last change of the local system in a BFD session state.
- Sta (State)—State of the current BFD session as seen by the transmitting system.
- P (Poll)—Indication that the transmitting system is requesting verification of connectivity or a parameter change, and is expecting a packet with the Final (F) bit in reply. If not set, the transmitting system is not requesting verification.
- F (Final)—Indication that the transmitting system is responding to a received BFD control packet that has the Poll (P) bit set. If not set, the transmitting system is not responding to a Poll.
- C (Control Plane Independent)—Indication that the implementation of the BFD session of the transmitting system is independent of the control plane. If not set, the implementation of the BFD session of the transmitting system depends on the control plane.
- A (Authentication Present)—Indication that the session is to be authenticated. If not set, the session is not authenticated.
- D (Demand)—Indication that the demand mode is active in the transmitting system. In demand mode, the system recognizes that the session is active in both directions, and directs the remote system to halt the periodic transmission of BFD control packets. If not set, demand mode is not active in the transmitting system.
- M (Multipoint)—Indication that the bit is reserved for future point-to-multipoint extensions to BFD. It must be zero on both transmit and receipt.
- Detect Mult—Detection time multiplier. The negotiated transmission interval, multiplied by this value, provides the detection time for the receiving system in asynchronous mode.
- Length—Length of the BFD control packet, in bytes.
- My Discriminator—Unique, nonzero discriminator that is generated by the transmitting system, used to demultiplex multiple BFD sessions between the same pair of systems.
- Your Discriminator—Discriminator received from the corresponding remote system. This field reflects the received value of My Discriminator, or is zero if that value is unknown.
- Desired Min TX Interval—Minimum interval, in microseconds, that the local system prefers to use when transmitting BFD control packets.
- Required Min RX Interval—Minimum interval, in microseconds, between received BFD control packets that the system is capable of supporting.
- Required Min Echo RX Interval—Minimum interval, in microseconds, between received BFD echo packets that the system is capable of supporting.
- Auth Type—Authentication type in use, if the Authentication Present (A) bit is set.
- Auth Len—Length, in bytes, of the authentication section, including the Auth Type and Auth Len fields.
- Authentication Data—Information used to authenticate the BFD sessions.

BFD authentication

Authentication for BFD sessions is disabled by default. AT&T recommends the implementation of BFD authentication when you run BFD over multiple hops or through insecure tunnels.

The authentication section in the BFD control packet is optional. Based on the type of authentication, the receiving system determines the validity of the received packet. The receiving system either accepts the packet for further processing or discards it. For authentication to work, both systems in a BFD session must use the same authentication type, authentication keys, and so on.

BFD authentication algorithms include the following:

- Simple password
- Keyed MD5
- Meticulous keyed MD5
- Keyed SHA1
- Meticulous keyed SHA1

Simple password authentication involves one or more passwords with corresponding key IDs that are configured in each system that is running BFD. One pair of a password and a key ID is carried in each BFD control packet. The

receiving system accepts the packet if the password-key ID pair matches a password-key ID pair configured in that system. The password is a binary character string, and is 1 to 16 bytes in length.

Note: AT&T supports simple password authentication currently.

Advantages of BFD

BFD was devised as an fault-detection mechanism because most modern routing protocols cannot detect failures in milliseconds (ms).

BFD has some other advantages as listed here:

- BFD is not tied to any particular routing protocol; it can be used as a generic and consistent failure-detection mechanism for all kinds of routing protocols.
- The periodic transmission and reception of BFD packets works from the data plane. It is less CPU intensive than routing protocols that exist only on the control plane.
- The timing of the BFD control packets can be adjusted dynamically to avoid the pitfalls of false failure messages.

Limitations of BFD for the AT&T Vyatta vRouter

The current release of the AT&T Vyatta vRouter has the following BFD limitations.

- Demand mode is not supported.
- BFD over LDP and over LAG are not supported.
- Echo mode is not supported.

The current release of the AT&T Vyatta vRouter has the following BFD features:

- Protocols such as BGP, OSPFv2, OSPFv3, and static routes are supported.
- Both IPv4 and IPv6 addresses are supported.
- Both single hop and multiple hops are supported.
- The following parameters for BFD are supported:
  - minimum-rx interval
  - minimum-tx interval
  - detect-multiplier
  - simple password authentication

Basic BFD Configuration

BFD workflow

Bidirectional Forwarding Detection (BFD) does not have a peer-discovery mechanism. You must configure BFD for each system.

Note: You must configure the routing protocol before configuring BFD. To configure the routing protocol, refer to the AT&T Vyatta vRouter documentation for that particular routing protocol.

The following steps give an overview of the BFD workflow.

1. The routing protocol registers itself to BFD as a BFD client.
2. The BFD timers are negotiated between the two BFD peers by using the periodic exchange of packets.
3. The BFD session can go down due to an expiry of the BFD timer or due to a link going down. In such a case, BFD informs all its registered clients about the failure. The clients then take failover action.
4. The BFD control packets are exchanged until the BFD session is terminated due to the removal of a BFD client or the removal of the BFD configuration. In either case, the BFD session is torn down.

For more information on how to configure BFD, refer to the next sections in this chapter.

Configuring a BFD parameter template

AT&T provides a BFD parameter template that you can use to configure BFD for each client. You can create multiple BFD parameter templates and associate a BFD parameter template with a client to configure fall-over BFD.

The following list presents some important guidelines for configuring and assigning the BFD parameter template:

- Creating and associating the BFD parameter template is optional. If you do not specify a BFD parameter template while registering a client for BFD, the default values for minimum-tx, minimum-rx, and detect-multiplier are used. Authentication is disabled by default.

  Note: Creating and associating the BFD parameter template is mandatory for static route clients.

- You can associate the BFD parameter template with either a source and destination address template or an interface. As the BFD template can be associated with either a source and destination address template or an interface, note the following priority levels:
  - If the BFD parameter template is associated with a source and destination address template, it has first priority.
  - If the BFD parameter template is associated with an interface, it has second priority.

  Note: Associating a BFD parameter template with an interface is supported for single-hop BFD sessions only.

- Instead of specifying the source IP address in the BFD configuration command, you can also use the `source any` parameter to associate the BFD parameter template with all BFD sessions that have the specific destination. For more information, refer to BFD Commands *(page 67)*.
- For OSPFv3 configurations, you must specify the link local addresses for source and destination in the BFD parameter template association command instead of the interface addresses or loopback addresses. For more information, refer to an example in Configuring BFD for OSPFv3 on a physical interface by using IPv6 addressing *(page 55)*.
- If the parameter template is not specified, minimum-tx is taken as 300, minimum-rx is taken as 300, and the multiplier is taken as 3.

Configuring with a source and destination address template

You can configure the BFD parameter template with the source and destination address template for a BFD session.

Consider two systems, R1 and R2, that already share a routing protocol such as BGP on a data plane interface named `dp0s7`. The following table provides a list of steps to configure the BFD parameter template and associate it with a source and destination address template on the R1 router.

Figure 3: Configuring with a source and destination address template

Table 1: Configuring with a source and destination address template

Router	Step	Command
R1	Specify a BFD parameter template name.	`vyatta@R1# set protocols bfd template test`
R1	Set the minimum-tx value.	`vyatta@R1# set protocols bfd template test minimum-tx 300`
R1	Set the minimum-rx value	`vyatta@R1# set protocols bfd template test minimum-rx 300`
R1	Set the detect-multiplier value.	`vyatta@R1# set protocols bfd template test multiplier 3`
R1	Set the authentication type and associate a key with the authentication type.	`vyatta@R1# set protocols bfd template test auth simple key brocade`
R1	Commit the configuration.	`vyatta@R1# commit`
R1	Save the configuration.	`vyatta@R1# save`

R1	Display the values of the BFD parameter template.	`vyatta@R1# show protocols`
		`bfd`
		`bfd {`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx`
		`300`
		`minimum-tx`
		`300`
		`multiplier 3`
		`}`
		`}`

R1	Assign the template for all BFD sessions with the destination as R2 and the source as R1.	`vyatta@R1# set protocols bfd destination 10.10.10.2 source 10.10.10.1 template test`

Router	Step	Command
R1	Commit the configuration.	`vyatta@R1# commit`
R1	Save the configuration.	`vyatta@R1# save`

R1	Display the configuration.	`vyatta@R1# show protocols`
		` bfd`
		`bfd {`
		`destination 10.10.10.2 {`
		`source 10.10.10.1`
		` {`
		` template test`
		`}`
		`}`
		`template test {`
		`auth {`
		` simple {`
		`key "********"`
		`}`
		`minimum-rx 300`
		`minimum-tx 300`
		`multiplier 3`
		`}`
		` }

Configuring with an interface

You can configure the BFD parameter template with the interface on which the BFD session is running. This method applies to single-hop BFD sessions only.

Consider two systems, R1 and R2, that already share a routing protocol such as BGP on a data plane interface named `dp0s7`. The following table provides a list of steps to associate a BFD template with an interface on the R1 router.

Figure 4: Configuring with an interface

Table 2: Configuring with an interface

Router	Step	Command
R1	Specify a BFD parameter template name.	`vyatta@R1# set protocols bfd template test`
R1	Set the minimum-tx value.	`vyatta@R1# set protocols bfd template test minimum-tx 300`
R1	Set the minimum-rx value	`vyatta@R1# set protocols bfd template test minimum-rx 300`

Router	Step	Command
R1	Set the detect-multiplier value.	`vyatta@R1# set protocols bfd template test multiplier 3`
R1	Set the authentication type and associate a key with the authentication type.	`vyatta@R1# set protocols bfd template test auth simple key brocade`
R1	Commit the configuration.	`vyatta@R1# commit`
R1	Save the configuration.	`vyatta@R1# save`
R1	Display the values of the BFD parameter template.	`vyatta@R1# show protocols bfd bfd { template test { auth { simple { key "********" } } minimum-rx 300 minimum-tx 300 multiplier 3 } }`
R1	Assign the template for all BFD sessions using dp0s7 as out-going interface.	`vyatta@R1# set interface dataplane dp0s7 bfd template test`
R1	Commit the configuration.	`vyatta@R1# commit`
R1	Save the configuration.	`vyatta@R1# save`
R1	Display the configuration.	`vyatta@R1# show interface dp0s7 interface { dataplane dp0s7 { template test { auth { simple { key "********" } } minimum-rx 300 minimum-tx 300 multiplier 3 } } }`

Configuring BFD by Using IPv4 Addressing

Configuring BFD for BGP

This section describes the procedure for configuring BFD for BGP single hop and multiple hop, so that BGP is a registered protocol with BFD and receives detection-failure messages about the forwarding path. Implementing BFD for BGP results in fast convergence timings.

Note: BFD is supported for both iBGP and eBGP. All these configurations are for BFD over iBGP. For BFD over eBGP, remote-as (asn) can be modified with remote system (asn) for both single-hop and multiple-hop BFD. The eBGP multiple-hop (hopcount) is configured with the required hop count for eBGP mutiple-hop BFD.

Note: BFD for BGP is supported for both IPv4 and IPv6 addressing.

Configuring BFD for BGP single hop by using IPv4 addressing

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share a BGP session, as illustrated in the reference network diagram. The following list provides the addresses of R1 and R2.

- R1 loopback address—`1.1.1.1`
- R1 interface address facing R2—`10.10.10.1`
- R2 loopback address—`2.2.2.2`
- R2 interface address facing R1—`10.10.10.2`
- Data plane interface name—`dp0s7`

Figure 5: Configuring BFD for BGP single hop by using IPv4 addressing

To configure BFD for BGP single hop by using IPv4 addressing, perform the following steps in configuration mode.

Table 3: Configuring BFD for BGP Single Hop by Using IPv4 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.

Router	Step	Command
R1	Associate the BFD `test` template with the destination address of R2 and the source address of R1. **Note:** You can also associate the BFD parameter template with the interface by using the `set interface dataplane <if_name> bfd template <template_name>` command.	`vyatta@R1#set protocols bfd destination 10.10.10.2 source 10.10.10.1 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set protocols bgp 100 neighbor 10.10.10.2 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show protocols`
		`bgp`
		`bgp 100 {`
		`neighbor 10.10.10.2`
		`{`
		`address-`
		`family {`
		`ipv4-unicast`
		`}`
		`fall-over {`
		`bfd`
		`}`
		`remote-as`
		`100`
		`}`
		`}`
		
		`vyatta@R1#show protocols`
		`bfd`
		`bfd {`
		`destination`
		`10.10.10.2 {`
		`source`
		`10.10.10.1 {`
		`template test`
		`}`
		`}`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx 300`
		`minimum-tx 300`
		`multiplier 3`
		`}`
		`}`

R2	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.

