9

Network Monitoring

Contents

Overview

Network monitoring serves two functions:

- It tests and controls static and Dynamic Host Configuration Protocol (DHCP) routes.
- It tests network performance, logging when performance falls below a certain level.

For the ProCurve Secure Router, testing routes is the primary purpose of network monitoring. Network monitoring can detect a route that fails at any point en route to the destination and remove the failed route so that a backup route can take effect.

Network monitoring relies on two connected mechanisms:

- Network monitor probes—A probe collects the information by which a route or a network server is tested. A probe can consist of packets as simple as Internet Control Message Protocol (ICMP) echo packets (pings) or as complex as HTTP requests for particular information.
- Network monitor tracks—A track uses probes to test routes or remote servers, with the goal of removing failed routes and logging poor performance.

This overview introduces probes and tracks so that you can understand how they work together to serve the functions of network monitoring.

Network Monitor Probes

A probe periodically sends out packets to collect information about the state of a route or of a device such as a remote server. Each time that a probe sends out a packet, the probe is said to have run one test.

Note

A probe packet is the packet sent out as a test. A probe is the entire process of periodically transmitting probe packets and determining whether tests pass or fail.

Probe Characteristics

A probe is defined by these configurable characteristics:

- period—specifies the frequency at which the probe runs a test (that is, transmits a probe packet), for example, every 60 seconds
- tolerance—determines how many tests must fail before a probe as a whole is considered to have failed
- timeout—the length of time before a test is considered to have failed; for example, a probe ping packet might have 500 milliseconds in which to reach its destination and return
- packet type—ICMP echo (ping), TCP connect, or HTTP request packets
- destination—the IP address or hostname to which probe packets are transmitted
- source—the source IP address for probe packets

Depending on the probe's type, you can define other characteristics for probe packets.

For ICMP echo packets, you can define:

- packet size
- data pattern

For TCP connect packets, you can define the source port in addition to the source address. You must define the destination TCP port.

ICMP echo packets either do or do not successfully reach their destination and return. Similarly, TCP connect packets either do or do not successfully open a session. HTTP request probes, on the other hand, can collect various types of information, determining success or failure based on the information returned. Therefore, you can define many characteristics for HTTP request packets, including:

- the type of packet (Get, Head, or Raw)
- for raw packets, a command string
- an expected status in the response packet
- an expected string in the response packet
- the Web server's root path
- the source port

You will learn more about configuring these options in "Configuring Probes" on page 9-11.

Probe States

A probe configured on your ProCurve Secure Router is also defined by its state—either Pass or Fail. The state is determined by the number of tests that have failed, together with the tolerance setting. For example, you could configure the tolerance so that a probe fails when 9 out of 10 tests fail or when 20 tests in a row fail.

To run tests, a probe must be active (up). If you manually shut down the probe, it stops transmitting packets, and its state becomes fixed at Pass, despite any previous failed tests.

Network Monitor Tracks

By itself, a probe can monitor network conditions, but it cannot take any action based on its findings. A track is responsible for tying probe findings to an action.

For example, if you are monitoring a route, the track is what actually removes a faulty route. Or, if you are using a track to monitor a server, the track is what logs status changes.

Track Characteristics

A track is defined by the probes (either one or two) that it uses to monitor the network. In addition, a track is defined by the action that it takes based on its state.

Track States

Like a probe, a track can have one of two states: Pass or Fail. The state depends on the state of the associated probe or probes. When a track relies on two probes, you can configure it to either:

- pass only when both probes pass
- pass when either one of the two probes passes

Track Actions

A track can take an action based on the results that probes return about network conditions. You can enable a track to take one or both of these actions when its state changes to Fail:

- remove a route
- create a log entry

Purposes of Network Monitoring

Now that you understand how network monitoring works, you can learn about the services it provides to your WAN.

Testing Static Routes

A static route has a low administrative distance, based on the assumption that the person who created the route can vouch for its accuracy and preferred status.

However, static routes do not always remain accurate. If one of the ProCurve Secure Router's own interfaces goes down, the router immediately stops using any route associated with that interface. However, links can fail at any point along a route; without dynamic routing, the local router has no way to know that the traffic that it continues to forward can no longer reach its destination.

For example, your ProCurve Secure Router may connect to the Internet through its Ethernet connection and a cable modem. The router's Ethernet interface stays up even if the modem's connection fails. As far as the router is concerned, its static route to the Internet is good, but traffic cannot reach its destination.

Network monitoring is a mechanism for testing static routes and removing those routes that fail the test. Thus static routing becomes more like dynamic routing, in that it can respond to failed network connections.

However, network monitoring does not help the router learn new routes. If you want the router to switch to an alternate route if the primary route fails, you must take one of these steps in advance:

- You enter the alternate route or routes and enable load sharing.
- More commonly, you create a floating static route (an alternate route with a higher administrative distance than the primary route). A common use for network monitoring is to create a floating route through a demand interface. If the ProCurve Secure Router cannot reach a key destination (such as a server at the company headquarters), the ProCurve Secure Router places a call to establish an Integrated Services Digital Network (ISDN) connection.

  Figure 9-1 shows an example of a WAN that uses network monitoring to detect when to establish a backup ISDN connection to the Internet service provider (ISP). When the probe fails to reach the vital headquarters' server, the track removes the primary route through the cable modem, and the floating static route becomes active.

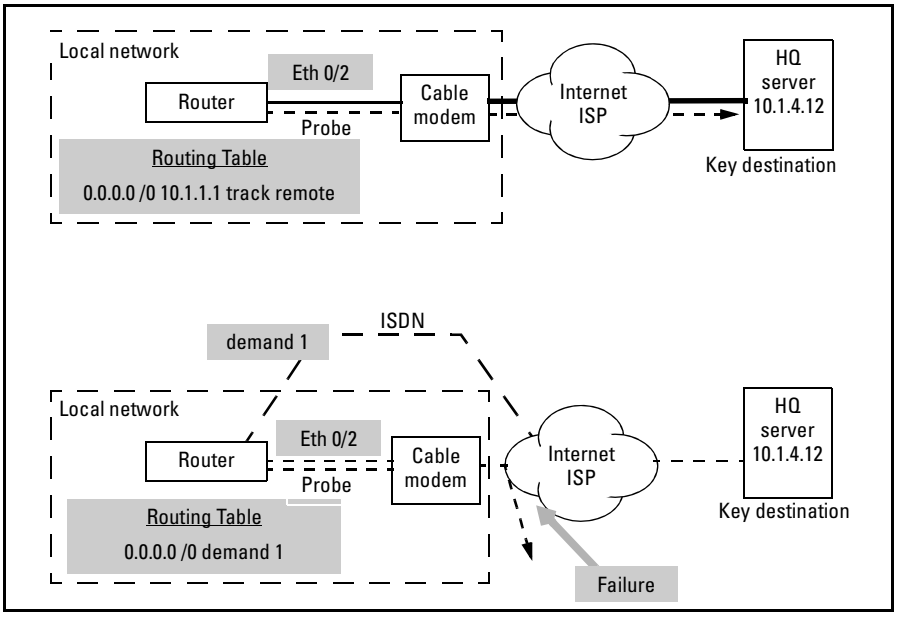

Figure 9-1. Detecting a Failed Primary Route with Network Monitoring

Figure 9-2 shows another scenario, in which the local ProCurve Secure Router establishes a site-to-site Virtual Private Network (VPN) through the Internet. If the remote network's connection to the Internet fails, the router at the local site places an ISDN call to connect to the remote site directly. The router at the local site continues to forward Internet traffic through the primary connection.

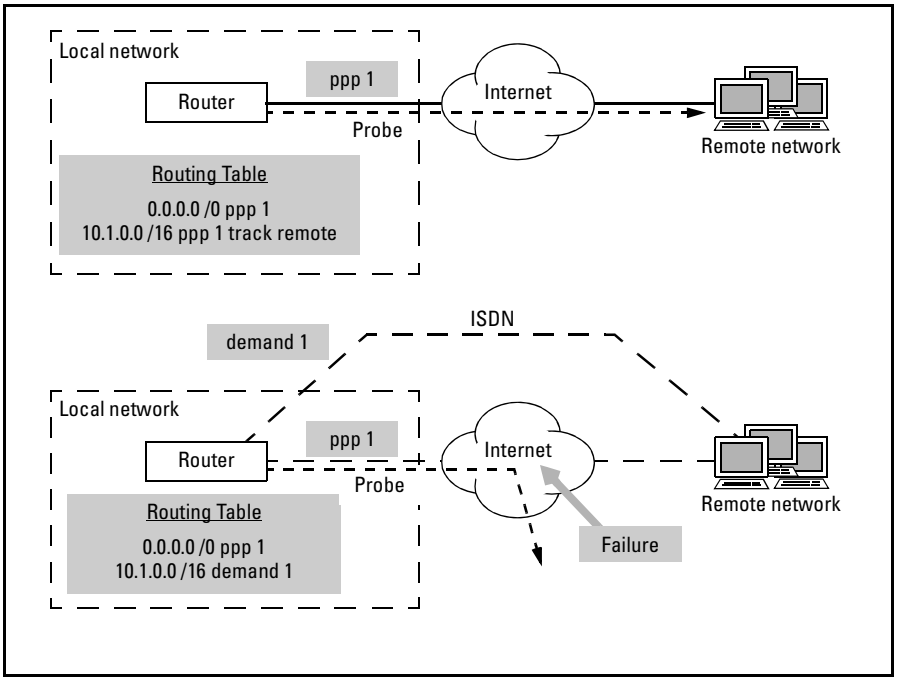

Figure 9-2. Detecting a Failed Connection to a Remote Network

For more information on configuring these routes, see the *Basic Management and Configuration Guide*, *Chapter 11: IP Routing—Configuring Static Routes*. For more information on demand routing and ISDN, see the *Basic Management and Configuration Guide*, *Chapter 8: Configuring Demand Routing for Primary ISDN Modules* and the *Advanced Management and Configuration Guide*, *Chapter 3: Configuring Backup WAN Connections*.

In addition to monitoring static routes, tracks can monitor default routes received with a DHCP address or a negotiated IP address. For example, your router might receive an IP address, default gateway, and other information from your ISP. If connectivity to necessary resources fails, network monitoring removes the information learned from the ISP. If you have configured an alternate route to the remote network, that route can become active.

Note

Even if no viable alternate route exists, network monitoring can serve a useful function: removing the route from the table conserves valuable bandwidth by preventing the router from forwarding traffic that will only be dropped.

The ProCurve Secure Router allows several types of probes to test routes. You can use ICMP echo probes to test simple connectivity to a remote device. Or, you can use TCP connect or HTTP probes to test connectivity to a particular application on a remote server.

Monitoring Network Performance

Network monitoring on the ProCurve Secure Router is designed primarily to test connectivity. However, you can use network monitoring to create limited performance tests. For example, you can set the timeout value low in order to test the link speed. When the link is too congested, the probe returns a failure to the track.

Sometimes you will want a backup connection to become active, and sometimes you will simply want to monitor the congestion. In the second case, you should *not* associate the track with a route. Simply enable the track to create a log each time the probe indicates unacceptable performance. You can enable the ProCurve Secure Router to save the log to event history and to forward it to an external server. (See “Enabling a Track to Log Changes” on page 9-29.)

TCP connect and HTTP request probes can perform limited tests on the status of a particular network resource. For example, you could configure the probe to fail if a remote server does not return a response within a set amount of time, indicating that the server is overburdened. For Web servers, you can configure the HTTP probe to fail when the server does not return the correct status. Again, the track can either remove a route to make way for a secondary route, or it can simply log the failure so that you can take corrective action as needed.

