Chapter 7

Configuring CSPF Tracing

The chapter describes how and when to configure Multiprotocol Label Switching (MPLS) Constrained Shortest Path First (CSPF) tracing. With each flag that you configure, more granular information about CSPF calculations is provided by the CSPF log file output. (See Table 11.)

Table 11: Checklist for Configuring CSPF Tracing

Configuring CSPF Tracing Tasks	Possible Action or Command
Understanding CSPF on page 74	
Configuring CSPF Tracing on page 76	[edit] edit protocols mpls [edit protocols mpls] set traceoptions file *filename* set traceoptions flag cspf set traceoptions flag cspf-link set traceoptions flag cspf-node show commit
Examining the CSPF Log File on page 77	
1. Trace Only CSPF Computations on page 78	[edit protocols mpls] run monitor start *filename* run show log *filename*
2. Trace Nodes Visited During CSPF Computations on page 79	[edit protocols mpls] run monitor start *filename* run show log *filename*
3. Trace Links Visited During CSPF Computations on page 81	[edit protocols mpls] run monitor start *filename* run show log *filename* show ted database

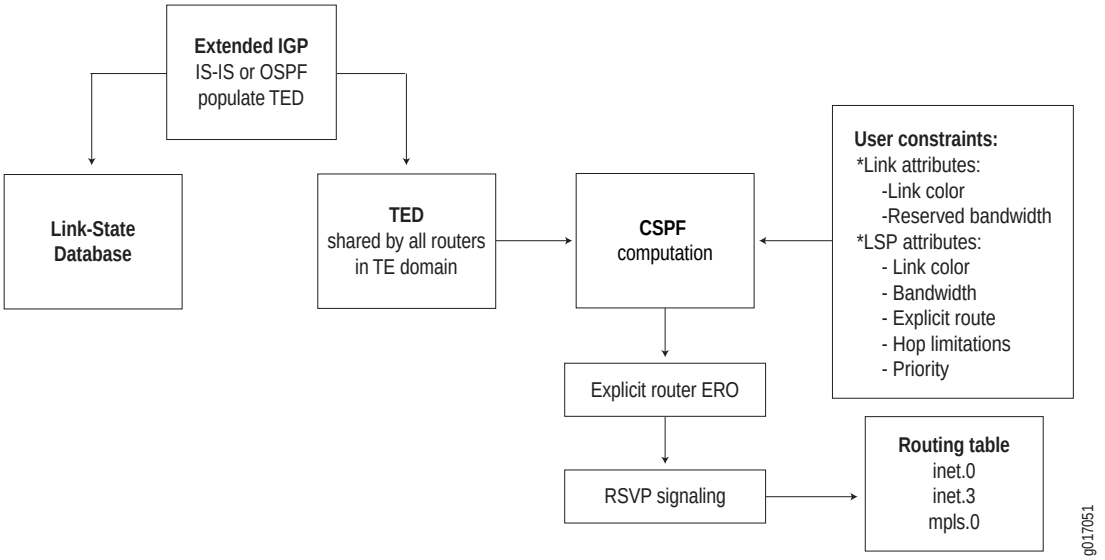
Understanding CSPF

Purpose CSPF is a link-state algorithm used in computing paths for label-switched paths (LSPs) that are subject to multiple constraints. When computing paths for LSPs, CSPF considers not only the topology of the network, but also the attributes of the LSP and the links, and attempts to minimize congestion by balancing the network load.

After pruning paths that do not meet the configured constraints from the shortest-path-first (SPF) tree, CSPF derives the best available path based on the information in the traffic engineering database (TED). Based on the best available path, CSPF produces a strict Explicit Route Object (ERO) which the Resource Reservation Protocol (RSVP) uses to signal the LSP.

The CSPF algorithm is a modified version of the SPF algorithm used within the link-state databases of Intermediate System-to-Intermediate System (IS-IS) and Open Shortest Path First (OSPF) protocols. CSPF operates on the traffic engineering database, which is constructed through extensions to IS-IS and OSPF. Figure 2 illustrates the various components that contribute to the CSPF computation.

Figure 2: CSPF Components



To select a path, CSPF follows these steps:

1. Computes LSPs one at a time, beginning with the highest priority LSP (the one with the lowest setup priority value). Among LSPs of equal priority, CSPF starts with those that have the highest bandwidth requirement.

2. Prunes the traffic engineering database of all links that are not full duplex and do not have sufficient reservable bandwidth.