R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2.
	Note: You can also associate the BFD parameter template with the interface by using the `set interface dataplane <if_name> bfd template <template_name>` command.
		`vyatta@R2#set protocols bfd`
		`destination 10.10.10.1`
		`source 10.10.10.2 template`
		`test`

Router	Step	Command
R2	Register R1 as a BFD neighbor.	`vyatta@R2#set protocols bgp 100 neighbor 10.10.10.1 fall-over bfd`
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

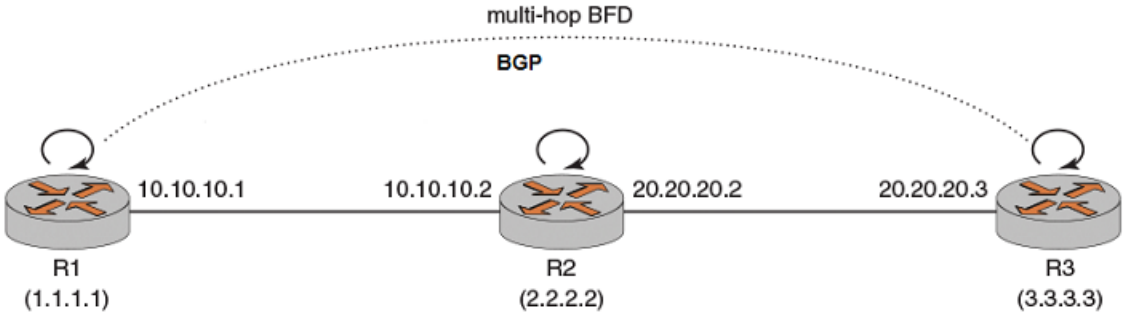
R2	Display the configuration.	`vyatta@R2#show protocols bgp bgp 100 { neighbor 10.10.10.1 { address-family { ipv4-unicast } fall-over { bfd } remote-as 100 } }`
		
		`vyatta@R2#show protocols bfd bfd { destination 10.10.10.1 { source 10.10.10.2 { template test } } template test { auth { simple { key "********" } } minimum-rx 300 minimum-tx 300 multiplier 3 } }`

Configuring BFD for BGP multiple hop by using IPv4 addressing

Consider a scenario in which you have three systems, R1, R2, and R3. R1 and R3 share a BGP session. The following list provides the addresses of R1, R2, and R3.

- R1 loopback address—`1.1.1.1`
- R1 interface address facing R2—`10.10.10.1`
- R2 loopback address—`2.2.2.2`
- R2 interface address facing R1—`10.10.10.2`
- R2 interface address facing R3—`20.20.20.2`
- R3 interface address facing R2—`20.20.20.3`
- R3 loopback address—`3.3.3.3`

Figure 6: Configuring BFD for BGP multiple hop by using IPv4 addressing



To configure a multiple-hop BFD session between R1 and R3, perform the following steps in configuration mode.

Table 4: Configuring BFD for BGP Multiple Hop by Using IPv4 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R3 and the source address of R1.	`vyatta@R1#set protocols bfd destination 20.20.20.3 source 10.10.10.1 template test`
R1	Register R3 as a BFD neighbor.	`vyatta@R1#set protocols bgp 100 neighbor 20.20.20.3 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show protocols bgp`
		`bgp 100 {`
		`neighbor 20.20.20.3 {`
		`fall-over {`
		`bfd`
		`}`
		`remote-as 100`
		`}`
		`}`
		
		`vyatta@R1#show protocols`
		`bfd`
		`bfd {`
		`destination 20.20.20.3 {`
		`source 10.10.10.1 {`
		
		`template test`
		`}`
		`}`
		`template test {`
		`auth {`
		
		`simple {`
		
		`key "********"`
		`}`
		`}`
		
		`minimum-rx 300`
		
		`minimum-tx 300`
		
		`multiplier 3`
		`}`
		`}`

R3	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R3	Associate the BFD `test` template with the destination address of R1 and the source address of R3.	`vyatta@R3#set protocols bfd destination 10.10.10.1 source 20.20.20.3 template test`
R3	Register R1 as a BFD neighbor.	`vyatta@R3#set protocols bgp 100 neighbor 10.10.10.1 fall-over bfd`
R3	Commit the configuration.	`vyatta@R3#commit`
R3	Save the configuration.	`vyatta@R3#save`

Router	Step	Command

R3	Display the configuration.	vyatta@R3#show protocols bgp
		bgp 100 {
		neighbor 10.10.10.1
		{
		fall-over {
		bfd
		}
		remote-as
		100
		}
		}
		
		vyatta@R3#show protocols
		bfd
		bfd {
		destination
		10.10.10.1 {
		source
		20.20.20.3 {
		
		template test
		}
		}
		template test {
		auth {
		
		simple {
		
		key "********"
		}
		}
		
		minimum-rx 300
		
		minimum-tx 300
		
		multiplier 3
		}
		}

Configuring BFD for static routes

To configure BFD for static routes by using IPv4 and IPv6 addressing, you must first set up the static route between the two peer systems.

If a static route exists between two nodes, BFD must be configured on both nodes. The static route is enabled and made available for use only when BFD is in the up state; otherwise, the static route is inactive. BFD for static routes is available for both single hop and multiple hops.

No keyword exists in the Vyatta CLI to configure BFD multiple-hop sessions, although multiple-hop sessions are supported. A BFD session is set up either in single-hop or multiple-hop mode based on the next-hop reachability from the source system. If the next hop for the destination is directly connected, BFD comes up as a single-hop session, and, if it is recursively reachable, BFD is set up as a multiple-hop session.

Configuring BFD for static routes single hop by using IPv4 addressing

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share a static route session, as illustrated in the reference network diagram. The following list provides the addresses of R1 and R2.

- R1 loopback address—`1.1.1.1/32`
- R1 interface address—`10.10.10.1/24`
- R2 loopback address—`2.2.2.2/32`
- R2 interface address—`10.10.10.2/24`

Figure 7: Configuring BFD for static routes single hop by using IPv4 addressing

To configure a BFD session between R1 and R2, perform the following steps in configuration mode.

Table 5: Configuring BFD for Static Routes Single Hop by Using IPv4 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R2 and the source address of R1. **Note:** Configuring session template is mandatory.	`vyatta@R1#set protocols bfd destination 10.10.10.2 source 10.10.10.1 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set protocols static route 2.2.2.2/32 next-hop 10.10.10.2 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show protocols`
		` static`
		`static {`
		`  route 2.2.2.2/32 {`
		`  next-hop`
		` 10.10.10.2 {`
		`  fall-over {`
		`  bfd`
		`  }`
		`  }`
		`  }`
		`}`
		
		`vyatta@R1#show protocols`
		` bfd`
		` bfd {`
		`  destination`
		` 10.10.10.2 {`
		`  source`
		` 10.10.10.1 {`
		` template test`
		`  }`
		`  }`
		`  template test {`
		`  auth {`
		` simple {`
		`  key "********"`
		`  }`
		`  }`
		` minimum-rx 300`
		` minimum-tx 300`
		` multiplier 3`
		`  }`
		` }

R2	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.

R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2.	`vyatta@R2#set protocols bfd`
		` destination 10.10.10.1`
		` source 10.10.10.2 template`
		` test`

R2	Register R1 as a BFD neighbor.	`vyatta@R2#set protocols`
		` static route 1.1.1.1/32`
		` next-hop 10.10.10.1 fall-`
		`over bfd`

Router	Step	Command
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

R2	Display the configuration.	`vyatta@R2#show protocols`
		` static`
		`static {`
		`  route`
		` 1.1.1.1/32 {`
		`  next-hop`
		` 10.10.10.1 {`
		` fall-over {`
		`  bfd`
		`  }`
		`  }`
		`  }`
		`}`

		`vyatta@R2#show protocols`
		` bfd`
		` bfd {`
		`  destination`
		` 10.10.10.1 {`
		`  source`
		` 10.10.10.2 {`
		` template test`
		`  }`
		`  }`
		`  template test {`
		`  auth {`
		` simple {`
		`  key "********"`
		`  }`
		` minimum-rx 300`
		` minimum-tx 300`
		` multiplier 3`
		`  }`
		`}`

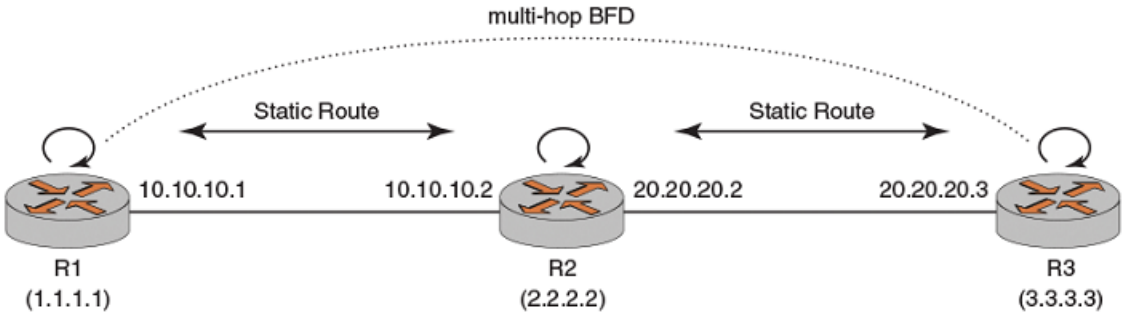
Configuring BFD for static routes multiple hop by using IPv4 addressing

To configure BFD multiple hop for static routes, you must first set up the static route between the two peer systems.

Consider a scenario in which you have three systems R1, R2, and R3. R1 and R2 share a static route, and R2 and R3 share a static route. The following list provides the addresses of R1, R2, and R3.

- R1 loopback address—`1.1.1.1/32`
- R1 interface address facing R2—`10.10.10.1/24`
- R2 loopback address—`2.2.2.2/32`
- R2 interface address facing R1—`10.10.10.2/24`
- R2 interface address facing R3—`20.20.20.2/24`
- R3 interface address facing R2—`20.20.20.3/24`
- R3 loopback address—`3.3.3.3/32`

Figure 8: Configuring BFD for static routes multiple hop by using IPv4 addressing



To configure a BFD session between R1 and R3, perform the following steps in configuration mode.

Table 6: Configuring BFD for Static Routes Multiple Hop by Using IPv4 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R3 and the source address of R1. **Note:** Configuring session template is mandatory.	`vyatta@R1# set protocols bfd destination 20.20.20.3 source 10.10.10.1 template test`
R1	Register R3 as a BFD neighbor.	`vyatta@R1# set protocols static route 3.3.3.3/32 next-hop 20.20.20.3 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1# commit`
R1	Save the configuration.	`vyatta@R1# save`

Router	Step	Command

R1	Display the configuration.	vyatta@R1# show protocols
		static
		static {
		route 3.3.3.3/32 {
		next-hop
		20.20.20.3 {
		fall-
		over {
		
		bfd
		}
		}
		}
		}
		
		vyatta@R1# show protocols
		bfd
		bfd {
		destination
		20.20.20.3 {
		source
		10.10.10.1 {
		
		template test
		}
		}
		template test {
		auth {
		
		simple {
		
		key "********"
		}
		}
		minimum-rx
		300
		minimum-tx
		300
		multiplier 3
		}
		}

R3	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R3	Associate the BFD `test` template with the destination address of R1 and the source address of R3.	vyatta@R3# set protocols bfd destination 10.10.10.1 source 20.20.20.3 template test
R3	Register R1 as a BFD neighbor.	vyatta@R3# set protocols static route 1.1.1.1/32 next-hop 10.10.10.1 fall- over bfd
R3	Commit the configuration.	vyatta@R3# commit

Router	Step	Command
R3	Save the configuration.	`vyatta@R3# save`

R3	Display the configuration.	`vyatta@R3# show protocols`
		` static`
		`static {`
		`route`
		` 1.1.1.1/32 {`
		`next-hop`
		` 10.10.10.1 {`
		` fall-over {`
		`bfd`
		`}`
		`}`
		`}`
		`}`
		
		`vyatta@R3# show protocols`
		` bfd`
		` bfd {`
		`destination`
		` 10.10.10.1 {`
		`source`
		` 20.20.20.3 {`
		` template test`
		`}`
		`}`
		`template test {`
		`auth {`
		` simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx`
		` 300`
		`minimum-tx`
		` 300`
		`multiplier 3`
		`}`
		`}`

Configuring a BFD helper session by using IPv4 addressing

You can configure a static route from a router R1 to another router R2, without configuring another static route from R2 to R1. In such a case, you can enable BFD only from R1 to R2 with static route as the client. BFD must be configured at both routers for it to be operational, you can use a helper session to compensate the lack of a static route from R2 to R1.