Routing Probe Traffic using Policy-Based Routing (PBR)

When you use tracks to monitor routes, you must use PBR to route probe traffic. Creating a route map for probe packets ensures that the probe monitors the connection that you want it to monitor at all times.

When a track that is monitoring a route fails, it removes that route from the routing table. However, the probes remain active. This behavior allows the track to reinstate a route after a failure is remedied; however, it can also cause a problem: reinstating the route before it is actually fixed.

The track has removed the failed primary route from the routing table, so the backup route is added to the table, and traffic can again reach its destination. Because probe packets can also reach their destination over the backup

connection, the track reinstates the failed primary route—only to remove the route again when the probes start to fail. Users lose their sessions as the connection toggles up and down.

Use PBR to solve this problem. You configure a route map to forward probe packets along the route that the probe tests. In this way, no matter what actions tracks take on the routing table, the probes continue to test the correct connection.

Configuring Network Monitoring

Network monitoring relies on tracks; each track monitors a certain network function. A track can:

- test the validity of a static route or of a default route received with a DHCP or negotiated address
- monitor a particular server or network service, such as a Web server or File Transfer Protocol (FTP) site

To create a track, you must complete these basic steps:

1. Configure one or more probes.
2. If tracks monitor routes, configure PBR for the probe traffic.
3. Configure the track and associate it with one or two probes.
4. Enable the track to take an action, which typically includes:
   - configuring the track to log changes in probe states
   - associating the track with a route
5. If you are using Network Address Translation (NAT), configure the router to use different translation statements depending on the current forwarding interface.
6. If you are using the ProCurve Secure Router firewall, create access control policies (ACPs) to disable the reverse path forwarding (RPF) check on traffic received on the primary and backup interfaces.

The following sections will first explain how to complete these steps and then give examples of tracks configured to monitor various network functions.

Configuring Probes

To configure a probe, you must complete these tasks:

1. Create and name the probe and select its type.
2. Specify the probe's destination.
3. Configure the probe's tolerance.
4. Activate the probe.

You can use the following default settings or your own custom settings:

- period—default: 60 seconds
- timeout—default: 1.5 seconds for ICMP, 10 seconds for TCP or HTTP
- source address—default: the outbound interface's address
- source port (for TCP and HTTP)—default: the port automatically selected by the probe

Other options vary, depending on the type of probe you have selected, and are documented separately for each type.

Each probe sends packets to a single destination address. You must configure a different probe for each monitored server or destination network.

Creating a Probe and Selecting Its Type

The probe type is one of your most important decisions. There are three probe types:

- ICMP echo—Use this probe type if you simply want to test connectivity so that a track can remove faulty static routes. This is the most basic probe type; the default packet size is relatively small, so the probe adds little overhead. For an ICMP echo probe, which transmits ICMP echo packets, a test passes if the destination returns a valid response before the timeout expires.
- TCP connect—Use this probe type if you want to test a remote server that runs a TCP application and does not respond to ICMP echo probes. For tests, the probe initiates a three-way TCP handshake. If the server sends a valid response before the timeout expires, the probe considers the test to have passed and terminates the TCP connection.

- HTTP request—Use this probe type if you want to monitor a Web server. Like a TCP connect probe, the HTTP probe initiates a connection with the server, requiring the server to respond within a set time, after which the probe terminates the session. However, you can also require the server's response to include a certain status, or you can even configure the probe to obtain information. (For a more detailed discussion of these options, see "Special Considerations for HTTP Request Probes" on page 9-20.)

To create the probe and select its type, from the global configuration mode context, enter:

Syntax: probe <*name*> [icmp-echo | tcp-connect | http-request]

You then move into the network monitor probe configuration mode context:

ProCurve(config-probe-<*name*>)#

For example, you might enter:

```
ProCurve(config)# probe MyWebPerformance http-request
ProCurve(config-probe-MyWebPerformance)#
```

Specifying the Probe's Destination

The next step for creating a probe is to specify the device to which the probe sends test packets. For example, if you want to test a particular route, you might enter the IP address of the gateway for the route's destination. Or, if you want to monitor a particular server, enter that server's hostname.

You can specify the destination as either a hostname or an IP address. The hostname should be fully qualified (for example, *www.company_a.com*).

To specify the destination of an ICMP echo probe, from the network monitor probe configuration mode context, enter:

Syntax: destination <*hostname* | *A.B.C.D*>

For example, enter this command to test a route to an ISP router at address 10.1.1.1:

```
ProCurve(config-probe-Branch2)# destination 10.1.1.1
```

For TCP connect probes, in addition to the hostname or IP address, you must specify the port for the service or application in question. Enter this command:

Syntax: destination <*hostname* | *A.B.C.D*> port <*number*>

Valid port numbers range from 1 to 65535. See Table 9-1 for the port numbers of some common TCP applications.

Table 9-1. Well-known TCP Ports

Application	TCP Port
Border Gateway Protocol (BGP)	179
Daytime server	13
Domain Name System (DNS)	53
FTP	21
Hostname	101
Internet Relay Chat (IRC)	194
Kerberos login	543
Kerberos shell	544
Microsoft directory services (such as Active Directory)	445
Network News Transfer Protocol (NNTP)	119
Protocol Independent Multicast (PIM) Rendezvous Point (RP)	496
Post Office Protocol (POP)2	109
POP3	110
Secure Shell (SSH)	22
Simple Mail Transfer Protocol (SMTP)	25
Simple Network Time Protocol (SNTP)	37
Sun Remote Procedure Call (RPC)	111
Syslog	514
Terminal Access Controller Access Control System (TACACS)	49
Talk	517
Telnet	23
Trivial File Transfer Protocol (TFTP)	69
Unix to Unix Copy (UUCP)	520

For example, a network administrator could enter this command to probe the company's FTP server:

```
ProCurve(config-probe-FTPServer)# destination www.company_a.com port 21
```

For HTTP request probes, the default destination port is 80. If your server uses a different port, you can specify that. For example, if you want to specify port 8080, enter:

```
ProCurve(config-probe-WebServer)# destination www.company_a.com port 8080
```

Specifying the Test's Timeout

The timeout value determines the length of time within which a probe packet must return. For ICMP echo and TCP connect probes, the timeout solely determines whether a test passes: the response must return within the allotted time, or the test fails. For HTTP request packets, other conditions can affect the success of the test. (See "Requiring a Particular Status in the Web Server's Response" on page 9-21 and "Requiring Particular Information from the Web Server" on page 9-23.)

To specify the timeout, from the network monitor probe configuration mode context, enter:

Syntax: timeout <*milliseconds*>

The valid range is from 250 to 4,294,967,295 milliseconds. The default value depends on the probe type:

- ICMP echo: 1,500 milliseconds (1.5 seconds)
- TCP connect: 10,000 milliseconds (10 seconds)
- HTTP request: 10,000 milliseconds (10 seconds)

Specifying the Probe's Tolerance

The probe's tolerance determines how sensitive the probe is to failed tests—that is, how many tests can fail before the probe reports a failure to the track. In an actual network, some packets will always be lost; you do not want the probe to base its status entirely on a single test.

You *must* specify the tolerance; it has no default setting.

You can set the tolerance in one of two ways:

- consecutive failures—If a certain number of tests in a row fail, the probe fails.

  With this type of tolerance, the probe counts consecutive failures. Any time a test passes, the probe resets the count. When the count reaches the tolerance setting, the probe fails.

- rate of failure—If tests fail at a certain rate, the probe fails. You specify the rate by configuring two settings: the number of tests that can fail within a set of tests, and the number of tests in the set. For example, you can specify that the probe enters the Fail state if 20 out of 25 tests fail.

  The size of the set determines the number of tests that the probe considers and is a running count. For example, if the set size is 25, the probe counts all failures in only the 25 *most recent* tests.

Note that, even after a probe has entered the Fail state, it continues to run tests. In this way, the probe can revert to the Pass state when the monitored conditions become acceptable again.

The conditions for a probe passing once again also depend on the type of tolerance. A probe with a consecutive failure tolerance reverts to the Pass state as soon as one test passes. For a probe with a rate of failure tolerance, reverting to the Pass state is slightly more complicated. Enough tests must pass that, when looking at the X most recent tests (X being the set size), the number of failures does not exceed the number set by the tolerance.

The consecutive failure option is the most straightforward, particularly when you simply want to test connectivity. You might choose the rate of failure option when setting up a probe to monitor the performance of a server. When a probe sends a series of requests to an over-burdened server, chances are that at least one request will get through, so a probe with a consecutive failures tolerance might detect a problem rarely, if ever. However, the server could still be failing to respond an unacceptable percentage of the time. A rate of failure tolerance would detect the poor performance.

To set the tolerance according to consecutive failures, from the network monitor probe configuration mode context, enter:

Syntax: tolerance consecutive-failures <*failures*>

The valid range for failures is from 1 to 255.

To set the tolerance according to rate of failure, from the network monitor probe configuration mode context, enter:

Syntax: tolerance rate-of-failure <*failures*> <*set size*>

The valid range for failures is from 1 to 254, and the valid range for set size is from 1 to 255. The value for failures allowed within a set must, of course, be smaller than the value for the set size.

Specifying the Probe's Period

The period determines how often the probe runs a test—that is, how often a probe packet is sent out.

To set the period, from the network monitor probe configuration mode context, enter:

Syntax: period <*seconds*>

For ICMP echo probes, enter a value from 1 to 65535 seconds. (If you want to set the period to 1 second, you must first decrease the timeout because the period must be greater than the timeout.) For TCP connect and HTTP request probes, enter a value from 60 to 65535 seconds. For all probe types, the default period is 60 seconds.

The period setting involves trade-offs: The shorter the period, the more granular the picture you receive of network performance and the more quickly an incorrect route is detected. However, shorter periods add more overhead to WAN connections.

You can combine the settings for period, timeout (expressed in seconds, not milliseconds), and tolerance to estimate the length of time a network would experience unacceptable conditions before the probe notifies the track that it has failed:

- For a probe with a consecutive failure tolerance:
  - (Period + Timeout) * Tolerance consecutive failures
- For a probe with a rate of failure tolerance:
  - (Period + Timeout) * Tolerance allowed failures (minimum)
  - (Period + Timeout) * Tolerance set size (maximum)

Such a calculation might be important for a probe that is intended to monitor the status of a route. For example, consider a configuration with a period of 10 seconds, a timeout of 1,000 milliseconds (1 second), and a tolerance of 5 consecutive failures. The sum of the period and the timeout (11 seconds in the example) is the maximum length for a test. Multiply that maximum length by the tolerance. In this example, the probe would detect a failed route after 55 seconds. If this time is unacceptably long, decrease the period or the tolerance.

Of course, you may not want to decrease the period too much, because probes add overhead to your network. Also, when you are testing for connectivity, you should set the tolerance for ICMP echo probes to at least 3 in order to compensate for routinely lost packets.

Setting the Source Address for Probe Packets

By default, the probe takes as its source the address of the interface through which it was transmitted. Leaving this setting at its default is often a good idea because the probe requires a valid address on one of the router's local interfaces in order to function properly. When you manually specify a probe's source address, you risk a probe malfunction if the local interface address is changed later.