3. If the LSP configuration includes the include statement, prunes all links that do not share any included colors.

4. If the LSP configuration includes the exclude statement, prunes all links that contain excluded colors. If the link does not have a color, it is accepted.

5. Finds the shortest path toward the LSP's egress router, taking into account explicit-path constraints. For example, if the path must pass through Router A, two separate SPFs are computed, one from the ingress router to Router A, and the other from Router A to the egress router.

6. If several paths have equal cost, chooses the path whose last-hop address is the same as the LSP's destination.

7. If several equal-cost paths remain, selects the path with the fewest number of hops.

8. If several equal-cost paths remain, applies the CSPF load-balancing rule configured on the LSP (least fill, most fill, or random).

The result of the above steps is a strict-hop ERO that details each hop along the calculated path. The ERO is passed to the RSVP protocol process, where it is used to signal and establish the LSP in the network.

Steps To Take To determine how and when to configure and examine MPLS CSPF tracing, follow these steps:

1. Configuring CSPF Tracing on page 76

2. Examining the CSPF Log File on page 77

Configuring CSPF Tracing

Purpose When the output of the show mpls lsp extensive command indicates that the CSPF algorithm has failed, configuring CSPF tracing on the ingress router can often provide more information about the problem.

Action On the ingress router, to configure a log file and specify MPLS tracing flags, follow these steps:

1. In configuration mode, go to the following hierarchy level:

   ```
   [edit]
   user@host# edit protocols mpls
   ```

2. Configure a log file:

   ```
   [edit protocols mpls]
   user@host# set traceoptions file filename
   ```

3. Depending on your situation, specify all or one of the following CSPF-specific tracing flags:

   ```
   [edit protocols mpls]
   user@host# set traceoptions flag cspf
   user@host# set traceoptions flag cspf-link
   user@host# set traceoptions flag cspf-node
   ```

4. Verify and commit the configuration:

   ```
   user@host# show
   user@host# commit
   ```

Sample Output

```
[edit protocols mpls]
user@R1# show
traceoptions {
    file cspf;
    flag cspf;
    flag cspf-link;
    flag cspf-node;
}
label-switched-path R6-to-R1 {
    to 10.0.0.1;
}
interface so-0/0/0.0;
interface so-0/0/1.0;
interface so-0/0/2.0;
interface so-0/0/3.0;
```

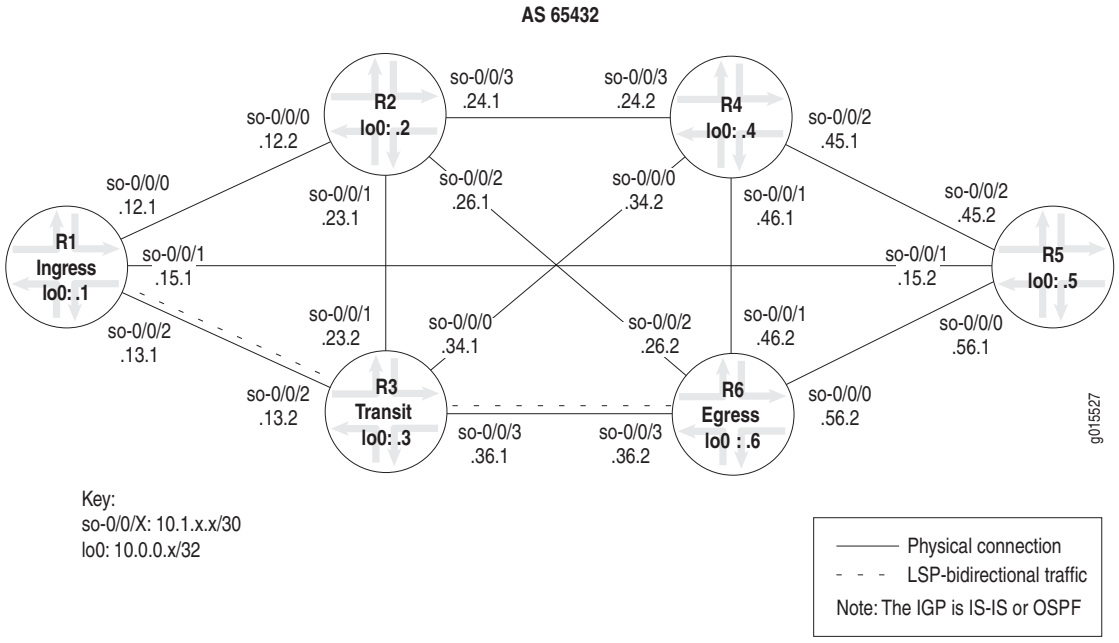
What It Means The sample output shows a typical CSPF tracing configuration. The log file cpsf contains all the information gathered for each configured flag. Each flag provides slightly different information about CSPF computations. The cspf flag traces CSPF computations only; the cspf-link flag traces links visited during CSPF computations, and the cspf-node flag traces nodes visited during CSPF computations. See “Examining the CSPF Log File” on page 77 for information about examining a CSPF log file.