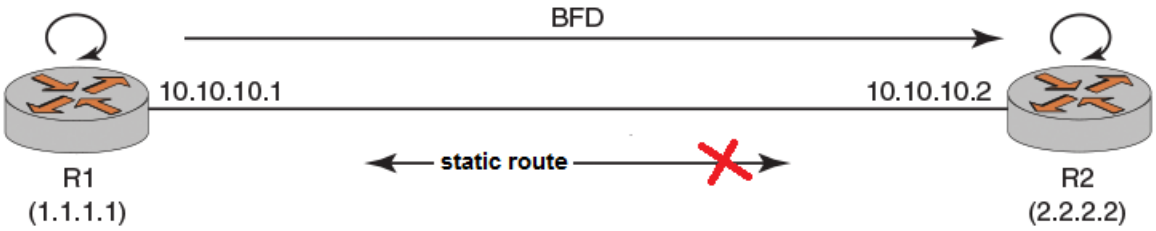
Consider two routers R1 and R2. There is a static route from R2 to R1, and but no static route from R1 to R2. You can enable BFD for R1.

Note: The `source any` parameter is not supported for the helper session command.

The helper session is supported for both IPv4 and IPv6 addresses. The following list provides the addresses of R1 and R2.

- R1 interface address facing R2—`10.10.10.1`
- R2 interface address facing R1—`10.10.10.2`

Figure 9: Configuring a BFD helper session by using IPV4 addressing



To configure BFD by using a helper session, perform the following steps in configuration mode.

Note: Instead of specifying the source IP address in this example, you can also use the `source any` parameter to associate the BFD template with all BFD sessions that have the specific destination. For more information, refer to BFD Commands *(page 67)*.

Table 7: Configuring a BFD Helper Session by Using IPV4 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R2 and the source address of R1.	`vyatta@R1#set protocols bfd destination 10.10.10.2 source 10.10.10.1 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set protocols bfd destination 10.10.10.2 source 10.10.10.1 helper-session`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	vyatta@R1#show protocols
		bfd
		protocols {
		bfd
		{
		destination 10.10.10.2 {
		source 10.10.10.1 {
		template test
		helper-session
		}
		}
		minimum-rx 300
		minimum-tx 300
		multiplier 3
		}
		}

Configuring BFD for OSPFv2

BFD for OSPFv2 is supported for physical interfaces, virtual interfaces, and virtual links. OSPFv2 uses IPv4 addressing.

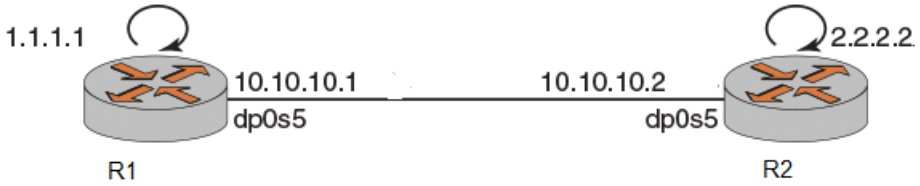
Configuring BFD for OSPFv2 on a physical interface by using IPv4 addressing

To configure BFD for all OSPFv2 neighbors on a physical interface, you must first configure OSPFv2 for all the neighbors.

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share an OSPFv2 session, as illustrated in the reference network diagram. The following list provides the addresses of R1 and R2.

- R1 loopback address—`1.1.1.1`
- R1 interface address—`10.10.10.1`
- R2 loopback address—`2.2.2.2`
- R2 interface address—`10.10.10.2`.
- R1 and R2 connected physical interface—`dp0s5`

Figure 10: Configuring BFD for OSPFv2 on a physical interface by using IPv4 addressing



To configure a BFD session between R1 and R2, perform the following steps in configuration mode. BFD for OSPFv2 is configured on the physical interface.

Table 8: Configuring BFD for OSPFv2 on a Physical Interface by Using IPv4 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the interface	`vyatta@R1#set interface dataplane dp0s5 bfd template test`
R1	Start a BFD instance on the interface.	`vyatta@R1# set interface dataplane dp0s5 ip ospf fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show interfaces`
		`interfaces {`
		`dataplane dp0s5 {`
		`address dhcp`
		`}`
		`dataplane dp0s5 {`
		`address 10.10.10.1`
		`ip {`
		`ospf {`
		`fall-over {`
		`bfd`
		`}`
		`}`
		`}`
		`}`
		
		`vyatta@R1#show protocols`
		`bfd`
		`bfd {`
		`destination 10.10.10.2 {`
		`source 10.10.10.1 {`
		`template test`
		`}`
		`}`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx 300`
		`minimum-tx 300`
		`multiplier 3`
		`}`
		`}`

R2	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.

Router	Step	Command
R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2. **Note:** You can also associate the BFD parameter template with the interface by using the `set interface dataplane <if_name> bfd template <template_name>` command.	`vyatta@R2#set protocols bfd destination 10.10.10.1 source 10.10.10.2 template test`
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

Router	Step	Command

R2	Display the configuration.	`vyatta@R2#show interfaces`
		`interfaces {`
		`dataplane dp0s5 {`
		`address dhcp`
		`}`
		`dataplane dp0s5 {`
		`address 10.10.10.2`
		`ip {`
		`ospf {`
		`fall-over {`
		`bfd`
		`}`
		`}`
		`}`
		`}`
		
		`vyatta@R2#show protocols`
		`bfd`
		`bfd {`
		`destination 10.10.10.1 {`
		`source 10.10.10.2 {`
		`template test`
		`}`
		`}`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx 300`
		`minimum-tx 300`
		`multiplier 3`
		`}`
		`}`

Configuring BFD for OSPFv2 on a virtual link by using IPv4 addressing

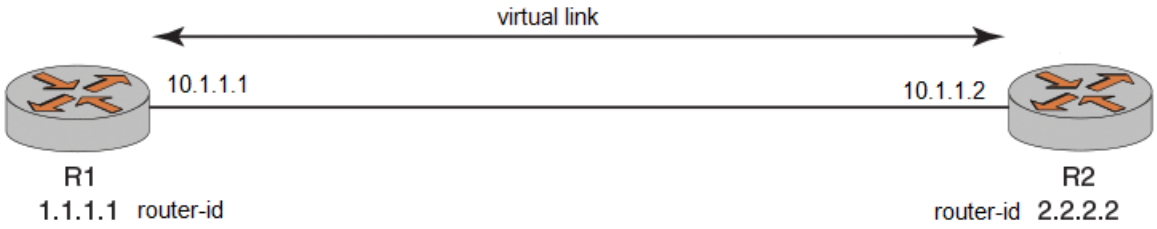
To configure BFD for OSPFv2 neighbors on a virtual link, you must first configure the virtual link between the disconnected backbone area routers and then enable BFD on the virtual link.

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share an OSPFv2 session, sharing a virtual link, as illustrated in the reference network diagram. The following list provides the addresses of R1 and R2.

- R1 router-id—`1.1.1.1`
- R2 router-id—`2.2.2.2`

- R1 interface address facing R2—`10.1.1.1`
- R2 interface address facing R1—`10.1.1.2`

Figure 11: Configuring BFD for OSPFv2 on a virtual link by using IPv4 addressing



Virtual link is configured in a transit area. A virtual link chooses any address to reach other virtual link end points in the same transit area. Therefore, the source and destination addresses in a three-router configuration or a more-complex configuration are selected dynamically. You must ensure that you select the correct source and destination addresses for your BFD commands for OSPFv2 and OSPFv3 virtual links. To configure a BFD session between R1 and R2, perform the following steps in configuration mode.

Table 9: Configuring BFD for OSPFv2 on a Virtual Link by Using IPv4 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R2 and the source address of R1.	`vyatta@R1#set protocols bfd destination 10.1.1.2 source 10.1.1.1 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set protocols ospf area 1 virtual-link 2.2.2.2 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show protocols ospf ospf { area 1 { network 10.1.1.0/24 virtual-link 2.2.2.2 { fall-over { bfd } } } parameters { router-id 1.1.1.1 } }`
		
		`vyatta@R1#show protocols bfd bfd { destination 10.1.1.2 { source 10.1.1.1 { template test } } template test { auth { simple { key "********" } } minimum-rx 300 minimum-tx 300 multiplier 3 } }`

R2	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2.	`vyatta@R2#set protocols bfd destination 10.1.1.1 source 10.1.1.2 template test`
R2	Register R1 as a BFD neighbor.	`vyatta@R2#set protocols ospf area 1 virtual-link 1.1.1.1 fall-over bfd`

Router	Step	Command
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

R2	Display the configuration.	`vyatta@R2#show protocols`
		` ospf`
		` ospf {`
		`  area 1 {`
		`  network`
		` 10.1.1.0/24`
		`  virtual-link`
		` 1.1.1.1 {`
		` fall-over {`
		`  bfd`
		`  }`
		`  }`
		`  parameters {`
		`  router-id`
		` 2.2.2.2`
		`  }`
		` }`
		
		`vyatta@R2#show protocols`
		` bfd`
		`  bfd {`
		`  destination 10.1.1.1`
		` {`
		`  source`
		` 10.1.1.2{`
		`  template`
		` test`
		`  }`
		`  }`
		`  template test {`
		`  auth {`
		` simple {`
		`  key "********"`
		`  }`
		`  }`
		`  minimum-rx`
		` 300`
		`  minimum-tx`
		` 300`
		`  multiplier 3`
		`  }`
		` }`

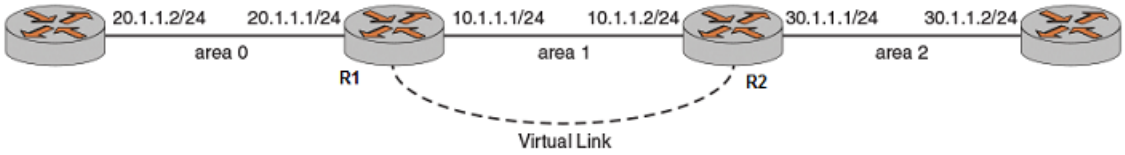
Configuring BFD for OSPFv2 on a virtual interface by using IPv4 addressing

To configure BFD for OSPFv2 neighbors on a virtual interface (VIF), you must first configure the OSPF and VIF for the two neighbors.

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share an OSPFv2 session. R1 and R2 are on a physical interface named dp0s5 which also has a virtual interface configured for the VLAN identifier of `vlan-51`. The following list provides the addresses of R1 and R2.

- R1 loopback address—`1.1.1.1/32`
- Physical data plane address—`dp0s5`
- Virtual link between R1 and R2—`vif 51`
- VLAN identifier—`vlan-51`
- R1 interface facing vif 51—`10.1.1.1/24`
- R2 interface facing vif 51—`10.1.1.2/24`
- R2 loopback address—`2.2.2.2/32`

Figure 12: Configuring BFD for OSPFv2 on a virtual interface by using IPv4 addressing



To configure a BFD session between R1 and R2, perform the following steps in configuration mode.

Table 10: Configuring BFD for OSPFv2 on a Virtual Interface by Using IPv4 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the interface	`vyatta@R1#set interface dataplane vif 51 bfd template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set interfaces dataplane dp0s4 vif 51 ip ospf fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	vyatta@R1#show interfaces
		interfaces {
		vif 51 {
		address
		10.1.1.1/24
		ip {
		ospf {
		fall-over {
		bfd
		}
		}
		}
		vlan
		51
		}
		}
		loopback lo {
		address
		1.1.1.1/32
		}
		
		vyatta@R1#show protocols
		protocols {
		bfd {
		destination
		10.1.1.2 {
		source 10.1.1.1 {
		template test
		}
		}
		template
		test{
		auth {
		simple {
		key "*******"
		}
		}
		minimum-rx
		300
		minimum-tx
		300
		multiplier 3
		}
		ospf {
		area 0 {
		network 30.30.30.0/24
		}
		redistribute
		{
		connected
		}
		}
		}

Router	Step	Command
R2	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2. **Note:** You can also associate the BFD parameter template with the interface by using the `set interface dataplane vif <vf_id> bfd template <template-name>` command.	`vyatta@R2#set protocols bfd destination 10.1.1.1 source 10.1.1.2 template test`
R2	Register R1 as a BFD neighbor.	`vyatta@R2#set interfaces dataplane dp0s4 vif 51 ip ospf fall-over bfd`
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

Router	Step	Command

R2	Display the configuration.	vyatta@R2#show interfaces
		interfaces {
		vif 51 {
		address
		10.1.1.2/24
		ip {
		ospf {
		fall-over {
		bfd
		}
		}
		}
		vlan 51
		}
		}
		loopback lo {
		address
		2.2.2.2/32
		}
		
		vyatta@R2#show protocols
		protocols {
		bfd {
		destination
		10.1.1.1 {
		source 10.1.1.2 {
		template test
		}
		}
		template
		test{
		auth {
		simple {
		key "*******"
		}
		}
		minimum-rx
		300
		minimum-tx
		300
		multiplier 3
		}
		ospf {
		area 0 {
		network 30.30.30.0/24
		}
		redistribute
		{
		connected
		}
		}
		}

Configuring BFD by Using IPv6 Addressing

Configuring BFD for BGP

This section describes the procedure for configuring BFD for BGP single hop and multiple hop, so that BGP is a registered protocol with BFD and receives detection-failure messages about the forwarding path. Implementing BFD for BGP results in fast convergence timings.