To set the source address for probe packets, from the network monitor probe configuration mode context, enter:

Syntax: source-address <*A.B.C.D*>

Setting the Source Port for Probe Packets

For TCP connect and HTTP request packets, you can also specify the source port. By default, the router starts sending probe packets out from port 1026 and increments by one with each probe packet. But you can override this behavior and configure the probe to send packets from a set port.

From the network monitor probe configuration mode context, enter:

Syntax: source-port <*number*>

Valid ports range from 1 to 65535.

Consider whether a firewall stands between your router and the remote server that it is testing. If it filters out certain source ports, you should change the probe to an acceptable port. Also consider allowed source ports in ACLs configured on other network devices.

Another reason that you might set the source port manually is to distinguish one probe's traffic from another probe's, even when the probes have the same destination. Thus you can configure a route map entry to forward one probe's packets out a primary interface, and you can configure another route map entry to forward a second probe's packets out a backup interface.

Special Considerations for Configuring Probes

The following sections list special considerations for ICMP echo probes, TCP connect probes, and HTTP request probes.

Special Considerations for ICMP Echo Probes

ICMP echo probes are used to test the current status of paths to particular networks or endpoints. Therefore, you should typically set the destination to the gateway device of the network in question. (You would not want a track to remove a route to an entire network simply because one endpoint in that network lost connectivity.)

The timeout value that you select depends largely on the purpose of the probe:

- If the probe is testing connectivity only, set the timeout higher so that a congested but functional route is not removed.
- If the probe is monitoring congestion, set the timeout lower so that the track is triggered when an average packet's transit time falls below the acceptable standard. In this case, you might want to set the tolerance according to rate of failure.

For ICMP echo probes, you can also configure these additional packet settings:

- size
- data pattern
- source address

Setting the Size. The default data length for an ICMP echo packet is 64 bytes. (The header size is fixed at 28 bytes, resulting in a total packet size of 92 bytes.)

You can customize the size of the data portion of an ICMP echo packet. Remember, though, that the default size is the minimum size and so adds the least overhead.

To set the length of the packet's data field, from the network monitor probe configuration mode context, enter:

Syntax: size <*data length*>

The valid range for the length is from 64 to 1448 bytes.

One reason to change the packet size is to test for fragmentation. Voice over IP (VoIP) frames require a route over which they will not be fragmented. You can test a link that you want to use for VoIP by creating a ICMP echo probe. Set the size for the probe to match the size of the payload of the VoIP frames (typically 20 to 160 bytes).

You must configure the router to mark the probe packets with a don't fragment bit. The probe then fails when a connection has a lower maximum transmission unit (MTU) than the probe packets. When you configure the route map for forwarding the probes, enter this command to set the don't fragment bit:

Syntax: set ip df

See "Implementing PBR to Route Probe Traffic" on page 9-34 for more information on configuring the route map.

Setting the Data Pattern. An ICMP echo packet typically tests the ability of two points to communicate. For test purposes, you only care that something is communicated, not what that something is. Therefore, you can usually leave the data pattern at the default for pings. This pattern begins at 0x00 and increments along the length of the packet. For example, an echo packet of length 64 would begin with these bytes in the data field:

00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11…

The bytes would continue incrementing up to 3f (the sixty-fourth byte).

If you so desire, however, you can change this pattern. To specify the new pattern, from the network monitor probe configuration mode context, enter:

Syntax: data <*pattern*>

Replace ***<pattern>*** with hexadecimal characters. For example, you could enter:

```
ProCurve(config-probe-StationC)# data 0FF0
```

The packet repeats the indicated pattern as many times as necessary to fill out the packet's data bits.

In some rare circumstances, a network interface may malfunction when transmitting or receiving a certain data string (often a long string of continuous ones or zeros). If you have reason to believe that your network has equipment with this problem, you can set up probes that use suspect patterns until you locate the problem.

Special Considerations for TCP Connect Probes

TCP Connect probes monitor TCP servers which include, among others:

- email servers
- FTP servers
- Domain Name System (DNS) servers
- time servers

Therefore, it is important that you set the destination port for the service you are testing, as well as the device's name or address. (See "Specifying the Probe's Destination" on page 9-12.)

Also, you might need to manually specify a valid source port. (See "Setting the Source Address for Probe Packets" on page 9-17.)

Special Considerations for HTTP Request Probes

HTTP requests come in several types. Typically, you should use the HTTP Get type, but you can create other types depending on your needs. (See "Selecting the HTTP Request Type" on page 9-21.)

For example, you can create an HTTP raw probe and then configure a string that executes a certain command on the server. You can then set the probe to fail the test if the server does not return the correct information. (See "Requiring Particular Information from the Web Server" on page 9-23).

In other words, unlike ICMP echo and TCP connect probes, HTTP request probes can demand a specific response instead of any valid response. You can require the Web server's response to include one or both of the following:

- an acceptable status—HTTP defines various statuses for responses such as OK, Forbidden, and Not Found.
- a certain regular expression (a string of characters)

If the server's response does not conform to your settings, the probe fails the test. For example, an HTTP request probe sends a Get packet to the destination server, but the server does not return a success status, so the probe considers that test failed. Or, the response does not include a keyword from your Web site's correct page, so the test fails.

A final optional setting for HTTP request probes is the absolute path for Web pages (the default path is the forward slash [/]). (See "Specifying the Web Server's Absolute Path" on page 9-24.)

Selecting the HTTP Request Type. You can choose from three types of requests:

- Get—An HTTP Get packet is the standard request sent to a Web server, and this is the default probe type. Because the probe sends the same type of request that a typical workstation would send, it is well-suited to testing a Web server's actual performance. However, because the server returns an entire response packet with all the data that it would send to a user who actually wanted to view the Web page, this probe adds overhead to the network.
- Head—A head packet requests the same response as a Get packet, but requests that the Web server send only the heading. Using this request type makes the probe more courteous; the server is not forced to congest the network with a large packet that the router does not really need to see. This type is ideal for testing simple connectivity to the Web server because it monitors the Web server's ability to return information, although not the actual information returned. You can also monitor the status for the response, as this status is included in the header.
- Raw—Use the Raw type if you want to customize the request. (For the purposes of monitoring routes, such customizing is rarely necessary.) You must then enter HTTP commands to configure the probe to request specific information from the server. (See "Configuring a Raw String of HTTP Commands" on page 9-24).

To select the HTTP request type, from the network monitor probe configuration mode context, enter:

Syntax: type [get | head | raw]

Requiring a Particular Status in the Web Server's Response. Sometimes, when you are monitoring an Web server, you want to know not just that the server is up and connected, but also that it is successfully returning Web pages to the users who request them. One way to verify this is to configure the probe to make the success of a test depend on the status indicated in the HTTP response header.

A Web server uses the status code to inform the HTTP client what type of response it is sending. Ideally, the status indicates success, and the response includes the Web page that the client has requested. But sometimes problems arise, and the status code indicates the type of problem.

Table 9-2 lists the HTTP response status codes defined by RFC 2616.

Table 9-2. HTTP Response Status Codes

Response Class	Response Status Code
1xx—Informational (Request received, continuing process.)	• 100: Continue • 101: Switching Protocols
2xx—Success (The action was successfully received, understood, and accepted.)	• 200: OK • 201: Created • 202: Accepted • 203: Non-Authoritative Information • 204: No Content • 205: Reset Content • 206: Partial Content • 207: Multi-Status
3xx—Redirection (The client must take additional action to complete the request.)	• 300: Multiple Choices • 301: Moved Permanently • 302: Moved Temporarily (HTTP/1.0) • 302: Found (HTTP/1.1) • 303: See Other (HTTP/1.1) • 304: Not Modified • 305: Use Proxy • 306: (No longer used, but reserved.) • 307: Temporary Redirect
4xx—Client Error (The request contains bad syntax or cannot be fulfilled.)	• 400: Bad Request • 401: Unauthorized • 402: Payment Required • 403: Forbidden • 404: Not Found • 405: Method Not Allowed • 406: Not Acceptable • 407: Proxy Authentication Required • 408: Request Timeout • 409: Conflict • 410: Gone • 411: Length Required • 412: Precondition Failed • 413: Request Entity Too Large • 414: Request-URL Too Long • 415: Unsupported Media Type • 416: Requested Range Not Satisfiable • 417: Expectation Failed • 449: Retry With

Response Class	Response Status Code
5xx—Server Error (The server failed to fulfill an apparently valid request.)	• 500: Internal Server Error • 501: Not Implemented • 502: Bad Gateway • 503: Service Unavailable • 504: Gateway Timeout • 505: HTTP Version Not Supported • 509: Bandwidth Limit Exceeded

As shown in Table 9-2, status codes in the 200 to 299 range indicate success. You might configure a probe to accept only responses with these codes as successful. Or you might configure the probe to accept any status that does not actually indicate a *server* error.

To specify the acceptable status codes, from the network monitor probe configuration mode context, enter:

Syntax: expect status <*minimum*> [<*maximum*>]

You can specify a single acceptable status by entering only the ***<minimum>*** value. Or you can specify a range of acceptable statuses by entering both ***<minimum>*** and ***<maximum>*** values; the range is inclusive. For example, a valid range for a status indicating success would be from 200 to 299.

Requiring Particular Information from the Web Server. You can require the Web server to return a message that matches a particular expected string. For example, to verify that the probe reaches the correct Web server, you configure the probe to test not just that the server sends *any* content, but that it sends the *correct* content.

The characters that the request must include is specified by a set of keywords, known as a *regular expression*. To configure the regular expression, from the network monitor probe configuration mode context, enter:

Syntax: expect regex "<*expression*>"

Enclose the expression in quotation marks so that you can enter spaces.

For example, to allow the test to pass only if the words "Welcome to ProCurve" is found in the Web server's response message, enter:

```
ProCurve(config-probe-MyWebServer)# expect regex "Welcome to ProCurve"
```

Specifying the Web Server's Absolute Path. Most Web servers use the default path: forward slash (/). However, some use a different path such as */home/index.htm*. You do not want a test to fail because the probe sent a faulty request.

To specify the correct absolute path for the destination server, from the network monitor probe configuration mode context, enter:

Syntax: absolute-path <*path*>

Remember to include the forward slash (/). For example, enter:

```
ProCurve(config-probe-MyWebServer)# absolute-path /home/index.htm
```

Configuring a Raw String of HTTP Commands. Complete this task only when you have selected Raw for the HTTP request type.

A raw string is a series of HTTP commands that the ProCurve Secure Router places in the data portion of the probe packet. For example, some Web servers store information about network connectivity. You can configure the probe to retrieve the file that stores this information.

To create the raw string, from the network monitor probe configuration mode context, enter:

Syntax: raw-string

The CLI moves you to a prompt from which you enter the HTTP commands. Typically, you should begin the script with this command: **GET /**. When you have finished entering the commands, specify the correct HTTP version (for example, **HTTP/1.1**) and enter two carriage returns (**\r\n\r\n**). To leave the prompt from which you configure the string, type **exit** and press **Enter**.

Table 9-3 lists variables that you can specify in the HTTP commands. The ProCurve Secure Router automatically replaces the variable with information about itself.

Table 9-3. Variables for Raw Strings in HTTP Packets

Variable	Replaced By	Corresponding HTTP Variable
$SYSTEM_NAME	Router's hostname	hostname
$SYSTEM_SERIAL_NUMBER	Router's serial number	serial
$SYSTEM_DESCRIPTION	Router's product name and part number	description
$SYSTEM_SOFTWARE_VERSION	Router's current software version	firmware

Be careful when configuring raw strings. The CLI does not stop you from inputting incorrect commands.