Examining the CSPF Log File

Purpose The CSPF log file provides useful information about the steps taken by the CSPF algorithm to calculate the shortest path from the ingress router to the egress router. The following steps and output illustrate the CSPF algorithm in the successful establishment of an LSP. With each flag that you configure, starting with the cspf flag, then the cspf-node flag, and finally the cspf-link flag, more granular information about CSPF calculations is provided by the output in the CSPF log file configured to gather the information.

Figure illustrates the example network topology used in this section. The example MPLS network uses IS-IS Level 2 and a policy to create traffic. However, IS-IS Level 1 or an OSPF area can be used and the policy omitted if the network has existing Border Gateway Protocol (BGP) traffic.

Figure 3: MPLS Network Topology



The MPLS network in Figure is a router-only network with SONET interfaces that consist of the following components:

- A full-mesh interior BGP (IBGP) topology, using AS 65432
- MPLS and RSVP enabled on all routers
- A send-statics policy on routers R1 and R6 that allows a new route to be advertised into the network
- Two unidirectional LSPs between R1 and R6, which allow bidirectional traffic

See the *JUNOS MPLS Network Operations Guide* for information on configuring an MPLS network. The ingress router R1 is configured with CSPF tracing, and the output examined in the following three steps is taken from R1.

Steps To Take To examine the CSPF log file, follow these steps:

1. Trace Only CSPF Computations on page 78
2. Trace Nodes Visited During CSPF Computations on page 79
3. Trace Links Visited During CSPF Computations on page 81

Step 1: Trace Only CSPF Computations

Purpose The cspf flag provides an overview of the CSPF computations performed and the resulting ERO for the LSP. Details about nodes or links visited during CSPF computations are not included in this log file.

Action To run trace CSPF computations and examine the CSPF log file, enter the following JUNOS command-line interface (CLI) commands:

```
[edit protocols mpls]
user@R1# run monitor start filename
user@R1# run show log filename
```

NOTE: To stop monitoring CSPF, issue the monitor stop command. If you are working in configuration mode, issue the run monitor stop command.

Sample Output 1

```
[edit protocols mpls]
user@R1# show
traceoptions {
    file cspf;
    flag cspf;
}
label-switched-path R6-to-R1 {
    to 10.0.0.1;
}
interface so-0/0/0.0;
interface so-0/0/1.0;
interface so-0/0/2.0;
interface so-0/0/3.0;
```

Sample Output 2

```
[edit protocols mpls]
user@R1> show log cspf
Apr 29 11:35:59 trace_on: Tracing to "/var/log/cspf" started
Apr 29 13:22:52 RPD_MPLS_LSP_DOWN: MPLS LSP R1-to-R6 down on primary()
Apr 29 13:22:52 RPD_MPLS_PATH_DOWN: MPLS path down on LSP R1-to-R6
Apr 29 13:22:52 CSPF adding path R1-to-R6(primary) to CSPF queue 1
Apr 29 13:22:52 CSPF creating CSPF job
Apr 29 13:22:52
Apr 29 13:22:52 CSPF for path R1-to-R6(primary), begin at R1.00, starting
Apr 29 13:22:52         bandwidth: CT0=0bps; setup priority: 0; random
Apr 29 13:22:52 CSPF final destination 10.0.0.6
Apr 29 13:22:52 CSPF starting from R1.00 (10.0.0.1) to 10.0.0.6, hoplimit 254
Apr 29 13:22:52 CSPF Reached target
Apr 29 13:22:52 CSPF completed in 0.000106s
Apr 29 13:22:52 CSPF ERO for R1-to-R6(primary) (2 hops)
Apr 29 13:22:52         node 10.1.15.2/32
Apr 29 13:22:52         node 10.1.56.2/32
Apr 29 13:22:52 CSPF for R1-to-R6 done!
```

```
Apr 29 13:22:52 RPD_MPLS_PATH_UP: MPLS path up on LSP R1-to-R6
Apr 29 13:22:52 RPD_MPLS_LSP_UP: MPLS LSP R1-to-R6 up on primary() Route 10.1.15.2 10.1.56.2
monitor stop
```

What It Means Sample Output 1 shows the configuration of the cspf file and cspf flag at the [edit protocols mpls traceoptions] hierarchy level. See "Configuring CSPF Tracing" on page 76 for steps to configure CSPF tracing.