Note: BFD is supported for both iBGP and eBGP. All these configurations are for BFD over iBGP. For BFD over eBGP, remote-as (asn) can be modified with remote system (asn) for both single-hop and multiple-hop BFD. The eBGP multiple-hop (hopcount) is configured with the required hop count for eBGP mutiple-hop BFD.

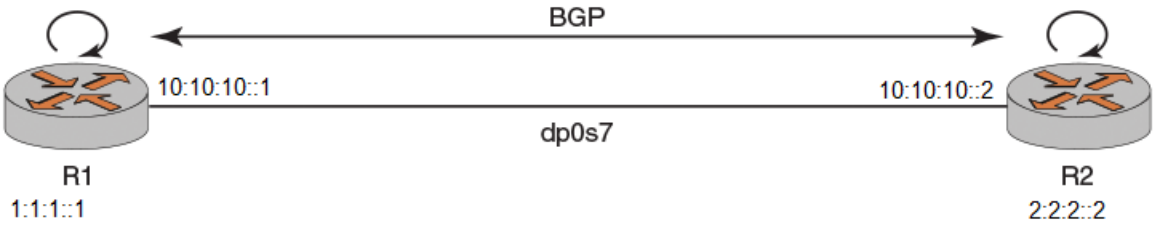
Note: BFD for BGP is supported for both IPv4 and IPv6 addressing.

Configuring BFD for BGP single hop by using IPv6 addressing

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share a BGP session, as illustrated in the reference network diagram. The following list provides the addresses of R1 and R2.

- R1 loopback address—`1:1:1::1`
- R1 interface address facing R2—`10:10:10::1`
- R2 loopback address—`2:2:2::2`
- R2 interface address facing R1—`10:10:10::2`
- Data plane interface name—`dp0s7`

Figure 13: Configuring BFD for BGP single hop by using IPv6 addressing



To configure BFD for BGP single hop by using IPv6 addressing, perform the following steps in configuration mode.

Table 11: Configuring BFD for BGP Single Hop by Using IPv6 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.

Router	Step	Command
R1	Associate the BFD `test` template with the destination address of R2 and the source address of R1. **Note:** You can also associate the BFD parameter template with the interface by using the `set interface dataplane <if_name> bfd template <template_name>` command.	`vyatta@R1#set protocols bfd destination 10:10:10::2 source 10:10:10::1 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set protocols bgp 100 neighbor 10:10:10::2 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show protocols bgp bgp 100 { neighbor 10:10:10::2 { address-family { ipv6-unicast } fall-over { bfd } remote-as 100 } }`
		
		`vyatta@R1#show protocols bfd bfd { destination 10:10:10::2 { source 10:10:10::1 { template test } } template test { auth { simple { key "********" } } minimum-rx 300 minimum-tx 300 multiplier 3 } }`

R2	Create a BFD template called `test`.	See section Configuring the BFD template.
R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2. **Note:** You can also associate the BFD parameter template with the interface by using the `set interface dataplane <if_name> bfd template <template_name>` command.	`vyatta@R2#set protocols bfd destination 10:10:10::1 source 10:10:10::2 template test`

Router	Step	Command
R2	Register R1 as a BFD neighbor.	`vyatta@R2#set protocols bgp 100 neighbor 10:10:10::1 fall-over bfd`
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

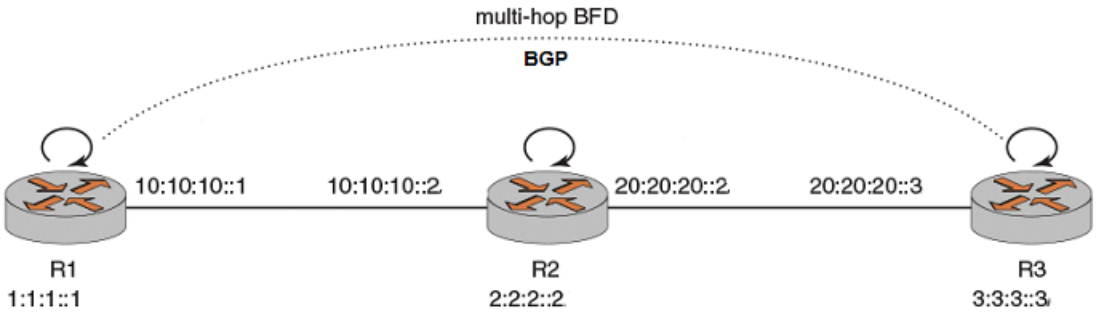
R2	Display the configuration.	`vyatta@R2# show protocols`
		`bgp`
		`bgp 100 {`
		`neighbor 10:10:10::1 {`
		`address-family {`
		`ipv6-unicast`
		`}`
		`fall-over {`
		`bfd`
		`}`
		`remote-as 100`
		`}`
		`}`
		
		`vyatta@R2#show protocols`
		`bfd`
		`bfd {`
		`destination 10:10:10::1 {`
		`source 10:10:10::2 {`
		`template test`
		`}`
		`}`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx 300`
		`minimum-tx 300`
		`multiplier 3`
		`}`
		`}`

Configuring BFD for BGP multiple hop by using IPv6 addressing

Consider a scenario in which you have three systems, R1, R2, and R3. R1 and R3 share a multiple-hop BGP session, as illustrated in the reference network diagram. The following list provides the addresses of R1, R2, and R3.

- R1 loopback address—`1:1:1::1`
- R1 interface address facing R2—`10:10:10::1`
- R2 loopback address—`2:2:2::2`
- R2 interface address facing R1—`10:10:10::2`
- R2 interface address facing R3—`20:20:20::2`
- R3 interface address facing R2—`20:20:20::3`
- R3 loopback address—`3:3:3::3`

Figure 14: Configuring BFD for BGP multiple hop by using IPv6 addressing



To configure a multiple-hop BFD session between R1 and R3, perform the following steps in configuration mode.

Table 12: Configuring BFD for BGP Multiple Hop by Using IPv6 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R3 and the source address of R1.	`vyatta@R1#set protocols bfd destination 20:20:20::3 source 10:10:10::1 template test`
R1	Register R3 as a BFD neighbor.	`vyatta@R1#set protocols bgp 100 neighbor 20:20:20::3 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show protocols bgp`
		`bgp 100 {`
		`neighbor 20:20:20::3 {`
		`fall-over {`
		`bfd`
		`}`
		`remote-as 100`
		`}`
		`}`
		
		`vyatta@R1#show protocols`
		`bfd`
		`bfd {`
		`destination 20:20:20::3 {`
		`source 10:10:10::1 {`
		`template test`
		`}`
		`}`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx 300`
		`minimum-tx 300`
		`multiplier 3`
		`}`
		`}`

R3	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R3	Associate the BFD `test` template with the destination address of R1 and the source address of R3.	`vyatta@R3#set protocols bfd destination 10:10:10::1 source 20:20:20::3 template test`
R3	Register R1 as a BFD neighbor.	`vyatta@R3#set protocols bgp 100 neighbor 10:10:10::1 fall-over bfd`
R3	Commit the configuration.	`vyatta@R3#commit`
R3	Save the configuration.	`vyatta@R3#save`

Router	Step	Command

R3	Display the configuration.	vyatta@R3#show protocols bgp
		bgp 100 {
		neighbor 10:10:10::1
		{
		fall-over {
		bfd
		}
		remote-as
		100
		}
		}
		
		vyatta@R3#show protocols
		bfd
		bfd {
		destination
		10:10:10::1 {
		source
		20:20:20::3 {
		
		template test
		}
		}
		template test {
		auth {
		
		simple {
		
		key "********"
		}
		}
		
		minimum-rx 300
		
		minimum-tx 300
		
		multiplier 3
		
		}
		}

Configuring BFD for static routes

To configure BFD for static routes by using IPv4 and IPv6 addressing, you must first set up the static route between the two peer systems.

If a static route exists between two nodes, BFD must be configured on both nodes. The static route is enabled and made available for use only when BFD is in the up state; otherwise, the static route is inactive. BFD for static routes is available for both single hop and multiple hops.

No keyword exists in the Vyatta CLI to configure BFD multiple-hop sessions, although multiple-hop sessions are supported. A BFD session is set up either in single-hop or multiple-hop mode based on the next-hop reachability from the source system. If the next hop for the destination is directly connected, BFD comes up as a single-hop session, and, if it is recursively reachable, BFD is set up as a multiple-hop session.

Configuring BFD for static route single hop by using IPv6 addressing

To configure BFD single hop by using IPv6 addressing, you must first set up a supported routing protocol between the two peer systems.

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share static route, as illustrated in the reference network diagram. The following list provides the addresses of R1 and R2.

- R1 loopback address—`1:1:1::1/128`
- R1 interface address—`10:10:10::1/64`
- R2 loopback address—`2:2:2::2/128`
- R2 interface address—`10:10:10::2/64`

Figure 15: Configuring BFD for static route single hop by using IPv6 addressing

To configure a BFD session between R1 and R2, perform the following steps in configuration mode.

Table 13: Configuring BFD for Static Route Single Hop by Using IPv6 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD Template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R1 and the source address of R2.	`vyatta@R1#set protocols bfd destination 10:10:10::2 source 10:10:10::1 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set protocols static route6 2:2:2::2/128 next-hop 10:10:10::2 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show protocols`
		` static route6`
		`static {`
		`  route6 2:2:2::2/128`
		` {`
		`  next-hop`
		` 10:10:10::2 {`
		`  fall-over {`
		`  bfd`
		`  }`
		`  }`
		`  }`
		`}`
		
		`vyatta@R1#show protocols`
		` bfd`
		` bfd {`
		`  destination`
		` 10:10:10::2 {`
		`  source`
		` 10:10:10::1 {`
		`  template test`
		`  }`
		`  }`
		`  template test {`
		`  auth {`
		`  simple {`
		`  key "********"`
		`  }`
		`  }`
		`  minimum-rx`
		` 300`
		`  minimum-tx`
		` 300`
		`  multiplier 3`
		`  }`
		` }`

R2	Create a BFD template called `test`.	See section Configuring the BFD Template *(page 11)*.

R2	Associate the BFD `test` template with the destination address of R2 and the source address of R1.	`vyatta@R2#set protocols bfd`
		` destination 10:10:10::1`
		` source 10:10:10::2 template`
		` test`

R2	Register R1 as a BFD neighbor.	`vyatta@R2#set protocols`
		` static route6 1:1:1::1/128`
		` next-hop 10:10:10::1 fall-`
		`over bfd`

R2	Commit the configuration.	`vyatta@R2#commit`

Router	Step	Command
R2	Save the configuration.	`vyatta@R2#save`

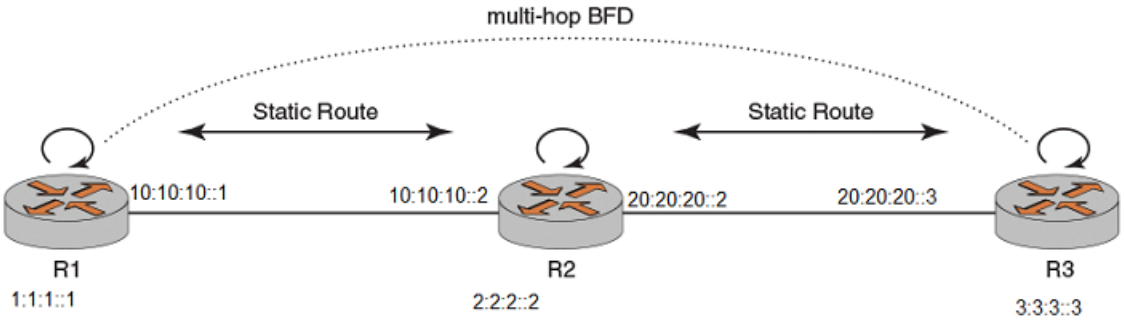
R2	Display the configuration.	`vyatta@R2#show protocols`
		` static route6`
		`static {`
		`  route6 1:1:1::1/128`
		` {`
		`  next-hop`
		` 10:10:10::1 {`
		` fall-over {`
		`  bfd`
		`  }`
		`  }`
		`}`

		`vyatta@R2#show protocols`
		` bfd`
		` bfd {`
		`  destination`
		` 10:10:10::1 {`
		`  source`
		` 10:10:10::2 {`
		` template test`
		`  }`
		`  }`
		`  template test {`
		`  auth {`
		` simple {`
		`  key "********"`
		`  }`
		`  }`
		`  minimum-rx`
		` 300`
		`  minimum-tx`
		` 300`
		`  multiplier 3`
		`  }`
		` }`

Configuring BFD for static route multiple hop by using IPv6 addressing

To configure BFD multiple hop by using IPv6 addressing, you must first set up the routing protocol such as static route or BGP between the two peer systems.

Figure 16: Configuring BFD for static route multiple hop by using IPv6 addressing



Consider a scenario in which you have three systems, R1, R2, and R3. R1 and R2 share a static route, R2 and R3 also share a static route. The following list provides the addresses of R1, R2, and R3.