You can use raw HTTP probes to submit information to the remote Web server. For example, you might want to notify the remote network whether the router is currently using a primary or a backup link. For each link, configure an HTTP raw probe that submits this notification. For example, the raw string that follows configures the MyPrimaryWeb probe to notify the remote Web server that the primary connection on this particular ProCurve Secure Router is active.

```
ProCurve(config-probe-MyPrimaryWeb)# raw-string
#GET /log.php?hostname=$SYSTEM_NAME
&serial=$SYSTEM_SERIAL_NUMBER&link=Primary HTTP/1.0
#\r\n
#\r\n
#exit
```

You would create a similar string for the MyBackupWeb probe:

```
ProCurve(config-probe-MyBackupWeb)# raw-string
#GET /log.php?hostname=$SYSTEM_NAME
&serial=$SYSTEM_SERIAL_NUMBER&link=Backup HTTP/1.0
#\r\n
#\r\n
#exit
```

You would give each probe the same destination, but set up PBR to route one probe over the primary connection and one probe over the backup. The remote Web server logs which probe actually reaches it.

Activating and Shutting Down the Probe

By default, a probe is shut down and does not initiate any tests until you activate it. Once you activate the probe, it begins running tests—and consuming bandwidth—even though it does not actually have any effect until you have associate it with a track. Therefore, you might want to configure the track before you activate the probe. When shutdown, the probe's status is always Pass, so the track will not fail in the time that it takes you to activate the probe.

To activate the prove, from the network monitor probe configuration mode context, enter:

Syntax: no shutdown

To shut down the probe again (returning it to a perpetual Pass status), enter:

Syntax: shutdown

Configuring Tracks

A track can monitor:

- remote devices, such as a main office's or service provider's router
- servers running a TCP application
- Web servers

Without network monitoring, the router has no means for detecting when a static route fails at a remote point. The primary purpose of network monitoring on the ProCurve Secure Router is to remove failed static routes so that the router can begin forwarding traffic over a backup connection.

A track can enable and disable:

- static routes
- default routes and other configuration information received from a DHCP server
- default routes received with a negotiated address

You must complete these tasks to configure the track:

1. Create and name the track.
2. Specify the track's probes.

Optionally, you can configure the track to log changes in probes' statuses.

Finally, you should:

3. Associate the track with the route or routes that it must monitor.

Creating a Track

To create a track, from the global configuration mode context, enter:

Syntax: track <*name*>

Give the track a name that helps you to remember its purpose. For example, if the track monitors your company's Web server, you might enter:

```
ProCurve(config)# track MyWebServer
```

For a track that monitors the route to one of the subnets at your company's branch office in Grenoble, you might enter:

```
ProCurve(config)# track Grenoble2
```

In either case, you move into the network monitor track configuration mode context, and the prompt reflects this change:

```
ProCurve(config-track-MyWebServer)#
```

You can configure multiple tracks, even in the hundreds. Practically, however, you would not want your router to run this many active tracks and probes.

Specifying the Track's Probes

A track can use either one or two probes to monitor the network. Reasons to specify two probes include:

- The track is monitoring the route to a network that includes two possible default gateways.
- You want the track to monitor more than one vital remote network server.
- You want the track to monitor the performance of a remote network server in more than one way. For example, one probe might simply test connectivity to the server, while the second probe might verify that the server returns responses in a timely manner.

When you specify one probe only, the track will fail when that probe fails.

When you specify two probes, you can configure the track to pass when one of its probes passes or to pass only when both probes pass.

In the first example listed above (monitoring a network with two possible gateways), you would configure the track to pass when either probe passes: the track should not remove a route that it is still valid.

In the second case—monitoring two vital services—you would set the track to pass only if *both* necessary services are available.

If you were using a track only to log changes in performance, you would probably set the track to pass only when both probes pass. In that way, you are alerted whenever a problem occurs with a network service.

To specify one probe, from the network monitor track configuration mode context, enter:

Syntax: test probe <*name*>

Replace ***<name>*** with the probe's name, which is case sensitive.

To specify two probes, of which both must pass for the track to pass, enter:

Syntax: test probe <*name*> and probe <*name*>

To specify two probes, of which only one must pass for the track to pass, enter:

Syntax: test probe <*name*> or probe <*name*>

Table 9-4 shows the track's state based on statuses of its probes.

Table 9-4. Track and Probe States

Command Type	Probe 1 Status	Probe 2 Status	Track Status
Single probe	Pass		Pass
	Fail		Fail
Or	Pass	Pass	Pass
	Pass	Fail	Pass
	Fail	Pass	Pass
	Fail	Fail	Fail
And	Pass	Pass	Pass
	Pass	Fail	Fail
	Fail	Pass	Fail
	Fail	Fail	Fail

Note

If you want, you can associate the same probe with multiple tracks.

Configuring a Dampening Interval

Removing a route from the routing table is a relatively drastic action. If, in addition to static routes, your ProCurve Secure Router also runs a routing protocol and that protocol redistributes static routes, removing a route can have repercussions throughout a network.

A dampening interval configures a track to delay its state change in response to its probes' state changes. For example, a dampening interval of 10 seconds forces a track to stay in the Pass state for 10 seconds after the associated probe fails. In effect, this interval delays a track from removing a monitored route (or creating a log). If the probe (or probes) again changes state before the dampening interval expires, the track will not take action at all. Thus the dampening interval prevents rapid fluctuations in the routing table and excess logs. (Remember that the dampening interval applies to all state changes—Fail to Pass as well as Pass to Fail.)

You should configure a dampening interval when you have found that a probe sometimes returns false positives (that is, fails without due cause) or when the consequences of the track taking the wrong action are more serious then the consequences of delaying the action slightly.

To specify a dampening interface, from the network monitor track configuration mode context, enter:

Syntax: dampening-interval <*seconds*>

You can set the interval from 1 to 4,294,967,295 seconds. The default setting is 1 second.

Enabling a Track to Log Changes

You can enable a track to create a log every time that its state changes, alerting you to changing network conditions. Enabling logs is a good idea both for tracks that monitor routes and tracks that do not monitor routes.

For tracks that monitor routes, logging changes lets you know when a network becomes unavailable and when a route is removed.

For tracks that monitor server or network performance, the logged changes are the primary purpose of the track. When a track monitors a particular remote server's performance, you may not want it to remove a route based on its findings. The fact that one server in a destination network is experiencing a problem does not mean that all servers in that network are. Instead, you want to track when problems occur. The log-changes command allows you to do so.

To enable the track to log changes, from the network monitor track configuration mode context, enter:

Syntax: log-changes

The **log-changes** command is like the **show events** command: it displays in the running-config file, and (as long as you save this configuration to the startup-config file) it persists when the ProCurve Secure Router is rebooted.

After you enter this command, the ProCurve Secure Router takes these actions when the track's state changes:

- displays a log to the terminal
- saves the log to the event history
- if you have enabled log forwarding, forwards the log to a syslog server

To enable logging forwarding, enter these commands from the global configuration mode context:

Syntax: logging forwarding on
Syntax: logging forwarding receiver-ip <*A.B.C.D*>

Track logs have a priority of notice (level 3), so make sure that the priority for logged events and forwarded logs is set at this level or above.

Syntax: logging forwarding priority-level notice

See *Chapter 4: ProCurve Secure Router OS Firewall—Protecting the Internal, Trusted Network* for more information.

Activating and Shutting Down a Track

Unlike a probe, a track is *active* by default, which means that:

- the track's state depends on the status of its probe or probes
- the track logs state changes to the terminal and saves those logs to the event history (if configured to do so)
- the track removes faulty routes and reinstates repaired routes (if configured to do so)

You can shut down the track, which, also unlike the probe, forces it into a permanent *Fail* state. Be careful: if the track is monitoring a route, it will then remove that route. If you want the track to stop monitoring the network, but remain in a Pass state, shut down its probes instead.

To shut down a track, from the network monitor track configuration mode context, enter:

Syntax: shutdown

To reactivate the track, enter:

Syntax: no shutdown

Configuring the Track's Action—Associating the Track with a Route

When you use network monitoring to control routes, you must associate a track with a route.

You can associate a track with:

- a static route
- a default route received from a DHCP server
- a default route received with a negotiated address

Associating a Track with a Static Route

When you associate a static route with a track, the route's presence in the routing table depends on the track's state (which, in turn, depends on probe tests). If the track's state is Pass, the route remains in the table and is used to forward traffic. If the track's state becomes Fail, the track removes the route from the table. And, when the track's status changes back to Pass, the route is reinstated.

Note

A track can reinstate a route that once again becomes valid. Remember to configure a route map to forward the probe traffic out the primary interface so that the probe is reinstated only when the primary route becomes valid.

Monitoring and controlling the route serves these functions:

- If your router knows an alternate route for the traffic, it can begin using that route instead of the failed one. (This alternate route might be a floating static route or a load sharing route.)

  Sometimes, of course, the alternate route is no longer valid, either, particularly if the loss of connectivity arises from a failed network connection close to the ultimate destination. You can configure one track to monitor one route and one track to monitor the other route so that no matter what the problem is, the routing table remains accurate.

- If your router does not know an alternate route for the traffic, at least the traffic does not consume any of the relatively limited bandwidth on a WAN connection.

You associate a route with a track with the same command with which you create the route. From the global configuration mode context, enter:

Syntax: ip route <*destination network A.B.C.D*> <*subnet mask | /prefix length*> <*next-hop A.B.C.D | forwarding interface ID*> [<*administrative distance*>] track <*name*>

Replace ***<name>*** with the track's name. For information about other options, see the *Basic Management and Configuration Guide*, *Chapter 11: IP Routing—Configuring Static Routes*.

If the particular route has already been entered, you can add the track by reentering the command and including the **track *<name>*** keyword. Make sure that you match the existing route exactly using the same forwarding interface or next-hop address. Otherwise, the ProCurve Secure Router OS will add the monitored route as a new floating static route and not monitor the existing route.

Associating a Track with a DHCP Default Route

Some service providers require that your ProCurve Secure Router receive a DHCP configuration on the interface that connects to that provider. These configurations can include a variety of settings besides an IP address, one of which is a default route.

Note

See the *Basic Management and Configuration Guide*, *Chapter 13: Dynamic Host Configuration Protocol (DHCP)* for information on the advantages and disadvantages of accepting a default routes; the instructions in this chapter assume that you accept the default route.

You can configure the ProCurve Secure Router to add the default route as a monitored route. The default route now behaves exactly as a monitored static route, in that it only remains in the routing table when the associated track's state is Pass.

You should associate the track with the default route at the same time that you activate the DHCP client on the interface; you perform this task from the configuration mode context of the interface that receives the DHCP address.

These interfaces include:

- Frame Relay subinterfaces
- ATM subinterfaces
- Ethernet interfaces
- PPP interfaces (only when bridging traffic)

The command for activating the client includes a variety of options; you can use any of the options with the **track** option (except **no-default-route** because, of course, the router cannot monitor the route if the interface does not accept it at all). See the *Basic Management and Configuration Guide*, *Chapter 13: Dynamic Host Configuration Protocol (DHCP)* for more information on these options.

To activate the client and monitor the default route, from the interface configuration mode context, enter one of the following commands:

Syntax: ip address dhcp {client-id [ethernet 0/<*port*> | HH:HH:HH:HH:HH:HH] | hostname <*hostname*>} track <*name*> [<*administrative distance*>]

Syntax: ip address dhcp [hostname <*hostname*> | no-default-route | no-domain-name | no-nameservers] track <*name*> [<*administrative distance*>]

Replace ***<name>*** with the appropriate track's name.