Sample Output 2 shows the contents of the cspf file in the /var/log/ directory on ingress router R1. The cspf file contains the CSPF computations obtained when the cspf flag is configured at the [edit protocols mpls traceoptions] hierarchy level and after the run monitor start cspf and run show log cspf commands were issued.

Each line of output describes the steps taken by the CSPF algorithm to calculate the shortest path between the ingress and egress routers. The result of the CSPF algorithm is formed into a strict-hop ERO that details each hop along the calculated path. For example, the ERO for the LSP R1-to-R6 contains two hops that pass through nodes 10.1.15.2/32 and 10.1.56.2.32. When the ERO is completed, CSPF for R1-to-R6 done!, the ERO is passed to the RSVP protocol process, where it is used for signaling and establishing the LSP in the network. The output shows RPD_MPLS_LSP_UP, indicating that the LSP was established successfully.

Step 2: Trace Nodes Visited During CSPF Computations

Purpose The configuration of the cspf-node flag provides details in the log file about the nodes visited during CSPF computations. The node information is in addition to the overview information provided by the cspf flag. Details about links visited during CSPF computations are not included in the log file.

Action To trace nodes visited during CSPF computations and to examine the CSPF log file, enter the following JUNOS CLI commands:

```
[edit protocols mpls]
user@R1# run monitor start filename
user@R1# run show log filename
```

NOTE: To stop monitoring CSPF, issue the monitor stop command. If you are working in configuration mode, as shown in the sample output, issue the run monitor stop command.

Sample Output 1

```
[edit protocols mpls]
user@R1# show
traceoptions {
    file cspf-node1;
    flag cspf;
    flag cspf-node;
}
label-switched-path R6-to-R1 {
    to 10.0.0.1;
}
interface so-0/0/0.0;
interface so-0/0/1.0;
interface so-0/0/2.0;
interface so-0/0/3.0;
```

Sample Output 2

```
[edit protocols mpls]
user@R1# run show log cspf-node1 | no-more
Apr 29 13:39:08 trace_on: Tracing to "/var/log/cspf-node1" started
Apr 29 13:40:43 RPD_MPLS_LSP_DOWN: MPLS LSP R1-to-R6 down on primary()
Apr 29 13:40:43 RPD_MPLS_PATH_DOWN: MPLS path down on LSP R1-to-R6
Apr 29 13:40:43 CSPF adding path R1-to-R6(primary) to CSPF queue 1
Apr 29 13:40:43 CSPF creating CSPF job
Apr 29 13:40:43
Apr 29 13:40:43 CSPF for path R1-to-R6(primary), begin at R1.00, starting
Apr 29 13:40:43         bandwidth: CT0=0bps; setup priority: 0; random
Apr 29 13:40:43 CSPF final destination 10.0.0.6
Apr 29 13:40:43 CSPF starting from R1.00 (10.0.0.1) to 10.0.0.6, hoplimit 254
Apr 29 13:40:43     Node R1.00 (10.0.0.1) metric 0, hops 0, avail 32000 32000 32000 32000
Apr 29 13:40:43     Node R3.00 (10.0.0.3) metric 10, hops 1, avail 32000 32000 32000 32000
Apr 29 13:40:43     Node R5.00 (10.0.0.5) metric 10, hops 1, avail 32000 32000 32000 32000
Apr 29 13:40:43     Node R2.00 (10.0.0.2) metric 10, hops 1, avail 32000 32000 32000 32000
Apr 29 13:40:43     Node R4.00 (10.0.0.4) metric 20, hops 2, avail 32000 32000 32000 32000
Apr 29 13:40:43     Node R6.00 (10.0.0.6) metric 20, hops 2, avail 32000 32000 32000 32000
Apr 29 13:40:43 CSPF Reached target
Apr 29 13:40:43 CSPF completed in 0.000304s
Apr 29 13:40:43 CSPF ERO for R1-to-R6(primary) (2 hops)
Apr 29 13:40:43         node 10.1.12.2/32
Apr 29 13:40:43         node 10.1.26.2/32
Apr 29 13:40:43 CSPF for R1-to-R6 done!
Apr 29 13:40:43 RPD_MPLS_PATH_UP: MPLS path up on LSP R1-to-R6
Apr 29 13:40:43 RPD_MPLS_LSP_UP: MPLS LSP R1-to-R6 up on primary() Route 10.1.12.2 10.1.26.2
[...Output truncated...]
```

```
[edit protocols mpls]
user@R1# run monitor stop
```

What It Means Sample Output 1 shows the configuration of the cspf-node file, cspf flag and cspf-node flag at the [edit protocols mpls traceoptions] hierarchy level. See "Configuring CSPF Tracing" on page 76 for steps to configure CSPF tracing.