- R1 loopback address—`1:1:1::1/128`
- R1 interface address facing R2—`10:10:10::1/64`
- R2 loopback address— `2:2:2::1/128`
- R2 interface address facing R1—`10:10:10::2/64`
- R2 interface address facing R3—`20:20:20::2/64`
- R3 interface address facing R2—`20:20:20::3/64`
- R3 loopback address—`3:3:3::3/128`

To configure a BFD session between R1 and R3, perform the following steps in configuration mode.

Table 14: Configuring BFD for Static Route Multiple Hop by Using IPv6 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R3 and the source address of R1.	`vyatta@R1#set protocols bfd destination 20:20:20::3 source 10:10:10::1 template test`
R1	Register R3 as a BFD neighbor.	`vyatta@R1#set protocols static route6 3:3:3::3/128 next-hop 20:20:20::3 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	vyatta@R1#show protocols
		static route6
		static {
		route6 3:3:3::3/128
		{
		next-hop
		20:20:20::3 {
		fall-over {
		bfd
		}
		}
		}
		}
		
		vyatta@R1#
		show protocols bfd
		bfd {
		destination
		20:20:20::3 {
		source
		10:10:10::1 {
		template test
		}
		}
		template test {
		auth {
		simple {
		key "********"
		}
		}
		minimum-rx 300
		minimum-tx 300
		multiplier 3
		}
		}

R3	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R3	Associate the BFD `test` template with the destination address of R1 and the source address of R3.	vyatta@R3#set protocols bfd destination 10:10:10::1 source 20:20:20::3 template test
R3	Register R1 as a BFD neighbor.	vyatta@R3#set protocols static route6 1:1:1::1/128 next-hop 10:10:10::1 fall- ver bfd

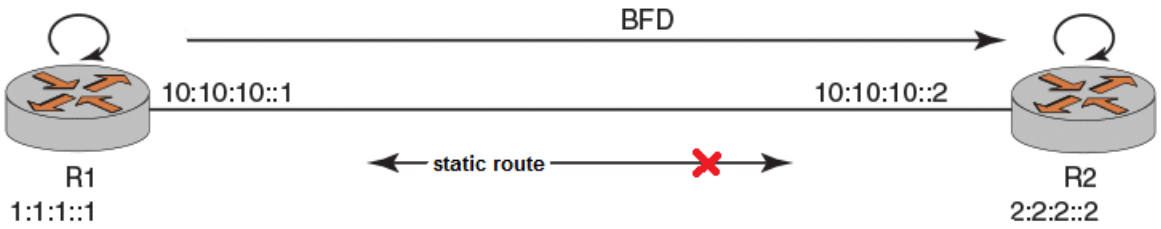
Router	Step	Command
R3	Commit the configuration.	`vyatta@R3#commit`
R3	Save the configuration.	`vyatta@R3#save`

R3	Display the configuration.	`vyatta@R3#show protocols`
		` static route6`
		`static {`
		`route6`
		` 1:1:1::1/128 {`
		`next-hop`
		` 10:10:10::1{`
		
		` fall-over {`
		
		`bfd`
		`}`
		`}`
		`}`
		`}`
		
		`vyatta@R3#`
		`show protocols bfd`
		` bfd {`
		`destination`
		` 10:10:10::1 {`
		`source`
		` 20:20:20::3 {`
		
		` template test`
		`}`
		`}`
		`}`
		`template test {`
		`auth {`
		
		` simple {`
		
		`key "********"`
		`}`
		`}`
		
		` minimum-rx 300`
		
		` minimum-tx 300`
		
		` multiplier 3`
		
		`}`
		` }

Configuring a BFD helper session by using IPv6 addressing

You can configure a static route from a router R1 to another router R2, without configuring another static route from R2 to R1. In such a case, you can enable BFD only from R1 to R2 with static route as the client. BFD must be configured at both routers for it to be operational, you can use a helper session to compensate the lack of a static route from R2 to R1.

Figure 17: Configuring a BFD helper session by using IPv6 addressing



Consider two routers R1 and R2. There is a static route from R2 to R1, and but no static route from R1 to R2.. You can enable BFD for R1.

Note: The `source any` parameter is not supported for the helper session command.

The helper session is supported for both IPv4 and IPv6 addresses. The following list provides the addresses of R1 and R2.

- R1 interface address facing R2—`10:10:10::1`
- R2 interface address facing R1—`10:10:10::2`

To configure BFD using a helper session, perform the following steps in configuration mode.

Table 15: Configuring a BFD Helper Session by Using IPV6 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R2 and the source address of R1.	`vyatta@R1#set protocols bfd destination 10:10:10::2 source 10:10:10::1 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set protocols bfd destination 10:10:10::2 source 10:10:10::1 helper-session`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	vyatta@R1#show protocols
		bfd
		protocols {
		bfd
		{
		destination 10:10:10::2 {
		source 10:10:10::1 {
		template test
		helper-session
		}
		}
		}
		}

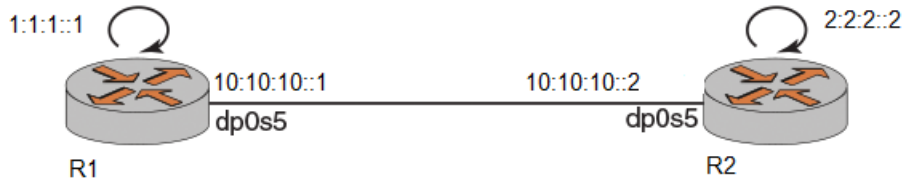
Configuring BFD for OSPFv3

BFD for OSPFv3 is supported for physical interfaces, virtual interfaces, and virtual links. OSPFv3 uses IPv6 addressing.

Configuring BFD for OSPFv3 on a physical interface by using IPv6 addressing

To configure BFD for all OSPFv3 neighbors on a physical interface, you must first configure OSPFv3 for all the neighbors.

Figure 18: Configuring BFD for OSPFv3 on a physical interface by using IPv6 addressing



Consider a scenario in which you have two systems, R1 and R2. R1 and R2 are OSPFv3 neighbors, as illustrated in the reference network diagram. The following list provides the addresses of R1 and R2.

- R1 loopback address—`1:1:1::1/128`
- R1 interface address—`10:10:10::1/64`
- R2 loopback address—`2:2:2::2/128`
- R2 interface address—`10:10:10::2/64`
- Data plane interface name—`dp0s5`
- R1 link local address—`fe80::6061:ff:fe00:b7d5`
- R2 link local address—`fe80::5054:ff:fe00:b6d5`

Note: For OSPFv3 configurations, you must use the link local addresses for R1 and R2 for source and destination.

To configure a BFD session between R1 and R2, perform the following steps in configuration mode. BFD for OSPFv3 is configured on the physical interface.

Table 16: Configuring BFD for OSPFv3 on a Physical Interface by Using IPv6 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the interface.	`vyatta@R1#set interface dataplane dp0s5 bfd template test`
R1	Start a BFD instance on the interface.	`vyatta@R1# set interface dataplane dp0s5 ipv6 ospfv3 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show interfaces`
		`interfaces {`
		`dataplane dp0s5 {`
		`address dhcp6`
		`}`
		`dataplane dp0s5 {`
		`address 10:10:10::1`
		`ipv6 {`
		`ospfv3 {`
		`fall-over {`
		`bfd`
		`}`
		`}`
		`}`
		`}`
		
		`vyatta@R1#show protocols`
		`bfd`
		`bfd {`
		`destination fe80::5054:ff:fe00:b6d5 {`
		`source fe80::6061:ff:fe00:b7d5 {`
		`template test`
		`}`
		`}`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx 300`
		`minimum-tx 300`
		`multiplier 3`
		`}`
		`}`

R2	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.

Router	Step	Command
R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2. **Note:** You can also associate the BFD parameter template with the interface by using the `set interface dataplane <if_name> bfd template <template_name>` command.	`vyatta@R2#set protocols bfd destination fe80::6061:ff:fe00:b7d5 source fe80::5054:ff:fe00:b6d5 template test`
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

Router	Step	Command

R2	Display the configuration.	`vyatta@R2#show interfaces`
		`interfaces {`
		`dataplane dp0s5 {`
		`address dhcp6`
		`}`
		`dataplane dp0s5 {`
		`address 10:10:10::2`
		`ipv6 {`
		`ospfv3 {`
		`fall-over {`
		`bfd`
		`}`
		`}`
		`}`
		`}`
		
		`vyatta@R2#show protocols`
		`bfd`
		`bfd {`
		`destination fe80::6061:ff:fe00:b7d5 {`
		`source fe80::5054:ff:fe00:b6d5 {`
		`template test`
		`}`
		`}`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx 300`
		`minimum-tx 300`
		`multiplier 3`
		`}`
		`}`

Configuring BFD for OSPFv3 on a virtual link by using IPv6 addressing

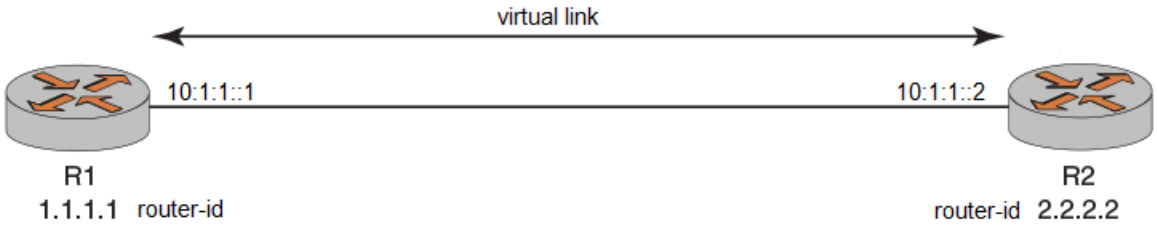
To configure BFD for OSPFv3 neighbors on a virtual link, you must first configure the virtual link between the disconnected backbone area routers and then enable BFD on the virtual link.

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share an OSPFv3 session, sharing a virtual link, as illustrated in the reference network diagram. The following list provides the addresses of R1 and R2.

- R1 router-id—`1.1.1.1`

- R2 router-id—`2.2.2.2`
- R1 interface address facing R2—`10:1:1::1`
- R2 interface address facing R1—`10:1:1::2`
- R1 link local address—`10:1:1::1`
- R2 link local address—`10:1:1::2`

Figure 19: Configuring BFD for OSPFv3 on a virtual link by using IPv6 addressing



Note: For OSPFv3 configurations, you must use the link local addresses for R1 and R2 for source and destination.

Virtual link is configured in a transit area. A virtual link chooses any address to reach other virtual link end points in the same transit area. Therefore, the source and destination addresses in a 3-router configuration or a more complex configuration are selected dynamically. You must ensure that you select the correct source and destination addresses for your BFD commands for OSPFv2 and OSPFv3 virtual links. To configure a BFD session between R1 and R2, perform the following steps in configuration mode.

Table 17: Configuring BFD for OSPFv3 on a Virtual Link by Using IPv6 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R2 and the source address of R1.	`vyatta@R1#set protocols bfd destination 10:1:1::2 source 10:1:1::1 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set protocols ospfv3 area 1 virtual-link 2.2.2.2 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	`vyatta@R1#show protocols`
		`ospf`
		`ospfv3 {`
		`area 1 {`
		`network`
		`10:1:1::0/128`
		`virtual-link`
		`2.2.2.2 {`
		`fall-over {`
		`bfd`
		`}`
		`}`
		`}`
		`parameters {`
		`}`
		`}`
		
		`vyatta@R1#show protocols`
		`bfd`
		`bfd {`
		`destination`
		`10:1:1::2 {`
		`source`
		`10:1:1::1 {`
		`template`
		`test`
		`}`
		`}`
		`template test {`
		`auth {`
		`simple {`
		`key "********"`
		`}`
		`}`
		`minimum-rx`
		`300`
		`minimum-tx`
		`300`
		`multiplier 3`
		`}`
		`}`

R2	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.

R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2.	`vyatta@R2#set protocols`
		`bfd destination 10:1:1::1`
		`source 10:1:1::2 template`
		`test`

R2	Register R1 as a BFD neighbor.	`vyatta@R2#set protocols`
		`ospfv3 area 1 virtual-link`
		`1.1.1.1 fall-over bfd`

Router	Step	Command
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

R2	Display the configuration.	`vyatta@R2#show protocols`
		` ospf`
		` ospfv3 {`
		`  area 1 {`
		`  network`
		` 10:1:1::0/128`
		`  virtual-link`
		` 1.1.1.1 {`
		` fall-over {`
		`  bfd`
		`  }`
		`  }`
		`  parameters {`
		`  }`
		` }`
		
		`vyatta@R2#show protocols`
		` bfd`
		`  bfd {`
		`  destination`
		` 10:1:1::1 {`
		`  source`
		` 10:1:1::2 {`
		`  template`
		` test`
		`  }`
		`  }`
		`  template test {`
		`  auth {`
		` simple {`
		`  key "********"`
		`  }`
		`  }`
		`  minimum-rx`
		` 300`
		`  minimum-tx`
		` 300`
		`  multiplier 3`
		`  }`
		` }`

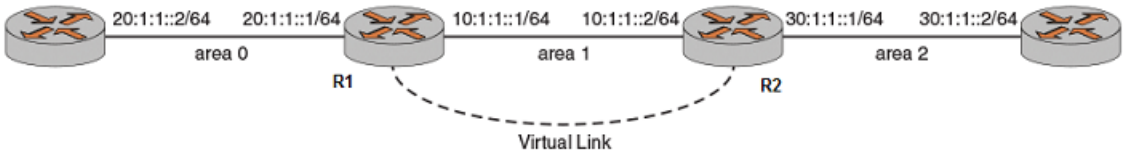
Configuring BFD for OSPFv3 on a virtual interface by using IPv6 addressing

To configure BFD for OSPFv3 neighbors on a virtual interface, you must first configure the OSPFv3 and the virtual interface for the two neighbors.