Optionally, replace ***<administrative distance>*** with a value between 1 and 255, indicating the route's priority in being added to the routing table. By default, the route has an administrative distance of 1, the highest priority. If you want the default route to act as a backup route, give it a higher administrative distance. The track monitors the route when it is added to the network, ensuring that it is still a good route.

Note

In addition to removing the default route, the track also removes other information received from the DHCP server, such as the DNS server address.

Associating a Track with a Default Route Received with a Negotiated Address

You can configure these router interfaces to receive an IP address from the far end of the link:

- PPP interfaces
- Demand routing interfaces (which always use PPP on the ProCurve Secure Router)

See the *Basic Management and Configuration Guide*, *Chapter 6: Configuring the Data Link Layer Protocol for E1, T1, and Serial Interfaces* and the *Basic Management and Configuration Guide*, *Chapter 8: Configuring Demand Routing for Primary ISDN Modules* for more information.

Just as a DHCP server can send a default route, the far end of the PPP connection can also send a default route with the IP address. When you decide to allow your router to accept that route, it is often a good idea to monitor the route, ensuring that it remains the best default route for your system.

To configure the router to monitor a negotiated default route, from the PPP or demand interface configuration mode context, enter:

Syntax: ip address negotiated track <*name*> [<*administrative distance*>]

Replace ***<name>*** with the appropriate track's name.

The same considerations for setting the administrative distance that are documented in the previous section apply.

Implementing PBR to Route Probe Traffic

You must use PBR to route probe traffic. This requirement is particularly important when you use probes to test routes to ensure that probe packets are always forwarded over the route in question.

If you do not create a PBR route map for probe packets, the router forwards them to their destination according to its routing table. This behavior is undesirable because the routing table may change based on the results of network monitoring, but you always want a probe to use the route that you have configured it to test.

Figure 9-3 shows the correct way to route traffic: network traffic is routed according to the routing table, but probe traffic is always sent over the primary connection.

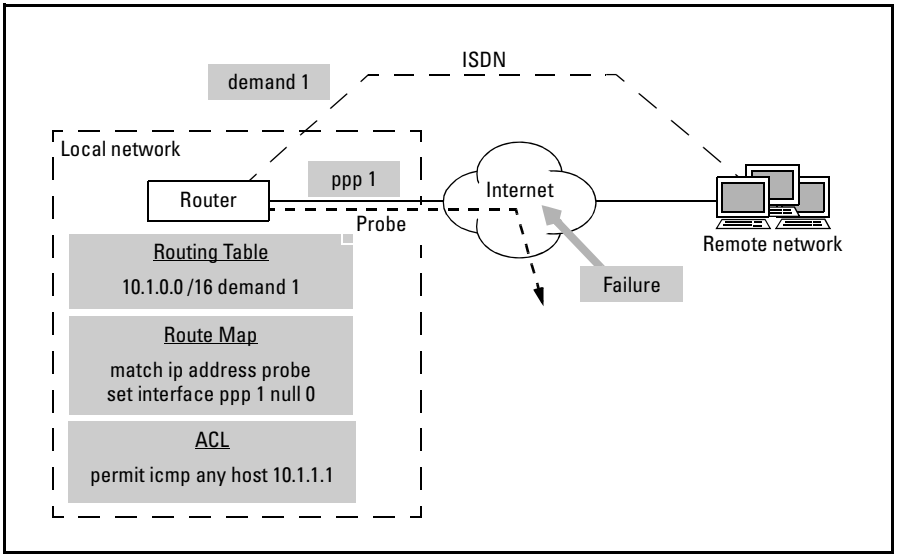

Figure 9-3. Using PBR to Route Probe Traffic

Complete the following steps to configure PBR for probes:

1. Create an extended access control list (ACL) to select each probe's traffic. From the global configuration mode context, enter:

   Syntax: ip access-list extended <*listname*>

2. Configure an ACL entry to permit:
   - packets of the probe's protocol type
   - packets with the probe's destination address or hostname and, for TCP or HTTP probes, destination port

   Syntax: permit tcp any [host <*destination A.B.C.D* | hostname <*hostname*>] [eq <*destination port*>]
   Syntax: permit icmp any [host <*destination A.B.C.D* | hostname <*hostname*>]

3. Create a route map entry. If you have created multiple probes to test different routes, create an entry for each probe. Give each entry the same name, but a different index number. From the global configuration mode context, enter:

   Syntax: route-map <*mapname*> <*index number*>

4. Match the route map entry to the ACL created for the probe in question.

   Syntax: match ip address <*listname*>

5. Set the forwarding interface to match the forwarding interface in the route that the probe tests. Remember to add the null interface at the end of the command so that the router will drop probes that it cannot forward on the correct interface.

   Syntax: set interface <*interface ID*> null 0

   Alternatively, set the next-hop address to match the route's next-hop address.

   Syntax: set ip next-hop <*A.B.C.D*>
   Syntax: set interface null 0

6. After you have configured all necessary entries, apply the route map to local router traffic. Exit to global configuration mode and enter:

   Syntax: ip local policy route-map <*mapname*>

For example, you have created an ICMP echo probe to test connectivity to a remote network (gateway address 10.5.1.1) reached through ATM subinterface 1.1. Enter these commands to configure PBR for the probe:

```
ProCurve(config)# ip access-list extended probe1
ProCurve(config-ext-nacl)# permit icmp any host 10.5.1.1
ProCurve(config-ext-nacl)# exit
ProCurve(config)# route-map probes 10
ProCurve(config-route-map)# match ip address probe1
ProCurve(config-route-map)# set interface atm 1.1 null 0
ProCurve(config-route-map)# exit
ProCurve(config)# ip local policy route-map probes
```

Table 9-5 gives examples of how to match key commands in the ACL and route map to the probe that you have configured to test a route.

Table 9-5. Examples of Commands to Configure PBR for Probe Traffic

Probe destination Command	Monitored Route	ACL permit Command	Route Map set Command
destination www.mycompany.com (ICMP echo probe)	DHCP route on Ethernet interface	permit icmp any hostname www.mycompany.com	set interface eth 0/1 null 0
destination 10.8.4.1 port 25 (TCP connect probe)	ip route 10.8.0.0 /12 ppp 1	permit tcp any host 10.8.4.1 eq 25	set interface ppp 1 null 0
destination 172.16.12.15 port 53 (TCP connect probe)	ip route 0.0.0.0 /0 atm 1.1	permit tcp any host 172.16.12.15 eq 53	set interface atm 1.1 null 0
destination www.mycompany.com (HTTP request probe)	ip route 192.168.0.0 /20 192.168.0.1	permit tcp any hostname www.mycompany.com eq www	set ip next-hop 192.168.0.1 set interface null 0

PBR is discussed in more detail in *Chapter 15: IP Routing—Configuring RIP, OSPF, BGP, and PBR*.

Using NAT with Network Monitoring

You must use the **policy** option with NAT commands when combining NAT with network monitoring. In this way, the ProCurve Secure Router will always translate source addresses to the current correct public address.

Overview

Typically, your ProCurve Secure Router performs NAT on traffic destined to the Internet, translating the source address of all traffic from the LAN to an address recognized by your ISP. Consider the problem that network monitoring can introduce:

1. You configure your router to NAT the source of address of all local traffic to the address on the primary WAN interface.
2. Network monitoring detects a failure and removes the route that forwards traffic through the primary interface.
3. A backup route appears, and the router begins to forward traffic through a secondary interface.
4. NAT continues to translate source addresses to the primary interface's address.
5. The secondary ISP router does not know how to reach the primary interface's address, so return traffic is dropped.
6. Users can no longer connect to the Internet.

To solve this problem, you must create *two* statements in the ACP that implements NAT. One statement translates source addresses to an address allowed by your primary ISP, and the second translates to an address allowed by your secondary ISP. The NAT statements are distinguished by different **policy *<policyname>*** options.

The **policy** keyword forces the ProCurve Secure Router to perform an additional check on traffic that matches the NAT statement:

1. The router determines which interface will forward the traffic, as determined by PBR and routes in the routing table.
2. The router verifies that the ACP applied to the forwarding interface matches the ACP specified by the NAT statement's **policy** option.
3. If the ACPs match, the router translates the packet's source address to the address indicated in the statement.
4. If the ACPs do not match, the router checks the next NAT statement configured in the policy.

In this way, the ProCurve Secure Router can select the correct NAT statement according to the interface that currently forwards Internet traffic.

Configuration Steps

Complete these steps to configure NAT correctly:

1. If you have not already done so, create the following ACLs:
   - one ACL to select traffic permitted on the primary WAN interface
   - one ACL to select traffic permitted on the secondary WAN interface
   - one ACL to select permitted local traffic destined to the Internet over the primary connection
   - optionally, one ACL to select permitted local traffic destined to the Internet over the secondary connection

   You might be able to combine some of these ACLs, depending on how you want to control traffic. In fact, the final ACL is necessary only when you want to allow different levels of access to the Internet depending on the active connection. For example, you could restrict a low-bandwidth secondary connection to mission-critical traffic only.

   See *Chapter 5: Applying Access Control to Router Interfaces* for instructions on configuring ACLs.

2. Create an ACP for the primary WAN interface. From the global configuration mode context, enter:

   Syntax: ip policy-class <*policyname*>

3. Allow the ACL that selects traffic permitted on the primary WAN interface. From the policy class configuration mode context, enter:

   Syntax: allow list <*listname*>

4. Apply the ACP to the primary interface. Starting from the global configuration mode context, enter:

   Syntax: interface <*interface ID*>
   Syntax: access-policy <*policyname*>

5. Repeat steps 2 through 4 to create an ACP to allow traffic on the secondary WAN interface.

6. Create an ACP to perform NAT.

7. Add a NAT statement that translates source addresses to the primary interface's address. Specify the ACL that selects local traffic destined to the Internet and the ACP applied to the primary interface. From the policy class configuration mode context, enter:

   Syntax: nat source list <*listname*> [address <*primary IP address*> | interface <*primary interface ID*>] overload policy <*policyname*>

8. Add a NAT statement that translates source addresses to the secondary interface's address. Specify the ACL that selects local traffic destined to the Internet and the ACP applied to the secondary interface.

9. Apply the NAT ACP to the interface on which local traffic arrives.

Example

The following commands configure NAT for a ProCurve Secure Router that has a primary Internet connection through a cable modem (connected to Ethernet interface 0/2) and uses demand routing for backup:

```
ProCurve(config)# ip firewall
ProCurve(config)# ip access-list standard MatchPrimary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# ip access-list standard MatchSecondary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# ip access-list standard MatchLocal
ProCurve(config-std-nacl)# permit 192.168.1.0 0.0.0.255
ProCurve(config-std-nacl)# exit
ProCurve(config)# ip policy-class Primary
```

```
ProCurve(config-policy-class)# allow list MatchPrimary
ProCurve(config-policy-class)# ip policy-class Secondary
ProCurve(config-policy-class)# allow list MatchSecondary
ProCurve(config-policy-class)# exit
ProCurve(config)# interface ethernet 0/2
ProCurve(config-eth 0/2)# access-policy Primary
ProCurve(config-eth 0/2)# interface demand 1
ProCurve(config-demand 1)# access-policy Secondary
ProCurve(config-demand 1)# exit
ProCurve(config)# ip policy-class NATInside
ProCurve(config-policy-class)# nat source list MatchLocal interface ethernet 0/2
overload policy Primary
ProCurve(config-policy-class)# nat source list MatchLocal interface demand 1 over-
load policy Secondary
ProCurve(config-policy-class)# exit
ProCurve(config)# interface ethernet 0/1
ProCurve(config-eth 0/1)# access-policy NATInside
```

Disabling the RPF Check

The ProCurve Secure Router OS firewall checks incoming traffic and determines whether it has arrived on a valid interface by looking up the source address in the routing table. While network monitoring changes the active route, traffic may seem to be arriving on an invalid interface. You must disable this check so that the firewall does not drop traffic.