Sample Output 2 shows the contents of the cspf-node file in the /var/log/ directory on ingress router R1. The cspf-node file contains the CSPF computations logged when the cspf and cspf-node flags are configured at the [edit protocols mpls traceoptions] hierarchy level and after the run monitor start cspf and run show log cspf commands are issued.

Each line of output describes the steps taken by the CSPF algorithm to calculate the shortest path between the ingress and egress routers. Because the cspf-node flag is configured, the output shows the nodes visited during the calculations performed by the CSPF algorithm. For example, all nodes in the network shown in Figure on page 77 are included.

The result of the CSPF algorithm is formed into a strict-hop ERO. For example, the ERO for the LSP R1-to-R6 contains two hops that pass through nodes 10.1.12.2/32 and 10.1.26.2.32. When the ERO is completed, CSPF for R1-to-R6 done!, the ERO is passed to the RSVP protocol process, where it is used for signaling and establishing the LSP in the network. The output shows RPD_MPLS_LSP_UP, indicating that the LSP was established successfully.

Step 3: Trace Links Visited During CSPF Computations

Purpose The configuration of the cspf-link flag provides details in the log file about the links visited during CSPF computations. The link information is in addition to the overview information provided by the cspf flag, and the node information provided by the cpsf-node flag.

Action To run trace links visited during CSPF computations and examine the CSPF log file, enter the following JUNOS CLI commands:

```
[edit protocols mpls]
user@R1# run monitor start filename
user@R1# run show log filename
user@R1# run show ted database
```

NOTE: To stop monitoring CSPF, issue the monitor stop command. If you are working in configuration mode, as shown in the sample output, issue the run monitor stop command.

Sample Output 1

```
[edit protocols mpls]
user@R1# show
traceoptions {
    file cspf-link;
    flag cspf;
    flag cspf-link;
}
label-switched-path R6-to-R1 {
    to 10.0.0.1;
}
interface so-0/0/0.0;
interface so-0/0/1.0;
interface so-0/0/2.0;
interface so-0/0/3.0;
```