Consider a scenario in which you have two systems, R1 and R2. R1 and R2 share an OSPFv3 session. R1 and R2 are on a physical interface named dp0s5 which also has a virtual interface configured for the VLAN identifier of vlan-51. The following list provides the addresses of R1 and R2.

- R1 loopback address—`1:1:1::1/128`
- Data plane interface name—`dp0s5`
- VLAN identifier—`vlan-51`
- R1 interface facing vif 51—`10:1:1::1/64`
- R2 interface facing vif 51—`10:1:1::2/64`
- R2 loopback address—`2:2:2::2/128`
- R1 link local address—`fe80::6061:ff:fe00:b7d5`
- R2 link local address—`fe80::5054:ff:fe00:b6d5`

Figure 20: Configuring BFD for OSPFv3 on a virtual interface by using IPv6 addressing



Note: For OSPFv3 configurations, you must use the link local addresses for R1 and R2 for source and destination.

To configure a BFD session between R1 and R2, perform the following steps in configuration mode.

Table 18: Configuring BFD for OSPFv3 on a Virtual Interface by Using IPv6 Addressing

Router	Step	Command
R1	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R1	Associate the BFD `test` template with the destination address of R2 and the source address of R1. **Note:** You can also associate the BFD parameter template with the interface by using the `set interface dataplane vif <vf_id> bfd template <template-name>` command.	`vyatta@R1#set protocols bfd destination fe80::5054:ff:fe00:b6d5 source fe80::6061:ff:fe00:b7d5 template test`
R1	Register R2 as a BFD neighbor.	`vyatta@R1#set interfaces dataplane dp0s4 vif 51 ipv6 ospfv3 fall-over bfd`
R1	Commit the configuration.	`vyatta@R1#commit`
R1	Save the configuration.	`vyatta@R1#save`

Router	Step	Command

R1	Display the configuration.	vyatta@R1#show interfaces
		interfaces {
		dataplane
		dp0s5 {
		ipv6 {
		ospfv3 {
		fall-over {
		bfd
		}
		}
		}
		vif 51 {
		address 10:1:1::1/64
		ipv6
		{
		ospfv3 {
		fall-over {
		bfd
		}
		}
		}
		vlan
		51
		}
		}
		loopback lo {
		address
		1:1:1::1/128
		}
		}
		
		vyatta@R1#show protocols
		protocols {
		bfd {
		destination
		fe80::5054:ff:fe00:b6d5 {
		source
		fe80::6061:ff:fe00:b7d5 {
		template test
		}
		}
		template
		test{
		auth {
		simple {
		key "*******"
		}
		}
		minimum-rx
		300
		minimum-tx
		300
		multiplier 3

Router	Step	Command
R2	Create a BFD template called `test`.	See section Configuring the BFD template *(page 11)*.
R2	Associate the BFD `test` template with the destination address of R1 and the source address of R2. **Note:** You can also associate the BFD parameter template with the interface by using the `set interface dataplane vif <vf_id> bfd template <template-name>` command.	`vyatta@R2#set protocols bfd destination fe80::6061:ff:fe00:b7d5 source fe80::5054:ff:fe00:b6d5 template test`
R2	Register R1 as a BFD neighbor.	`vyatta@R2#set interfaces dataplane dp0s4 vif 51 ipv6 ospfv3 fall-over bfd`
R2	Commit the configuration.	`vyatta@R2#commit`
R2	Save the configuration.	`vyatta@R2#save`

Router	Step	Command

R2	Display the configuration.	vyatta@R2#show interfaces
		interfaces {
		dataplane
		dp0s5 {
		ipv6 {
		ospfv3 {
		fall-over {
		bfd
		}
		}
		}
		vif 51 {
		address 10:1:1::2/64
		ipv6
		{
		ospfv3 {
		fall-over {
		bfd
		}
		}
		}
		vlan
		51
		}
		}
		loopback lo {
		address
		2:2:2::2/128
		}
		}
		
		vyatta@R2#show protocols
		protocols {
		bfd {
		destination
		fe80::6061:ff:fe00:b7d5 {
		source
		fe80::5054:ff:fe00:b6d5 {
		template test
		}
		}
		template
		test{
		auth {
		simple {
		key "*******"
		}
		}
		minimum-rx
		300
		minimum-tx
		300
		multiplier 3

BFD Commands

interface dataplane <if_name> bfd template <template_name>

Associates a BFD parameter template with a single-hop BFD session, which originates from the specified interface.

Syntax:
`set interface dataplane` *if_name* **`bfd template`** *template_name*

Syntax:
`delete interface dataplane` *if_name* **`bfd template`** *template_name*

Syntax:
`show interface dataplane` *if_name*

If a BFD parameter template is not specified, minimum-tx is taken as 300, minimum-rx is taken as 300, and the multiplier is taken as 3.

`dataplane` ***if_name***
Specifies name of a data plane interface in the format of dp0*xy*.
`template` ***template_name***
Specifies name of a BFD parameter template.

Configuration mode

```
interface {
       dataplane if_name {
            bfd {
              template template_name
           }
     }

}
```

Use this command to associate a BFD parameter template with a single-hop BFD session, which is specified by the interface name.

Use the `set` form of the command to associate a BFD parameter template with a single-hop BFD session.

Use the `delete` form of the command to remove a BFD parameter template that is associated with a single-hop BFD session.

Use the `show` form of the command to display the BFD parameter template configured for the interface.

The following example shows how to associate a BFD parameter template called `test` with a data plane interface called `dp0s7` for a single-hop BFD session.

```
vyatta@vyatta#set interface dataplane dp0s7 bfd template test
```

interface dataplane <if_name> ip ospf fall-over bfd

Initiates a BFD session for all OSPFv2 neighbors on a physical interface.

Syntax:
`set interface dataplane` *if_name* **`ip ospf fall-over bfd`**

Syntax:
`delete interface dataplane` *if_name* **`ip ospf fall-over bfd`**

Syntax:
`show interface dataplane` *if_name*

BFD for OSPFv2 is disabled by default.

`dataplane` ***if_name***
Specifies name of a data plane interface in the format of dp0*xy*.

Configuration mode

```
interface dataplane if_name {
             ip {
               ospf {
                 fall-over {
                    bfd
                 }
             }
     }
}
```

Use this command to initiate a BFD session for all OSPFv2 neighbors on a physical interface.

Use the `set` version of the command to initiate a BFD session for all OSPFv2 neighbors on a physical interface.

Use the `delete` version of the command to delete a BFD session for all OSPFv2 neighbors on a physical interface.

Use the `show` form of the command to display the BFD parameter template configured for the interface.

The following example shows how to initiate a BFD session for all OSPFv2 neighbors on a physical interface called `dp0s4`.

```
vyatta@vyatta#set interface dataplane dp0s4 ip ospf fall-over bfd
```

interface dataplane <if_name> ipv6 ospfv3 fall-over bfd

Initiates a BFD session for all OSPFv3 neighbors on a physical interface.

Syntax:
`set interface dataplane` *if_name* **`ipv6 ospfv3 fall-over bfd`**

Syntax:
`delete interface dataplane` *if_name* **`ipv6 ospfv3 fall-over bfd`**

Syntax:
`show interface dataplane` *if_name*

BFD for OSPFv3 is disabled by default.

`dataplane` ***if_name***
Specifies name of a data plane interface in the format of dp0*xy*.

Configuration mode

```
interface dataplane if_name {
                ipv6 {
```

```
                    ospfv3 {
                     fall-over {
                           bfd
                        }
                  }
          }
}
```

Use this command to initiate a BFD session for all OSPFv3 neighbors on a physical interface.

Use the `set` version of the command to initiate a BFD session for all OSPFv3 neighbors on a physical interface.

Use the `delete` version of the command to delete a BFD session for all OSPFv3 neighbors on a physical interface.

Use the `show` form of the command to display the BFD parameter template configured for the interface.

The following example shows how to initiate a BFD session for all OSPFv3 neighbors on a physical interface called `dp0s4`.

```
vyatta@vyatta#set interface dataplane dp0s4 ipv6 ospf fall-over bfd
```

interface dataplane <if_name> vif <vif_id> bfd template <template-name>

Associates a BFD parameter template with a BFD session, which is specified by the virtual interface name.

Syntax:

`set interface dataplane` *if_name* **`vif`** *vif_id* **`bfd template`** *template-name*

Syntax:

`delete interface dataplane`*if_name* **`vif bfd template`** *template-name*

Syntax:

`show interface dataplane` *if_name*

If a BFD parameter template is not specified, minimum-tx is taken as 300, minimum-rx is taken as 300, and the multiplier is taken as 3.

`dataplane` ***if_name***
Specifies name of a data plane interface in the format of dp0*xy*.

`vif` ***vif_id***
Specifies a VLAN identifier for a virtual interface. The identifier ranges from 0 through 4094.

`template` ***template_name***
Specifies name of a BFD parameter template.

Configuration mode

```
interface {
  dataplane if_name {
             vif vif_id {
                      bfd {
                        template template_name
                          }
                    }
            }
}
```

Use this command to associate a BFD parameter template with a single-hop BFD session that is specified by the virtual interface name.

Use the `set` form of the command to associate a BFD parameter template with a BFD session.

Use the `delete` form of the command to remove a BFD parameter template that is associated with a BFD session.

Use the `show` form of the command to display the BFD parameter template configured for the interface.

The following example shows how to associate a BFD parameter template called `test` with a virtual interface `vif 10` for a BFD session.

```
vyatta@vyatta# set interface dataplane dp0s4 vif 10 bfd template test
```

interfaces dataplane <if_name> vif <vif-id> ip ospf fall-over bfd

Initiates a BFD session for all OSPFv2 neighbors on a virtual interface.

Syntax:
`set interfaces dataplane` *if_name* **`vif`** *vif-id* **`ip ospf fall-over bfd`**

Syntax:
`delete interfaces dataplane` *if_name* **`vif`** *vif-id* **`ip ospf fall-over bfd`**

Syntax:
`show interface dataplane` *if_name*

BFD for OSPFv2 is disabled by default.

`dataplane` ***if_name***
Specifies name of a data plane interface.

`vif` ***vif-id***
Specifies a VLAN identifier for a virtual interface. The identifier ranges from 0 through 4094.

Configuration mode

```
interface dataplane if_name {
            vif  vif-id {
                       ip {
                         ospf {
                   fall-over {
                             bfd
                             }
                         }
                }
        }
}
```

Use this command to initiate a BFD session for all OSPFv2 neighbors on a virtual interface.

Use the `set` version of the command to initiate a BFD session for all OSPFv2 neighbors on a virtual interface.

Use the `delete` version of the command to delete a BFD session for all OSPFv2 neighbors on a virtual interface.

Use the `show` form of the command to display the BFD parameter template configured for the interface.

The following example shows how to initiate a BFD session for all OSPFv2 neighbors on a virtual interface with a VLAN identifier of `51`.

```
vyatta@vyatta#set interfaces dataplane dp0s4 vif 51 ip ospf fall-over bfd
```

interfaces dataplane <if_name> vif <vif-id> ipv6 ospfv3 fall-over bfd

Initiates a BFD session for all OSPFv3 neighbors on a virtual interface.

Syntax:
`set interfaces dataplane` *if_name* **`vif`** *vif-id* **`ipv6 ospfv3 fall-over bfd`**

Syntax:
`delete interfaces dataplane` *if_name* **`vif`** *vif-id* **`ipv6 ospfv3 fall-over bfd`**

Syntax:
`show interface dataplane` *if_name*

BFD for OSPFv3 is disabled by default.

`dataplane` ***if_name***
Specifies name of a data plane interface in the format of dp0*xy*.

`vif` ***vif-id***
Specifies a VLAN identifier for a virtual interface. The identifier ranges from 0 through 4094.

Configuration mode

```
interface dataplane if_name{
            vif vif-id {
               ipv6 {
                ospfv3 {
                  fall-over {
                     bfd
                  }
               }
          }
     }
}
```

Use this command to initiate a BFD session for all OSPFv3 neighbors on a virtual interface.