You disable the RPF check on a particular ACP. When you apply that ACP to an interface, the router forwards incoming traffic allowed by the ACP regardless of whether this traffic seems to arrive on the correct interface.

If you have properly set up NAT, you have already created ACPs to control incoming traffic on the primary and secondary WAN interfaces. (See "Using NAT with Network Monitoring" on page 9-37.) Enter this command, from the global configuration mode context, to disable the RPF check on these two ACPs:

Syntax: no ip policy-class <*policyname*> rpf-check

If you have not set up NAT, you must complete these steps:

1. Create ACLs:
   - one ACL to select traffic permitted on the primary interface
   - one ACL to select traffic permitted on the secondary interface
   - alternatively, one ACL that permits all traffic

   See *Chapter 5: Applying Access Control to Router Interfaces* for instructions on configuring ACLs.

2. Create ACPs:
   - one ACP to control traffic incoming on the primary interface
   - one ACP to control traffic incoming on the secondary interface

   Allow the appropriate ACL. See *Chapter 5: Applying Access Control to Router Interfaces* for instructions on configuring ACPs.

3. Disable the RPF check on the ACPs.

   Enter this command from the global configuration mode context:

   Syntax: no ip policy-class <*policyname*> rpf-check

4. Apply the ACPs to the primary and backup interfaces. The firewall should be enabled.

For example, enter these commands:

```
ProCurve(config)# ip access-list standard MatchPrimary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# exit
ProCurve(config)# ip access-list standard MatchSecondary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# exit
ProCurve(config)# ip policy-class Primary
ProCurve(config-policy-class)# allow list MatchPrimary
ProCurve(config-policy-class)# exit
ProCurve(config)# no ip policy-class Primary rpf-check
ProCurve(config)# ip policy-class Backup
ProCurve(config-policy-class)# allow list MatchSecondary
ProCurve(config-policy-class)# exit
ProCurve(config)# no ip policy-class Backup rpf-check
ProCurve(config)# ip firewall
ProCurve(config)# interface atm 1.1
ProCurve(config-atm 1.1)# access-policy Primary
ProCurve(config-atm 1.1)# interface demand 1
ProCurve(config-demand 1)# access-policy Backup
```

Examples of Network Monitoring

This section provides examples of how you can configure probes and tracks to:

- monitor connectivity to the Internet and initiate a backup connection should the primary connection fail
- monitor static routes to remote networks and initiate a backup connection should the primary route fail
- monitor connectivity to a mission-critical remote server
- monitor network congestion and the performance of TCP servers, such as FTP, Web, or email servers
- inform your company's Web server whether a secondary or backup connection is being used

Each example includes all of the commands necessary for configuring that type of network monitoring.

Monitor Connectivity to the Internet

In this scenario, Company A maintains two Internet connections:

- The ProCurve Secure Router's Ethernet 0/2 interface connects to a cable modem. This is the primary Internet connection. The Ethernet interface receives an IP address, a default route, and a DNS server addresses through DHCP.
- The router uses demand routing to initiate an ISDN connection to another ISP only when the primary connection fails.

Company A's ProCurve Secure Router must monitor the primary Internet connection because the Ethernet connection to the modem might remain good even when the router cannot actually reach the Internet.

Figure 9-4 illustrates the Company A WAN.

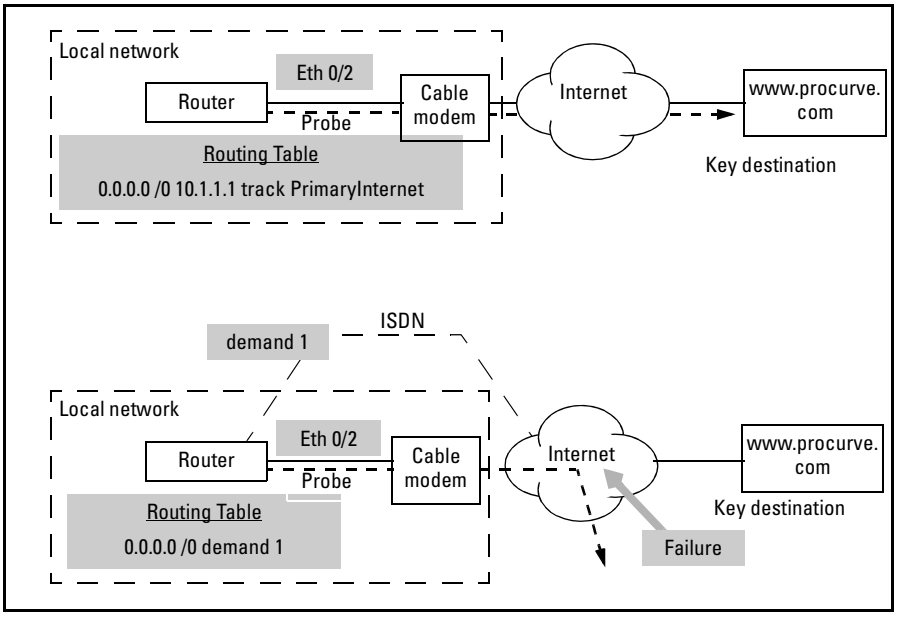

Figure 9-4. Example of Monitoring an Internet Connection

Follow these steps:

1. Configure an ICMP echo probe to test connectivity to the key destination.

   ```
   ProCurve(config)# probe Ping icmp-echo
   ProCurve(config-probe-Ping)# destination www.procurve.com
   ProCurve(config-probe-Ping)# period 10
   ProCurve(config-probe-Ping)# tolerance consecutive-failures 3
   ProCurve(config-probe-Ping)# no shutdown
   ProCurve(config-probe-Ping)# exit
   ```

2. Create a track to monitor the state of the probe.

   ```
   ProCurve(config)# track PrimaryInternet
   ProCurve(config-track-PrimaryInternet)# test probe Ping
   ProCurve(config-track-PrimaryInternet)# exit
   ```

3. Configure the track to monitor the Ethernet 0/2 interface's DHCP configuration.

   ```
   ProCurve(config)# interface ethernet 0/2
   ProCurve(config-eth 0/2)# ip address dhcp track PrimaryInternet
   ProCurve(config-eth 0/2)# exit
   ```

4. Configure PBR for the probe traffic.

```
ProCurve(config)# ip access-list extended MatchPing
ProCurve(config-ext-nacl)# permit icmp any hostname www.procurve.com
ProCurve(config-ext-nacl)# exit
ProCurve(config)# route-map Probes 10
ProCurve(config-route-map)# match ip address MatchPing
ProCurve(config-route-map)# set interface eth 0/2 null 0
ProCurve(config-route-map)# exit
ProCurve(config)# ip local policy route-map Probes
```

5. Configure NAT. Also activate the firewall and disable the RPF check.

```
ProCurve(config)# ip firewall
ProCurve(config)# ip access-list standard MatchAllPrimary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# ip access-list standard MatchAllSecondary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# ip policy-class Primary
ProCurve(config-policy-class)# allow list MatchAllPrimary
ProCurve(config-policy-class)# ip policy-class Secondary
ProCurve(config-policy-class)# allow list MatchAllSecondary
ProCurve(config-policy-class)# interface ethernet 0/2
ProCurve(config-eth 0/2)# access-policy Primary
ProCurve(config-eth 0/2)# interface demand 1
ProCurve(config-demand 1)# access-policy Secondary
ProCurve(config-demand 1)# exit
ProCurve(config)# no ip policy-class Primary rpf-check
ProCurve(config)# no ip policy-class Secondary rpf-check
ProCurve(config)# ip policy-class NATInside
ProCurve(config-policy-class)# nat source list MatchAllPrimary interface ether-
net 0/2 overload policy Primary
ProCurve(config-policy-class)# nat source list MatchAllSecondary interface
demand 1 overload policy Secondary
ProCurve(config-policy-class)# interface ethernet 0/1
ProCurve(config-eth 0/1)# access-policy NATInside
ProCurve(config-eth 0/1)# exit
```

6. Verify that the floating static route has been created through the demand interface. (See *Chapter 3: Configuring Backup WAN Connections* for instructions on configuring this interface.)

```
ProCurve(config)# ip route 0.0.0.0 /0 demand 1 20
```

Monitor Static Routes to Remote Networks

In this scenario, Company A maintains a large central office and several branch offices. Each branch office connects to the central office through an Asymmetric Digital Subscriber Line (ADSL) connection, and the central office routes traffic among these offices. The Branch Office 1 router knows a static route to each remote office.

As a backup to reaching the remote offices through the central office, the routers can use demand routing and create a temporary ISDN connection to one of the remote offices. Because the Branch Office 1 router uses static routing, you decide to configure network monitoring to test the routes to remote offices and remove routes that fail the test. A floating route through a demand interface will then appear in the routing table, allowing the router to place a call to the correct branch office.

Figure 9-5 illustrates the Company A WAN.

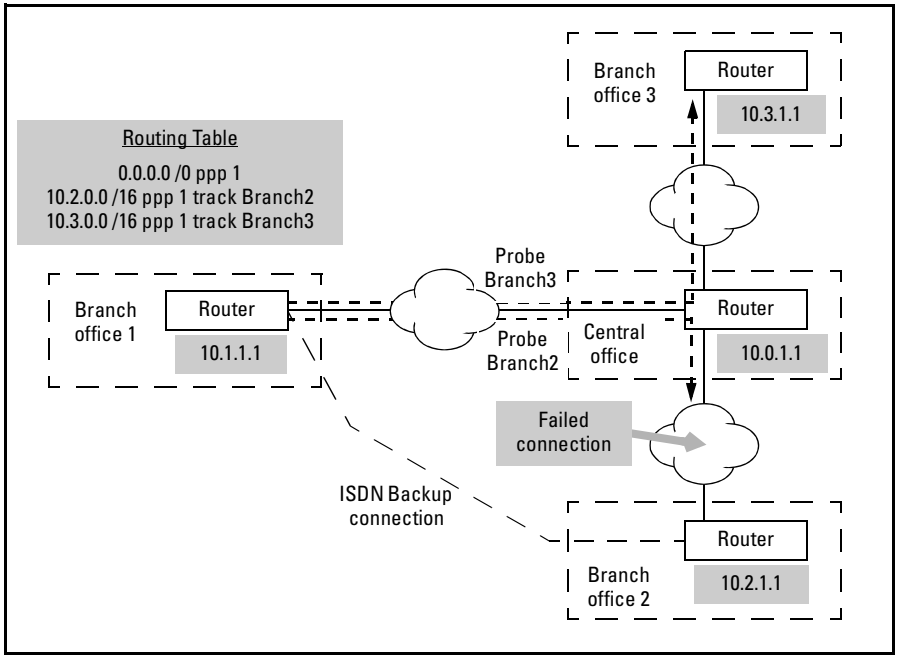

Figure 9-5. Example of Monitoring Static Routes

Follow these steps:

1. Configure one ICMP echo probe for each branch office LAN. Because you are testing connectivity, set the tolerance to consecutive failures, and set the timeout and tolerance high enough to ensure that congestion is not interpreted as a failed connection. Probes will report lost contact with a branch office after two minutes and eighteen seconds

   ```
   ProCurve(config)# probe Branch2 icmp-echo
   ProCurve(config-probe-Branch2)# destination 10.2.1.1
   ProCurve(config-probe-Branch2)# timeout 3000
   ProCurve(config-probe-Branch2)# period 20
   ProCurve(config-probe-Branch2)# tolerance consecutive-failures 6
   ProCurve(config-probe-Branch2)# no shutdown
   ProCurve(config-probe-Branch2)# probe Branch3 icmp-echo
   ProCurve(config-probe-Branch3)# destination 10.3.1.1
   ProCurve(config-probe-Branch3)# timeout 3000
   ProCurve(config-probe-Branch2)# period 20
   ProCurve(config-probe-Branch3)# tolerance consecutive-failures 6
   ProCurve(config-probe-Branch3)# no shutdown
   ProCurve(config-probe-Branch3)# exit
   ```

2. Configure a track to monitor the route to each branch office LAN.

   ```
   ProCurve(config)# track Branch2
   ProCurve(config-track-Branch2)# test probe Branch2
   ProCurve(config-track-Branch2)# log-changes
   ProCurve(config-track-Branch2)# track Branch3
   ProCurve(config-track-Branch3)# test probe Branch3
   ProCurve(config-track-Branch3)# log-changes
   ProCurve(config-track-Branch3)# exit
   ```

3. Create a static route to each branch office and associate it with the correct track.

   ```
   ProCurve(config)# ip route 10.2.0.0 /16 ppp 1 track Branch2
   ProCurve(config)# ip route 10.3.0.0 /16 ppp 1 track Branch3
   ```

4. Configure PBR for the probe traffic. In this case, the probes test routes that use the same forwarding interface, so you can create a single route map entry:

```
ProCurve(config)# ip access-list extended ProbeBranch
ProCurve(config-ext-nacl)# permit icmp any host 10.2.1.1
ProCurve(config-ext-nacl)# permit icmp any host 10.3.1.1
ProCurve(config-ext-nacl)# exit

ProCurve(config)# route-map Probes 10
ProCurve(config-route-map)# match ip address ProbeBranch
ProCurve(config-route-map)# set interface ppp 1 null 0
ProCurve(config-route-map)# exit
ProCurve(config)# ip local policy route-map Probes
```

5. This router has two demand interfaces, one to place a call to each remote office. (See the *Basic Management and Configuration Guide*, *Chapter 8: Configuring Demand Routing for Primary ISDN Modules* for instructions on configuring those interfaces, including selecting the traffic that initiates a call.) Create floating backup routes through those interfaces:

```
ProCurve(config)# ip route 10.2.0.0 /16 demand 1 2
ProCurve(config)# ip route 10.3.0.0 /16 demand 2 2
```

6. The routers at the remote branch offices must also use network monitoring and be configured with the same retries and timeout settings in order for this configuration to function correctly.

Monitor Connectivity to a Mission-Critical TCP Server

This company includes a main office and a branch office, which connect through the Internet. The branch office network administrator wants to monitor the connection to the main office's primary and backup FTP servers. If the connection to both of these servers fails, an ISDN call to the headquarters initiates. (The primary connection can continue forwarding other Internet traffic.)

This scenario requires two probes:

- one probe to monitor the connection to the primary FTP server (FTPServer1)
- one probe to monitor the connection to the backup FTP server (FTPServer2)

The probes will use the default TCP timeout (10 seconds), but a longer period to minimize overhead.

This scenario requires a single track, which will test whether the connection exists to at least one of the FTP servers. If the track fails, it removes the primary route to the headquarters. The ISDN connection initiates the next time that interesting traffic (which you could define as any traffic or as traffic destined to the FTP servers) arrives on the demand interface. See Figure 9-6.

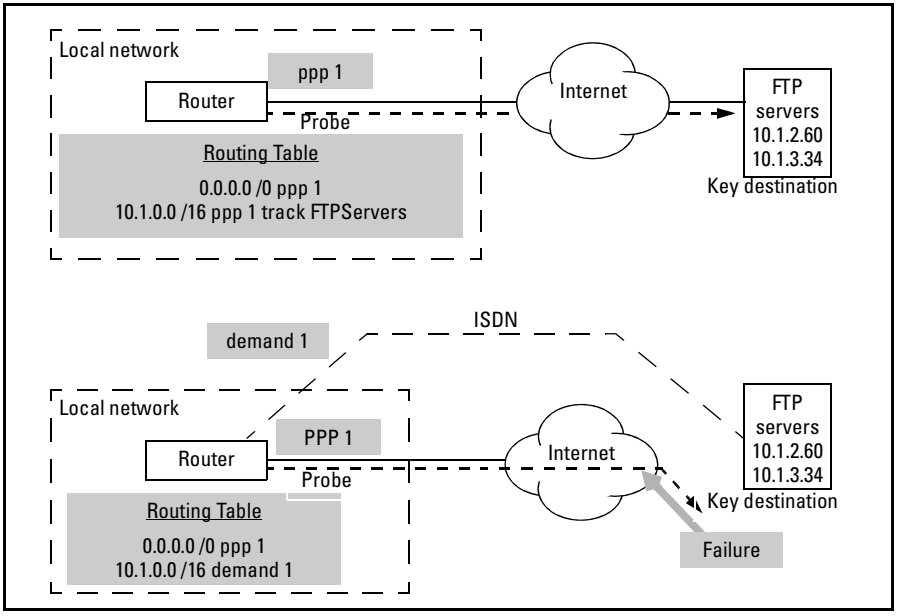

Figure 9-6. Monitoring Connectivity to a Mission-Critical Server

Follow these steps:

1. Configure the two probes.

```
ProCurve(config)# probe FTPServer1 tcp-connect
ProCurve(config-probe-FTPServer1)# destination 10.1.2.60 port 21
ProCurve(config-probe-FTPServer1)# tolerance rate-of-failure 2 3
ProCurve(config-probe-FTPServer1)# period 70
ProCurve(config-probe-FTPServer1)# no shutdown
ProCurve(config-probe-FTPServer1)# probe FTPServer2 tcp-connect
ProCurve(config-probe-FTPServer2)# destination 10.1.3.34 port 21
ProCurve(config-probe-FTPServer2)# tolerance rate-of-failure 2 3
ProCurve(config-probe-FTPServer2)# period 70
ProCurve(config-probe-FTPServer2)# no shutdown
ProCurve(config-probe-FTPServer2)# exit
```

2. Configure the track.

```
ProCurve(config)# track FTPServers
ProCurve(config-track-FTPServers)# test probe FTPServer1 or probe FTPServer2
ProCurve(config-track-FTPServers)# dampening-interval 10
ProCurve(config-track-FTPServers)# log-changes
ProCurve(config-track-FTPServers)# exit
```

3. Configure routes. Enable the track to monitor the primary route.

```
ProCurve(config)# ip route 0.0.0.0 /0 ppp 1
ProCurve(config)# ip route 10.1.0.0 /16 ppp 1 track FTPServers
ProCurve(config)# ip route 10.1.0.0 /16 demand 1 20
```

4. Configure PBR for the probes. These probes use the same forwarding interface, so you can configure a single route map entry.

```
ProCurve(config)# ip access-list extended ProbeFTP
ProCurve(config-ext-nacl)# permit tcp any host 10.1.2.60 eq 21
ProCurve(config-ext-nacl)# permit tcp any host 10.1.3.34 eq 21
ProCurve(config-ext-nacl)# exit
ProCurve(config)# route-map Probes 10
ProCurve(config-route-map)# match ip address ProbeFTP
ProCurve(config-route-map)# set interface ppp 1 null 0
ProCurve(config-route-map)# exit
ProCurve(config)# ip local policy route-map Probes
```

5. Configure NAT and disable reverse path forwarding. Configure NAT on both the Internet connection and the demand routing connection. Make sure that the HQ network has a route to your router's demand interface.

```
ProCurve(config)# ip firewall
ProCurve(config)# ip access-list standard MatchAllPrimary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# ip access-list standard MatchAllBackup
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# ip policy-class Primary
ProCurve(config-policy-class)# allow list MatchAllPrimary
ProCurve(config-policy-class)# ip policy-class Backup
ProCurve(config-policy-class)# allow reverse list MatchAllBackup
ProCurve(config-policy-class)# exit
ProCurve(config)# interface ppp 1
ProCurve(config-ppp 1)# access-policy Primary
ProCurve(config-ppp 1)# interface demand 1
ProCurve(config-demand 1)# access-policy Backup
ProCurve(config-demand 1)# exit
ProCurve(config)# no ip policy-class Primary rpf-check
ProCurve(config)# no ip policy-class Backup rpf-check
```

```
ProCurve(config)# ip policy-class NATInside
ProCurve(config-policy-class)# nat source list MatchAllPrimary interface ppp 1
overload policy Primary
ProCurve(config-policy-class)# nat source list MatchAllBackup interface ppp 1
overload policy Backup
ProCurve(config)# interface ethernet 0/1
ProCurve(config-eth 0/1)# access-policy NATInside
ProCurve(config-eth 0/1)# exit
```

Monitor Network Congestion and the Performance of Servers

In this scenario, you want to monitor congestion at two often-used and mission-critical remote servers. You will send the information that the tracks collect to your service provider to document the connection's performance.

Two probes monitor each server: one probe tests for optimal performance, and the other probe tests for minimum adequate performance. What passes for optimal and minimal performance depends, of course, on your company's needs and normal network conditions, and you should set the probes' timeout and rate of failure tolerance accordingly. The commands below merely provide one example.

One track monitors each probe, creating logs when performance changes, but does not affect routing.

Follow these steps:

1. Configure the pair of probes for each network point to be monitored. In this example, you will use HTTP Get probes for testing one key destination, a remote Web server, and TCP connect probes for testing the second key destination, a remote email server.