Sample Output 2

```
[edit protocols mpls]
user@R1# run show log cspf-link | no-more
Apr 29 13:29:52 trace_on: Tracing to "/var/log/cspf-link" started
Apr 29 13:30:27 RPD_MPLS_LSP_DOWN: MPLS LSP R1-to-R6 down on primary()
Apr 29 13:30:27 RPD_MPLS_PATH_DOWN: MPLS path down on LSP R1-to-R6
Apr 29 13:30:27 CSPF adding path R1-to-R6(primary) to CSPF queue 1
Apr 29 13:30:27 CSPF creating CSPF job
Apr 29 13:30:27
Apr 29 13:30:27 CSPF for path R1-to-R6(primary), begin at R1.00, starting
Apr 29 13:30:27 bandwidth: CT0=0bps; setup priority: 0; random
Apr 29 13:30:27 CSPF final destination 10.0.0.6
Apr 29 13:30:27 CSPF starting from R1.00 (10.0.0.1) to 10.0.0.6, hoplimit 254
Apr 29 13:30:27   Node R1.00 (10.0.0.1) metric 0, hops 0, avail 32000 32000 32000 32000
Apr 29 13:30:27      Link 10.1.13.1->10.1.13.2(R3.00/10.0.0.3) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.13.1->10.1.13.2 is 10.1.13.2->10.1.13.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27         encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27      Link 10.1.12.1->10.1.12.2(R2.00/10.0.0.2) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.12.1->10.1.12.2 is 10.1.12.2->10.1.12.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
```

```
Apr 29 13:30:27      Link 10.1.15.1->10.1.15.2(R5.00/10.0.0.5) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.15.1->10.1.15.2 is 10.1.15.2->10.1.15.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27     Node R3.00 (10.0.0.3) metric 10, hops 1, avail 32000 32000 32000 32000
Apr 29 13:30:27      Link 10.1.13.2->10.1.13.1(R1.00/10.0.0.1) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       skipped: end point already visited
Apr 29 13:30:27      Link 10.1.34.1->10.1.34.2(R4.00/10.0.0.4) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.34.1->10.1.34.2 is 10.1.34.2->10.1.34.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27      Link 10.1.23.2->10.1.23.1(R2.00/10.0.0.2) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.23.2->10.1.23.1 is 10.1.23.1->10.1.23.2
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27       metric: 20 vs 10; hops: 2 vs 1; avail: 32000 32000 32000 32000
Apr 29 13:30:27      Link 10.1.36.1->10.1.36.2(R6.00/10.0.0.6) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.36.1->10.1.36.2 is 10.1.36.2->10.1.36.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27     Node R5.00 (10.0.0.5) metric 10, hops 1, avail 32000 32000 32000 32000
Apr 29 13:30:27      Link 10.1.15.2->10.1.15.1(R1.00/10.0.0.1) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       skipped: end point already visited
Apr 29 13:30:27      Link 10.1.45.2->10.1.45.1(R4.00/10.0.0.4) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.45.2->10.1.45.1 is 10.1.45.1->10.1.45.2
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27       metric: 20 vs 20; hops: 2 vs 2; avail: 32000 32000 32000 32000
Apr 29 13:30:27       Better path: random wins
Apr 29 13:30:27      Link 10.1.56.1->10.1.56.2(R6.00/10.0.0.6) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.56.1->10.1.56.2 is 10.1.56.2->10.1.56.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27       metric: 20 vs 20; hops: 2 vs 2; avail: 32000 32000 32000 32000
Apr 29 13:30:27       Old path is better
Apr 29 13:30:27   Node R2.00 (10.0.0.2) metric 10, hops 1, avail 32000 32000 32000 32000
Apr 29 13:30:27      Link 10.1.12.2->10.1.12.1(R1.00/10.0.0.1) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       skipped: end point already visited
Apr 29 13:30:27      Link 10.1.23.1->10.1.23.2(R3.00/10.0.0.3) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       skipped: end point already visited
Apr 29 13:30:27      Link 10.1.24.1->10.1.24.2(R4.00/10.0.0.4) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.24.1->10.1.24.2 is 10.1.24.2->10.1.24.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27       metric: 20 vs 20; hops: 2 vs 2; avail: 32000 32000 32000 32000
Apr 29 13:30:27       Old path is better
Apr 29 13:30:27      Link 10.1.26.1->10.1.26.2(R6.00/10.0.0.6) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.26.1->10.1.26.2 is 10.1.26.2->10.1.26.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27       metric: 20 vs 20; hops: 2 vs 2; avail: 32000 32000 32000 32000
Apr 29 13:30:27       Old path is better
Apr 29 13:30:27   Node R4.00 (10.0.0.4) metric 20, hops 2, avail 32000 32000 32000 32000
Apr 29 13:30:27      Link 10.1.34.2->10.1.34.1(R3.00/10.0.0.3) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       skipped: end point already visited
```

```
Apr 29 13:30:27      Link 10.1.24.2->10.1.24.1(R2.00/10.0.0.2) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       skipped: end point already visited
Apr 29 13:30:27      Link 10.1.45.1->10.1.45.2(R5.00/10.0.0.5) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       skipped: end point already visited
Apr 29 13:30:27      Link 10.1.46.1->10.1.46.2(R6.00/10.0.0.6) metric 10 color 0x00000000 bw 155.52Mbps
Apr 29 13:30:27       Reverse Link for 10.1.46.1->10.1.46.2 is 10.1.46.2->10.1.46.1
Apr 29 13:30:27       link's interface switch capability descriptor #1
Apr 29 13:30:27        encoding: Packet, switching: Packet
Apr 29 13:30:27       link passes constraints
Apr 29 13:30:27       metric: 30 vs 20; hops: 3 vs 2; avail: 32000 32000 32000 32000
Apr 29 13:30:27     Node R6.00 (10.0.0.6) metric 20, hops 2, avail 32000 32000 32000 32000
Apr 29 13:30:27 CSPF Reached target
Apr 29 13:30:27 CSPF completed in 0.001880s
Apr 29 13:30:27 CSPF ERO for R1-to-R6(primary) (2 hops)
Apr 29 13:30:27         node 10.1.13.2/32
Apr 29 13:30:27         node 10.1.36.2/32
Apr 29 13:30:27 CSPF for R1-to-R6 done!
Apr 29 13:30:27 RPD_MPLS_PATH_UP: MPLS path up on LSP R1-to-R6
Apr 29 13:30:27 RPD_MPLS_LSP_UP: MPLS LSP R1-to-R6 up on primary() Route 10.1.13.2 10.1.36.2
```