Use the `set` version of the command to initiate a BFD session for all OSPFv3 neighbors on a virtual interface.

Use the `delete` version of the command to delete a BFD session for all OSPFv3 neighbors on a virtual interface.

Use the `show` form of the command to display the BFD parameter template configured for the interface.

The following example shows how to initiate a BFD session for all OSPFv3 neighbors on a virtual interface with a VLAN identifier of `51`.

```
vyatta@vyatta#set interfaces dataplane dp0s4 vif 51 ipv6 ospfv3 fall-over bfd
```

protocols bfd destination <destination_ip_address> source <source_ip_address> helper-session

Initiates a BFD session with a neighboring system with which the source system shares a static route, however the neighboring system does not share a static route with the source system.

Syntax:
`set protocols bfd destination` *destination_ip_address* **`source`** *source_ip_address* **`helper-session`**

Syntax:

`delete protocols bfd destination` *destination_ip_address* **`source`** *source_ip_address* **`helper-session`**

Syntax:
`show protocols bfd`

The BFD helper session is disabled by default.

destination_ip_address
IPv4 or IPv6 address of the destination system.
source_ip_address
IPv4 or IPv6 address of the source system.

Configuration mode

```
protocols {
       bfd {
         destination destination_ip_address  {
            source source_ip_address {
                    helper-session
                }
         }
     }
}
```

Use this command to monitor a BFD session initiated by the peer, where only the peer has a static client.

Use the `set` form of the command to initiate a BFD session with a neighboring system.

Use the `delete` form of the command to delete a BFD session with a neighboring system.

Use the `show` form of the command to display the details of a BFD session.

Note: The `source any` parameter is not supported for the helper session command.

The following example shows how to set a BFD session with a neighboring system with which the source system shares a static route, however the neighboring system does not share a static route with the source system. The interface address of the source is `10.37.99.161` and the interface address of the destination system is `10.37.99.162`.

```
vyatta@vyatta#set protocols bfd destination 10.37.99.162 source 10.37.99.161 helper-session
```

protocols bfd destination <destination_ip_address> source <source_ip_address> template <template_name>

Associates a BFD parameter template with a BFD session, which is specified by the source and destination address template.

Syntax:
`set protocols bfd` **`destination`** *destination_ip_address* **`source`** { *source_ip_address* | **`any`** } **`template`** *template_name*

Syntax:
`delete protocols bfd` **`destination`** *destination_ip_address* **`source`** { *source_ip_address* | **`any`** } **`template`** *template_name*

Syntax:
`show protocols bfd`

If the BFD parameter template is not specified, minimum-tx is taken as 300, minimum-rx is taken as 300, and the multiplier is taken as 3.

destination_ip_address
IPv4 or IPv6 address of a destination system.

source_ip_address
IPv4 or IPv6 address of a source system.

any
Associates a BFD template with all BFD sessions to the destination system.

template *template_name*
Specifies name of a BFD parameter template.

Configuration Mode

```
protocols {
       bfd {
        destination destination_ip_address {
             source source_ip_address | any }
        {
        template template_name
                 }
              }
          }
```

Use this command to associate a BFD parameter template with a BFD session, which is specified by the source and destination IP addresses.

Use the `set` form of the command to associate a BFD parameter template with a BFD session.

Use the `delete` form of the command to remove a BFD parameter template associated with a BFD session.

Use the `show` form of the command to display the details of a BFD session.

The following example shows how to associate a BFD parameter template called `test` with a BFD source and destination address template.

```
vyatta@vyatta#set protocols bfd destination 10.37.99.162 source 10.37.99.161 template test
```

The following example shows how to associate a BFD parameter template called `test` with a BFD session by using the `any` parameter. The `any` parameter associates the BFD parameter template with all BFD sessions with destination `10.37.99.162`.

```
vyatta@vyatta#set protocols bfd destination 10.37.99.162 source any template test
```

protocols bfd template <template_name>

Creates a BFD template that specifies the minimum-rx value, minimum-tx value, multiplier value, and authentication type for the BFD session.

Syntax:
`set protocols bfd template` *template_name* [**`minimum-rx`** *minimum-rx_value* | **`minimum-tx`** *minimum-tx_value* | **`multiplier`** *multiplier_value* | **`auth simple key`** *key_string*]

Syntax:
`delete protocols bfd` **`template`** *template_name*

Syntax:
```
show protocols bfd
```

If a BFD parameter template is not specified, minimum-tx is taken as 300, minimum-rx is taken as 300, and the multiplier is taken as 3.

`template` *template_name*
Specifies name of a BFD parameter template.
`minimum-rx` *minimum-rx_value*
Specifies the required minimum receiving interval for the BFD session. The interval ranges 20 through 10000 ms.
`minimum-tx` *minimum-tx_value*
Specifies a minimum transmission interval for the BFD session. The interval ranges 20 through 10000 ms.
`multiplier` *multiplier_value*
Specifies a multiplier for the BFD session. The mutiplier ranges 1 through 100.
`key` *key_string*
Specifies an alphanumeric password.

Configuration mode

```
bfd
{
 template template_name {
      auth {
         simple {
            key key_string
                }
           }
            minimum-rx minimum-rx_value
            minimum-tx minimum-tx_value
            multiplier multiplier_value
        }
}
```

Use the command to create a BFD parameter template that specifies the minimum-rx value, minimum-tx value, multiplier value, and authentication type for the BFD session.

Use the `set` form of the command to set a BFD template that specifies the minimum-rx value, minimum-tx value, multiplier value, and authentication type for the BFD session.

Use the `delete` form of the command to delete a BFD template.

Use the `show` form of the command to display the details of a BFD session.

The following example shows how to set the values for a BFD template called `test`.

```
vyatta@vyatta#set protocols bfd template test multiplier 3
vyatta@vyatta#set protocols bfd template test minimum-rx 300
vyatta@vyatta#set protocols bfd template test minimum-tx 300
vyatta@vyatta#set protocols bfd template test auth simple key lotr
```

protocols bgp <asn> neighbor <ip_address> fall-over bfd

Initiates a BFD session with a neighboring peer with which the system already shares a BGP session.

Syntax:
`set protocols bgp` *asn* **`neighbor`** *ip_address* **`fall-over bfd`**

Syntax:

```
delete protocols bgp asn neighbor ip_address fall-over bfd
```

Syntax:

```
show protocols bfd
```

BFD for BGP is disabled by default.

asn
Number of the AS in which a system resides.

neighbor *ip_address*
Specifies an IPv4 or IPv6 address of a peer system with which BFD is set up.

Configuration mode

```
bgp  {
    asn neighbor ip_address {
           fall-over {
              bfd
               }
          }
     }
```

Use this command to initiate a BFD session with a neighboring peer with which the system already shares a BGP session.

Use the `set` form of the command to initiate a BFD session with a neighboring peer with which the system already shares a BGP session.

Use the `delete` form of the command to delete a BFD session with a neighboring peer with which the system already shares a BGP session.

Use the `show` form of the command to display the details of a BFD session.

The following example shows how to initiate a BFD session with a neighboring peer whose interface address facing the system is `10.37.99.162`. The two systems share a BGP session.

```
vyatta@vyatta#  set protocols bgp 100 neighbor 10.37.99.162 fall-over bfd
```

protocols ospf area <area-id> virtual-link <dest_router_id> fall-over bfd

Initiates a BFD session for two OSPFv2 neighbors on a virtual link.

Syntax:

```
set protocols ospf area area-id virtual-link dest_router_id fall-over bfd
```

Syntax:

```
delete protocols ospf area area-id virtual-link dest_router_id fall-over bfd
```

Syntax:

```
show protocols bfd
```

BFD for OSPFv2 is disabled by default.

area *area-id*
Specifies the identifier of an OSPFv2 area configured. The identifier is an IP address or a decimal value.

dest_router_id
Destination router identifier of an OSPFv2 process. The identifier is an IPv4 address.

Configuration mode

```
protocols {
            ospf {
                   area area-id {
                     virtual-link dest_router_id {
                          fall-over bfd
                       }
                }
         }
}
```

Use this command to initiate a BFD session for two OSPFv2 neighbors on a virtual link.

Use the `set` form of the command to initiate a BFD session for two OSPFv2 neighbors on a virtual link.

Use the `delete` form of the command to delete the BFD session for two OSPFv2 neighbors on a virtual link.

Use the `show` form of the command to display the details of a BFD session.

The following example shows how to initiate a BFD session for two OSPFv2 neighbors on a virtual link. The destination router identifier is `2.2.2.2`.

```
vyatta@vyatta# set protocols ospf area 1 virtual-link 2.2.2.2 fall-over bfd
```

protocols ospfv3 area <area-id> virtual-link <dest_router_id> fall-over bfd

Initiates a BFD session for two OSPFv3 neighbors on a virtual link.

Syntax:

`set protocols ospfv3 area` *area-id* **`virtual-link`** *dest_router_id* **`fall-over bfd`**

Syntax:

`delete protocols ospfv3 area` *area-id* **`virtual-link`** *dest_router_id* **`fall-over bfd`**

Syntax:

`show protocols bfd`

BFD for OSPFv3 is disabled by default.

`area` ***area-id***
: Specifies the identifier of an OSPFv3 area configured. The identifier is an IP address or a decimal value.

dest_router_id
: Destination router identifier of an OSPFv3 process. The identifier is an IPv4 address.

Configuration mode

```
protocols {
            ospfv3 {
                   area area-id {
                     virtual-link dest_router_id {
                          fall-over bfd
                       }
                }
         }
}
```

Use this command to initiate a BFD session for two OSPFv3 neighbors on a virtual link.

Use the `set` form of the command to initiate a BFD session for two OSPFv3 neighbors on a virtual link.

Use the `delete` form of the command to delete the BFD session for two OSPFv3 neighbors on a virtual link.

Use the `show` form of the command to display the details of a BFD session.

The following example shows how to initiate a BFD session for two OSPFv3 neighbors on a virtual link. The destination router identification is `2.2.2.2`.

```
vyatta@vyatta#set protocols ospfv3 area 1 virtual-link 2.2.2.2 fall-over bfd
```

protocols static route <destination_ipv4_address> next-hop <nexthop_ipv4_address> fall-over bfd

Initiates a BFD session between two BFD peers on a static route by using IPv4 addressing.

Syntax:

```
set protocols static route destination_ipv4_address next-hop nexthop_ipv4_address fall-over bfd
```

Syntax:

```
delete protocols static route destination_ipv4_address next-hop nexthop_ipv4_address fall-over bfd
```

Syntax:

```
show protocols bfd
```

BFD for static routes is disabled by default.

destination_ipv4_address
IPv4 address of the destination system.

nexthop_ipv4_address
IPv4 address of the next-hop system enroute to the destination system.

Configuration mode

```
protocols {
    static {
        route destination_ipv4_address {
          next-hop nexthop_ipv4_address {
              fall-over {
                     bfd
              }
          }
        }
    }
}
```

Use this command to set up a BFD session between two systems on a static route by using IPv4 addressing.

Use the `set` form of this command to initiate a BFD session between two systems on a static route by using IPv4 addressing.

Use the `delete` form of this command to remove a BFD session between two systems on a static route by using IPv4 addressing.

Use the `show` form of this command to view the details of a BFD session.

The following example shows how to initiate a BFD session on a system that is connected to another system on a static route by using IPv4 addressing. The loopback address of the destination system is `2.2.2.2/32` and the interface address of the destination system is `10.10.10.2`.

```
vyatta@vyatta#set protocols static route 2.2.2.2/32 next-hop 10.10.10.2 fall-over bfd
```

protocols static route6 <destination_ipv6_address> next-hop <nexthop_ipv6_address> fall-over bfd

Initiates a BFD session between two systems on a static route by using IPv6 addressing.

Syntax:
`set protocols static route6` *destination_ipv6_address* **`next-hop`** *nexthop_ipv6_address* **`fall-over bfd`**

Syntax:
`delete protocols static route6` *destination_ipv6_address* **`next-hop`** *nexthop_ipv6_address* **`fall-over bfd`**

Syntax:
`show protocols bfd`

BFD for static routes is disabled by default.

destination_ipv6_address
IPv6 address of the destination system.

nexthop_ipv6_address
IPv6 address of the next-hop system enroute to the destination system.

Configuration mode

```
protocols {
    static {
        route6 destination_ipv6_address {
          next-hop nexthop_ipv6_address {
              fall-over {
                        bfd
                    }
                }
            }
      }
}
```

Use this command to set up a BFD session between two systems on a static route by using IPv6 addressing.

Use the `set` form of this command to initiate a BFD session between two systems on a static route by using IPv6 addressing.

Use the `delete` form of this command to remove a BFD session between two systems on a static route by using IPv6 addressing.

Use the `show` form of this command to view the details of a BFD session.

The following example shows how to initiate a BFD session on a system that is connected to another system on a static route by using IPv6 addressing. The loopback address of the destination system is `2:2:2::2/128` and the interface address of the destination system is `10:10:10::20`.

```
vyatta@vyatta#set protocols static route6 2:2:2::2/128 next-hop 10:10:10::20 fall-over bfd
```

show bfd session interface <if_name>

Displays BFD session information for a system interface.

Syntax:

```
show bfd session [ interface if_name [ detail [ destination_ipv4_address | destination_ipv6_address ] ] ]
```

interface *if_name*
Specifies a data plane interface name.
detail
Specifies a detailed report for the BFD session.
destination_ipv4_address
IPv4 address of the destination of the BFD session.
destination_ipv6_address
IPv6 address of the destination of the BFD session.

Operational mode.

Use the command to display the BFD session on a system interface.

The `show bfd session interface` command for the `dp0s6` interface displays the following information:

```
vyatta@vyatta:~$ show bfd session interface ifname dp0s6
Interface:      dp0s6  state: Up
Sess-Idx   Remote-Disc  Lower-Layer  Sess-Type   Sess-State  UP-Time   Remote-Addr
IPv4 Sessions:
1          0            IPv4         Single-Hop  Down        00:00:00  20.20.20.20/32



vyatta@vyatta:~$ show bfd session interface ifname dp0s6 detail
=======================================================

Session Interface Name : dp0s6           Session Index : 1
Lower Layer : IPv4                       Version : 1
Session Type : Single Hop                Session State : Down
Local Discriminator : 1                  Local Address : 20.20.20.10/32
Remote Discriminator : 0                 Remote Address : 20.20.20.20/32
Local Port : 49152                       Remote Port : 3784

Diagnostics : None

Configured parameters :
Template name : NULL                     Template applied at : Default
Min Tx : 300                             Min Rx : 300
Multiplier: 3
Bfd authentication : Disabled            BFD GTSM : Disabled

Negotiated parameters :
Neg Tx : 300                               Neg Rx : 300
Multiplier: 10

Counters values:
Pkt In : 0000000000000000                Pkt Out : 0000000000000572
Echo Out : 0000000000000000
Session down time : 00:00:00             Session up time : 00:00:00
UP Count : 0
Registered Clients:
Static

-------------------------------------------------------
```

Output field	Description
Session Interface Name	Interface for which you are running the command.
Session Index	Local discriminator for the session.
Lower Layer	IPv4 or IPv6.
Version	Version of the lower layer.

Output field	Description
Session Type	Single hop or multiple hop.
Session State	State of the BFD session: Admin-Down, Down, Init or Up.
Local Discriminator	Local discriminator for the session.
Local Address	Local address for the session.
Remote Discriminator	Remote discriminator for the session.
Remote Address	Remote address for the session.
Local Port	Source UDP port for the session.
Remote Port	Standard UDP Destination port
Diagnostics	Diagnostics for the current state of the session.
Configured Parameters	
Template name	Name for the parameter template.
Template applied at	Protocol or interface where the parameter template is applied.
Min Tx	Required minimum transmission time for the BFD session.
Min Rx	Required minimum receiver time for the BFD session.
Multiplier	Detection multiplier for the BFD session.
Bfd GTSM Disabled	Generalized TTL security mechanism (disabled for now).
Bfd Authentication Disabled	BFD authentication type supported.
Negotiated Parameters	
Neg Tx	Negotiated transmission time.
Neg Rx	Negotiated receiver time.
Counters values	
Pkt In	IPv4 asynchronous packet receiver count.
Pkt Out	IPv4 asynchronous packet transmission count.
Echo Out	IPv4 Echo packet transmission count.
Session down time	Time since the BFD session went DOWN.
Session up time	Time since the BFD session went UP.
UP Count	Number of times the BFD session has reached the UP state.
Registered Clients	Client or clients registered for the BFD service.

VRF Support

VRF support for BFD

All BFD configuration commands are supported on routing instances.

The following example shows how to configure the BFD source and destination for the default routing instance.

```
vyatta@R1# set protocols bfd destination 10.16.1.12 source 10.14.10.3
```

The following example shows how to apply the same configuration to the GREEN routing instance.

```
vyatta@R1# set routing routing-instance GREEN protocols bfd destination 10.16.1.12 source 10.14.10.3
```

Command support for VRF routing instances

VRF allows an AT&T Vyatta vRouter to support multiple routing tables, one for each VRF routing instance. Some commands in this guide support VRF and can be applied to particular routing instances.

Use the guidelines in this section to determine correct syntax when adding VRF routing instances to commands. For more information about VRF, refer to AT&T Vyatta Network Operating System Basic Routing Configuration Guide. This guide includes an overview of VRF, VRF configuration examples, information about VRF-specific features, and a list of commands that support VRF routing instances.

Adding a VRF routing instance to a Configuration mode command

For most Configuration mode commands, specify the VRF routing instance at the beginning of a command. Add the appropriate VRF keywords and variable to follow the initial action (**`set`**, **`show`**, or **`delete`**) and before the other keywords and variables in the command.

Example: Configuration mode example: syslog

The following command configures the syslog logging level for the specified syslog host. The command does not include a VRF routing instance, so the command applies to the default routing instance.

```
vyatta@R1# set system syslog host 10.10.10.1 facility all level debug
vyatta@R1# show system syslog
syslog {
    host 10.10.10.1 {
            facility all {
                    level debug
            }
    }
}
```

The following example shows the same command with the VRF routing instance (GREEN) added. Notice that **`routing routing-instance GREEN`** has been inserted between the basic action (**set** in the example) and the rest of the command. Most Configuration mode commands follow this convention.

```
vyatta@R1# set routing routing-instance GREEN system syslog host 10.10.10.1 facility all
 level debug
vyatta@R1# show routing
routing {
     routing-instance GREEN {
             system {
                     syslog {
                             host 11.12.13.2:514 {
                                     facility all {
                                             level debug
```

```
                                    }
                                }
                            }
                        }
                    }
}
```

Example: Configuration mode example: SNMP

Some features, such as SNMP, are not available on a per-routing instance basis but can be bound to a specific routing instance. For these features, the command syntax is an exception to the convention of specifying the routing instance at the beginning of Configuration mode commands.

The following example shows how to configure the SNMPv1 or SNMPv2c community and context for the RED and BLUE routing instances. The first two commands specify the RED routing instance as the context for community A and BLUE routing instance as the context for community B. The subsequent commands complete the configuration.

For more information about configuring SNMP, refer to AT&T Vyatta Network Operating System Remote Management Configuration Guide.

```
vyatta@R1# set service snmp community commA context RED
vyatta@R1# set service snmp community commB context BLUE
vyatta@R1# set service snmp view all oid 1
vyatta@R1# set service snmp community commA view all
vyatta@R1# set service snmp community commB view all
vyatta@R1# show service snmp community
 community commA {
        context RED
        view all
 }
 community commB {
        context BLUE
        view all
 }
[edit]
vyatta@vyatta#
```

Adding a VRF routing instance to an Operational mode command

The syntax for adding a VRF routing instance to an Operational mode command varies according to the type of command parameters:

- If the command does not have optional parameters, specify the routing instance at the end of the command.
- If the command has optional parameters, specify the routing instance after the required parameters and before the optional parameters.

Example: Operational mode examples without optional parameters

The following command displays dynamic DNS information for the default routing instance.

```
vyatta@vyatta:~$ show dns dynamic status
```

The following command displays the same information for the specified routing instance (GREEN). The command does not have any optional parameters, so the routing instance is specified at the end of the command.

```
vyatta@vyatta:~$ show dns dynamic status routing-instance GREEN
```

Example: Operational mode example with optional parameters

The following command obtains multicast path information for the specified host (10.33.2.5). A routing instance is not specified, so the command applies to the default routing instance.

```
vyatta@vyatta:~$ mtrace 10.33.2.5 detail
```

The following command obtains multicast path information for the specified host (10.33.2.5) and routing instance (GREEN). Notice that the routing instance is specified before the optional **detail** keyword.

```
vyatta@vyatta:~$ mtrace 10.33.2.5 routing-instance GREEN detail
```

Example: Operational mode example output: SNMP

The following SNMP **show** commands display output for routing instances.

```
vyatta@vyatta:~$ show snmp routing-instance
Routing Instance SNMP Agent is Listening on for Incoming Requests:
Routing-Instance            RDID
-----------------           ----
RED                         5

vyatta@vyatta:~$ show snmp community-mapping
SNMPv1/v2c Community/Context Mapping:
Community                   Context
---------                   -------
commA                       'RED'
commB                       'BLUE'
deva                        'default'


vyatta@vyatta:~$ show snmp trap-target
SNMPv1/v2c Trap-targets:
Trap-target                   Port   Routing-Instance Community
-----------                   ----   ---------------- ---------
1.1.1.1                              'RED'           'test'


vyatta@vyatta:~$ show snmp v3 trap-target
SNMPv3 Trap-targets:
Trap-target                   Port   Protocol Auth Priv Type   EngineID              Routing-
Instance User
-----------                   ----   -------- ---- ---- ----   --------
 ---------------- ----
2.2.2.2                       '162'  'udp'    'md5'      'infor                        'BLUE'
         'test'
```

List of Acronyms

Acronym	Description
ACL	access control list
ADSL	Asymmetric Digital Subscriber Line
AH	Authentication Header
AMI	Amazon Machine Image
API	Application Programming Interface
AS	autonomous system
ARP	Address Resolution Protocol
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BIOS	Basic Input Output System
BPDU	Bridge Protocol Data Unit
CA	certificate authority
CCMP	AES in counter mode with CBC-MAC
CHAP	Challenge Handshake Authentication Protocol
CLI	command-line interface
DDNS	dynamic DNS
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6
DLCI	data-link connection identifier
DMI	desktop management interface
DMVPN	dynamic multipoint VPN
DMZ	demilitarized zone
DN	distinguished name
DNS	Domain Name System
DSCP	Differentiated Services Code Point
DSL	Digital Subscriber Line
eBGP	external BGP
EBS	Amazon Elastic Block Storage
EC2	Amazon Elastic Compute Cloud
EGP	Exterior Gateway Protocol
ECMP	equal-cost multipath
ESP	Encapsulating Security Payload
FIB	Forwarding Information Base
FTP	File Transfer Protocol
GRE	Generic Routing Encapsulation
HDLC	High-Level Data Link Control
I/O	Input/Output
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers

Acronym	Description
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IPS	Intrusion Protection System
IKE	Internet Key Exchange
IP	Internet Protocol
IPOA	IP over ATM
IPsec	IP Security
IPv4	IP Version 4
IPv6	IP Version 6
ISAKMP	Internet Security Association and Key Management Protocol
ISM	Internet Standard Multicast
ISP	Internet Service Provider
KVM	Kernel-Based Virtual Machine
L2TP	Layer 2 Tunneling Protocol
LACP	Link Aggregation Control Protocol
LAN	local area network
LDAP	Lightweight Directory Access Protocol
LLDP	Link Layer Discovery Protocol
MAC	medium access control
mGRE	multipoint GRE
MIB	Management Information Base
MLD	Multicast Listener Discovery
MLPPP	multilink PPP
MRRU	maximum received reconstructed unit
MTU	maximum transmission unit
NAT	Network Address Translation
NBMA	Non-Broadcast Multi-Access
ND	Neighbor Discovery
NHRP	Next Hop Resolution Protocol
NIC	network interface card
NTP	Network Time Protocol
OSPF	Open Shortest Path First
OSPFv2	OSPF Version 2
OSPFv3	OSPF Version 3
PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PAT	Port Address Translation
PCI	peripheral component interconnect
PIM	Protocol Independent Multicast
PIM-DM	PIM Dense Mode
PIM-SM	PIM Sparse Mode
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PPPoA	PPP over ATM

Acronym	Description
PPPoE	PPP over Ethernet
PPTP	Point-to-Point Tunneling Protocol
PTMU	Path Maximum Transfer Unit
PVC	permanent virtual circuit
QoS	quality of service
RADIUS	Remote Authentication Dial-In User Service
RHEL	Red Hat Enterprise Linux
RIB	Routing Information Base
RIP	Routing Information Protocol
RIPng	RIP next generation
RP	Rendezvous Point
RPF	Reverse Path Forwarding
RSA	Rivest, Shamir, and Adleman
Rx	receive
S3	Amazon Simple Storage Service
SLAAC	Stateless Address Auto-Configuration
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical Network
SPT	Shortest Path Tree
SSH	Secure Shell
SSID	Service Set Identifier
SSM	Source-Specific Multicast
STP	Spanning Tree Protocol
TACACS+	Terminal Access Controller Access Control System Plus
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
ToS	Type of Service
TSS	TCP Maximum Segment Size
Tx	transmit
UDP	User Datagram Protocol
VHD	virtual hard disk
vif	virtual interface
VLAN	virtual LAN
VPC	Amazon virtual private cloud
VPN	virtual private network
VRRP	Virtual Router Redundancy Protocol
WAN	wide area network
WAP	wireless access point
WPA	Wired Protected Access

Index