Sample Output 3

```
user@R1# run show ted database | no-more
TED database: 6 ISIS nodes 6 INET nodes
ID                            Type Age(s) LnkIn LnkOut Protocol
R1.00(10.0.0.1)               Rtr     148     3      3 IS-IS(2)
    To: R3.00(10.0.0.3), Local: 10.1.13.1, Remote: 10.1.13.2
    To: R5.00(10.0.0.5), Local: 10.1.15.1, Remote: 10.1.15.2
    To: R2.00(10.0.0.2), Local: 10.1.12.1, Remote: 10.1.12.2
ID                            Type Age(s) LnkIn LnkOut Protocol
                                                       OSPF(0.0.0.0)
    To: R3.00(10.0.0.3), Local: 10.1.13.1, Remote: 10.1.13.2
    To: R5.00(10.0.0.5), Local: 10.1.15.1, Remote: 10.1.15.2
    To: R2.00(10.0.0.2), Local: 10.1.12.1, Remote: 10.1.12.2
ID                            Type Age(s) LnkIn LnkOut Protocol
R2.00(10.0.0.2)               Rtr     580     4      4 IS-IS(2)
    To: R1.00(10.0.0.1), Local: 10.1.12.2, Remote: 10.1.12.1
    To: R3.00(10.0.0.3), Local: 10.1.23.1, Remote: 10.1.23.2
    To: R4.00(10.0.0.4), Local: 10.1.24.1, Remote: 10.1.24.2
    To: R6.00(10.0.0.6), Local: 10.1.26.1, Remote: 10.1.26.2
ID                            Type Age(s) LnkIn LnkOut Protocol
                                                       OSPF(0.0.0.0)
    To: R1.00(10.0.0.1), Local: 10.1.12.2, Remote: 10.1.12.1
    To: R3.00(10.0.0.3), Local: 10.1.23.1, Remote: 10.1.23.2
    To: R4.00(10.0.0.4), Local: 10.1.24.1, Remote: 10.1.24.2
    To: R6.00(10.0.0.6), Local: 10.1.26.1, Remote: 10.1.26.2
ID                            Type Age(s) LnkIn LnkOut Protocol
R3.00(10.0.0.3)               Rtr     390     4      4 IS-IS(2)
    To: R1.00(10.0.0.1), Local: 10.1.13.2, Remote: 10.1.13.1
    To: R4.00(10.0.0.4), Local: 10.1.34.1, Remote: 10.1.34.2
    To: R2.00(10.0.0.2), Local: 10.1.23.2, Remote: 10.1.23.1
    To: R6.00(10.0.0.6), Local: 10.1.36.1, Remote: 10.1.36.2
ID                            Type Age(s) LnkIn LnkOut Protocol
                                                       OSPF(0.0.0.0)
    To: R1.00(10.0.0.1), Local: 10.1.13.2, Remote: 10.1.13.1
    To: R4.00(10.0.0.4), Local: 10.1.34.1, Remote: 10.1.34.2
    To: R2.00(10.0.0.2), Local: 10.1.23.2, Remote: 10.1.23.1
    To: R6.00(10.0.0.6), Local: 10.1.36.1, Remote: 10.1.36.2
ID                            Type Age(s) LnkIn LnkOut Protocol
R4.00(10.0.0.4)               Rtr     677     4      4 IS-IS(2)
    To: R3.00(10.0.0.3), Local: 10.1.34.2, Remote: 10.1.34.1
    To: R5.00(10.0.0.5), Local: 10.1.45.1, Remote: 10.1.45.2
    To: R2.00(10.0.0.2), Local: 10.1.24.2, Remote: 10.1.24.1
    To: R6.00(10.0.0.6), Local: 10.1.46.1, Remote: 10.1.46.2
ID                            Type Age(s) LnkIn LnkOut Protocol
```

```
                                                        OSPF(0.0.0.0)
    To: R3.00(10.0.0.3), Local: 10.1.34.2, Remote: 10.1.34.1
    To: R5.00(10.0.0.5), Local: 10.1.45.1, Remote: 10.1.45.2
    To: R2.00(10.0.0.2), Local: 10.1.24.2, Remote: 10.1.24.1
    To: R6.00(10.0.0.6), Local: 10.1.46.1, Remote: 10.1.46.2
ID                           Type Age(s) LnkIn LnkOut Protocol
R5.00(10.0.0.5)              Rtr     609     3      3 IS-IS(2)
    To: R1.00(10.0.0.1), Local: 10.1.15.2, Remote: 10.1.15.1
    To: R4.00(10.0.0.4), Local: 10.1.45.2, Remote: 10.1.45.1
    To: R6.00(10.0.0.6), Local: 10.1.56.1, Remote: 10.1.56.2
ID                           Type Age(s) LnkIn LnkOut Protocol
                                                        OSPF(0.0.0.0)
    To: R1.00(10.0.0.1), Local: 10.1.15.2, Remote: 10.1.15.1
    To: R4.00(10.0.0.4), Local: 10.1.45.2, Remote: 10.1.45.1
    To: R6.00(10.0.0.6), Local: 10.1.56.1, Remote: 10.1.56.2
ID                           Type Age(s) LnkIn LnkOut Protocol
R6.00(10.0.0.6)              Rtr     633     4      4 IS-IS(2)
    To: R3.00(10.0.0.3), Local: 10.1.36.2, Remote: 10.1.36.1
    To: R4.00(10.0.0.4), Local: 10.1.46.2, Remote: 10.1.46.1
    To: R5.00(10.0.0.5), Local: 10.1.56.2, Remote: 10.1.56.1
    To: R2.00(10.0.0.2), Local: 10.1.26.2, Remote: 10.1.26.1
ID                           Type Age(s) LnkIn LnkOut Protocol
                                                        OSPF(0.0.0.0)
    To: R3.00(10.0.0.3), Local: 10.1.36.2, Remote: 10.1.36.1
    To: R4.00(10.0.0.4), Local: 10.1.46.2, Remote: 10.1.46.1
    To: R5.00(10.0.0.5), Local: 10.1.56.2, Remote: 10.1.56.1
    To: R2.00(10.0.0.2), Local: 10.1.26.2, Remote: 10.1.26.1
```

What It Means Sample Output 1 shows the configuration of the cspf-link file, cspf flag, and cspf-link flag at the [edit protocols mpls traceoptions] hierarchy level. See “Configuring CSPF Tracing” on page 76 for steps to configure CSPF tracing.

Sample Output 2 shows the contents of the cspf-link file in the /var/log/ directory on ingress router R1. The cspf-link file contains the CSPF computations logged when the cspf and cspf-link flags are configured at the [edit protocols mpls traceoptions] hierarchy level and after the run monitor start cspf and run show log cspf commands are issued.

Each line of output describes the steps taken by the CSPF algorithm to calculate the shortest path between the ingress and egress routers. Because the cspf-link flag is configured, the output shows the node and link information included in the calculations performed by the CSPF algorithm. For example, R1 (ingress router) has three links with three possible paths to the egress router (R6), Link 10.1.13.1->10.1.13.2, Link 10.1.12.1->10.1.12.2, and Link 10.1.15.1->10.1.15.2. In this instance, the CSPF algorithm selects the 10.1.13.1->10.1.13.2 link as the shortest path to the egress router.

The result of the CSPF algorithm is formed into a strict-hop ERO. For example, the ERO for the LSP R1-to-R6 contains two hops that pass through nodes 10.1.13.2/32 and 10.1.36.2.32. When the ERO is completed, CSPF for R1-to-R6 done!, the ERO is passed to the RSVP protocol process, where it is used for signaling and establishing the LSP in the network. The output shows RPD_MPLS_LSP_UP, indicating that the LSP was established successfully

Sample Output 3 shows a brief summary of the contents of the traffic engineering database. (For more detailed information, use the detail or extensive options.) When CSPF tracing is configured, the contents of the specified CSPF log file should correlate to the contents of the traffic engineering database; that is, the links shown in the output for the show log *filename* command should also appear in the output for the show ted database command. In the example network shown in Figure on page 77, the six nodes and all links associated with those nodes appear in the output of both commands.

The traffic engineering database is built through link–state routing protocol extensions that allow for the flooding of information regarding available link bandwidth, link coloring, and so on. Also, the traffic engineering database includes information contained in the OSPF and IS-IS databases. For example, R1 has three links configured at IS-IS Level 2, and the same three links configured in OPSF area 0.0.0.0.

For a more detailed examination of the traffic engineering database, see “Examining a CSPF Failure” on page 87.