   ```
   ProCurve(config)# probe OptimalWeb http-request
   ProCurve(config-probe-OptimalWeb)# destination www.mycompany.com
   ProCurve(config-probe-OptimalWeb)# timeout 500
   ProCurve(config-probe-OptimalWeb)# tolerance rate-of-failure 2 10
   ProCurve(config-probe-OptimalWeb)# no shutdown
   ProCurve(config-probe-OptimalWeb)# exit
   ProCurve(config)# probe MinimalWeb http-request
   ProCurve(config-probe-MinimalWeb)# destination www.mycompany.com
   ProCurve(config-probe-MinimalWeb)# timeout 2500
   ProCurve(config-probe-MinimalWeb)# tolerance rate-of-failure 14 16
   ProCurve(config-probe-MinimalWeb)# no shutdown
   ProCurve(config-probe-MinimalWeb)# exit
   ProCurve(config)# probe OptimalEmail tcp-connect
   ProCurve(config-probe-OptimalEmail)# destination www.mycompany.com port 25
   ```

```
ProCurve(config-probe-OptimalEmail)# timeout 500
ProCurve(config-probe-OptimalEmail)# tolerance rate-of-failure 4 10
ProCurve(config-probe-OptimalEmail)# no shutdown
ProCurve(config-probe-OptimalEmail)# exit
ProCurve(config)# probe MinimalEmail tcp-connect
ProCurve(config-probe-MinimalEmail)# destination www.mycompany.com port 25
ProCurve(config-probe-MinimalEmail)# timeout 2500
ProCurve(config-probe-MinimalEmail)# tolerance rate-of-failure 14 16
ProCurve(config-probe-MinimalEmail)# no shutdown
ProCurve(config-probe-MinimalEmail)# exit
```

2. Configure the tracks that monitor changes in performance.

```
ProCurve(config)# track OptimalWeb
ProCurve(config-track-OptimalWeb)# test probe OptimalWeb
ProCurve(config-track-OptimalWeb)# log-changes
ProCurve(config-track-OptimalWeb)# track MinimalWeb
ProCurve(config-track-MinimalWeb)# test probe MinimalWeb
ProCurve(config-track-MinimalWeb)# log-changes
ProCurve(config-track-MinimalWeb)# track OptimalEmail
ProCurve(config-track-OptimalEmail)# test probe OptimalEmail
ProCurve(config-track-OptimalEmail)# log-changes
ProCurve(config-track-OptimalEmail)# track MinimalEmail
ProCurve(config-track-MinimalEmail)# test probe MinimalEmail
ProCurve(config-track-MinimalEmail)# log-changes
ProCurve(config-track-MinimalEmail)# exit
```

3. Enable logs to be forwarded to a syslog server.

```
ProCurve(config)# logging forwarding on
ProCurve(config)# logging forwarding receiver-ip 10.12.1.43
ProCurve(config)# logging forwarding priority-level notice
```

Submit Information to a Remote Web Server

This scenario is similar to the first scenario. In this case, the local ProCurve Secure Router monitors the connection to the Internet. If it cannot reach the primary ISP (IP address 10.1.1.1) through an Ethernet connection to a cable modem, a backup ISDN connection initiates. The router also periodically notifies the headquarters Web server (IP address 10.5.4.20) which connection is currently in use.

This scenario requires three probes:

- Ping monitors connectivity to the Internet.
- HTTPPrimary informs the Web server when the router is using the primary link.
- HTTPBackup informs the Web server when the router is using the backup link.

This scenario requires one track to monitor the state of the RemotePing probe and control the primary static route. The HTTP probes do not require a track.

This scenario requires three PBR route map entries:

- Probes 10 routes RemotePing probe traffic out the primary interface.
- Probes 20 routes HTTPPrimary probe traffic out the primary interface.
- Probes 30 routes HTTPBackup probe traffic out the backup interface. (Remember to add the probe traffic to the interesting traffic for demand routing. See "Matching the Interesting Traffic" on page 3-24 in *Chapter 3: Configuring Backup WAN Connections*.)

Follow these steps to configure your router:

1. Create the probes. The HTTPPrimary and HTTPBackup probes must use different source ports so that PBR can distinguish their traffic. The raw strings for the HTTP probes configure them to submit necessary information to the remote Web server.

```
ProCurve(config)# probe Ping icmp-echo
ProCurve(config-probe-Ping)# destination 10.1.1.1
ProCurve(config-probe-Ping)# period 10
ProCurve(config-probe-Ping)# tolerance consecutive-failures 5
ProCurve(config-probe-Ping)# no shutdown
ProCurve(config-probe-Ping)# probe HTTPPrimary http-request
ProCurve(config-probe-HTTPPrimary)# destination 10.5.4.20
ProCurve(config-probe-HTTPPrimary)# source-port 5150
ProCurve(config-probe-HTTPPrimary)# type raw
ProCurve(config-probe-HTTPPrimary)# raw-string
#GET /log.php?hostname=$SYSTEM_NAME
&serial=$SYSTEM_SERIAL_NUMBER&link=Primary HTTP/1.0
#\r\n
#\r\n
#exit
ProCurve(config-probe-HTTPPrimary)# period 300
ProCurve(config-probe-HTTPPrimary)# tolerance consecutive-failures 3
ProCurve(config-probe-HTTPPrimary)# no shutdown
ProCurve(config-probe-HTTPPrimary)# exit
```

```
ProCurve(config)# probe HTTPSecondary http-request
ProCurve(config-probe-HTTPSecondary)# destination 10.5.4.20
ProCurve(config-probe-HTTPSecondary)# source-port 5151
ProCurve(config-probe-HTTPSecondary)# type raw
ProCurve(config-probe-HTTPSecondary)# raw-string
#GET /log.php?hostname=$SYSTEM_NAME
&serial=$SYSTEM_SERIAL_NUMBER&link=Backup HTTP/1.0
#\r\n
#\r\n
#exit
ProCurve(config-probe-HTTPSecondary)# period 300
ProCurve(config-probe-HTTPSecondary)# tolerance consecutive-failures 3
ProCurve(config-probe-HTTPSecondary)# no shutdown
ProCurve(config-probe-HTTPSecondary)# exit
```

2. Configure a track to monitor the connection to the headquarters and to control the default route received from the ISP.

```
ProCurve(config)# track Remote
ProCurve(config-track-Remote)# test probe Ping
ProCurve(config-track-Branch2)# exit
```

3. Enable the track to control the route that the Ethernet interface receives from the ISP.

```
ProCurve(config)# interface ethernet 0/2
ProCurve(config-eth 0/2)# ip address dhcp track Remote
ProCurve(config-eth 0/2)# exit
```

4. Configure PBR for the probe traffic.

```
ProCurve(config)# ip access-list extended MatchPing
ProCurve(config-ext-nacl)# permit icmp any host 10.1.1.1
ProCurve(config-ext-nacl)# exit
ProCurve(config)# ip access-list extended HTTPPrimary
ProCurve(config-ext-nacl)# permit tcp any eq 5150 host 10.5.4.20 eq www
ProCurve(config-ext-nacl)# exit
ProCurve(config)# ip access-list extended HTTPSecondary
ProCurve(config-ext-nacl)# permit tcp any eq 5151 host 10.5.4.20 eq www
ProCurve(config-ext-nacl)# exit
ProCurve(config)# route-map Probes 10
ProCurve(config-route-map)# match ip address MatchPing
ProCurve(config-route-map)# set interface eth 0/2 null 0
ProCurve(config-route-map)# route-map Probes 20
ProCurve(config-route-map)# match ip address HTTPPrimary
ProCurve(config-route-map)# set interface eth 0/2 null 0
ProCurve(config-route-map)# route-map Probes 30
ProCurve(config-route-map)# match ip address HTTPSecondary
```

```
ProCurve(config-route-map)# set interface demand 1 null 0
ProCurve(config-route-map)# exit
ProCurve(config)# ip local policy route-map Probes
```

5. Modify the ACL that selects Interesting traffic for demand routing so that the HTTPSecondary probe does not bring the ISDN link up. For example, enter:

```
ProCurve(config)# ip access-list extended Interesting
ProCurve(config-ext-nacl)# deny tcp any eq 5151 host 10.5.4.20 eq www
ProCurve(config-ext-nacl)# permit ip any any
ProCurve(config-ext-nacl)# interface demand 1
ProCurve(config-demand 1)# match-interesting list Interesting
ProCurve(config-demand 1)# exit
```

6. Configure NAT and disable reverse path forwarding.

```
ProCurve(config)# ip firewall
ProCurve(config)# ip access-list standard MatchAllPrimary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# ip access-list standard MatchAllSecondary
ProCurve(config-std-nacl)# permit any
ProCurve(config-std-nacl)# exit
ProCurve(config)# ip policy-class Primary
ProCurve(config-policy-class)# allow list MatchAllPrimary
ProCurve(config-policy-class)# ip policy-class Secondary
ProCurve(config-policy-class)# allow list MatchAllSecondary
ProCurve(config-policy-class)# exit
ProCurve(config)# interface ethernet 0/2
ProCurve(config-eth 0/2)# access-policy Primary
ProCurve(config-eth 0/2)# interface demand 1
ProCurve(config-demand 1)# access-policy Secondary
ProCurve(config-demand 1)# exit
ProCurve(config)# no ip policy-class Primary rpf-check
ProCurve(config)# no ip policy-class Secondary rpf-check
ProCurve(config)# ip policy-class NATInside
ProCurve(config-policy-class)# nat source list MatchAllPrimary interface
ethernet 0/2 overload policy Primary
ProCurve(config-policy-class)# nat source list MatchAllSecondary interface
demand 1 overload policy Secondary
ProCurve(config-policy-class)# exit
ProCurve(config)# interface ethernet 0/1
ProCurve(config-eth 0/1)# access-policy NATInside
ProCurve(config-eth 0/1)# exit
```

7. Create the floating static route.

```
ProCurve(config)# ip route 0.0.0.0 /0 demand 1 20
```

Viewing Network Monitor Tracks and Probes

Network monitoring automates the process of testing network and server performance. However, nothing can relieve you of the necessity of viewing the results of these tests and taking appropriate action.

When you enable a track to log changes in probe status, the track automatically displays those logs to the terminal screen, as well as saves them to the event history. Enter this enable mode command to view the event history:

```
ProCurve# show events
```

If you want to view the logs on a remote server, enter these commands from the global configuration mode context:

Syntax: logging forwarding on
Syntax: logging forwarding receiver-ip <A.B.C.D>

For more information about forwarding logs, see *Chapter 4: ProCurve Secure Router OS Firewall—Protecting the Internal, Trusted Network*.

You can also periodically check on the status of a track and associated probes as you manage your ProCurve Secure Router using the commands described in the sections below.

Viewing Network Monitor Tracks

Enter this enable mode command to check the status and configuration of all tracks configured on your router:

Syntax: show track

The CLI displays the following information about each track:

- Current State—Pass or Fail
- Test Value—three pieces of information:
  - the probe or probes associated with the track
  - whether one or both probes must pass (OR or AND)
  - the status of each probe
- Track Changes—the number of times that the track's state has changed.
- Time in current state—listed in days, hours, minutes, and seconds
- Tracking—the routes that the track monitors, if any

You can view a single track by adding its name to the command:

Syntax: show track <*name*>

Viewing the track in real time shows moment-to-moment changes:

Syntax: show track <*name*> realtime

You can view the entire configuration for all tracks configured on the ProCurve Secure Router by entering this enable mode command:

Syntax: show running-config track [verbose]

Use the **verbose** option to see all commands, including default settings that you have not altered.

Debugging Network Monitor Tracks

Another way to monitor track activity in real time is to view debug messages. These messages give you more detailed information about the track's functions. Enter this enable mode command:

Syntax: debug track

Debug messages display when the track disables or enables a route. For example:

NETWORK.MON track SiteB Static Route 192.168.100.0 255.255.255.0 ppp 1 disabled

Viewing Network Monitor Probes

You might see from a track that a probe's state has changed. View the probe itself for more information, entering this command from the enable mode context:

Syntax: show probe [<*name*> {realtime}]

Use the **realtime** keyword to view the results of each test as they occur. Specify a name to see one probe only. If you do not specify a name, the CLI displays all configured probes (but you cannot view multiple probes in real time).

The **show probe** command displays the following information about a probe's status and configuration:

- Current State—Pass or Fail
- Admin. Status—Up (active) or Down (inactive or shut down)

  The state is automatically Pass if the status if Down.
- Type—ICMP echo, TCP connect, or HTTP request
- Period—the time (in seconds) between tests

- Timeout—the time (in milliseconds) that a test has in which to pass
- Hostname or IP address—the probe's destination
- Tracked by—the tracks that use the probe to monitor the network
- Tests run—the total number of tests run since statistics were last cleared
- Failed—the total number of failures since statistics were last cleared
- Time in current state—listed in days, hours, minutes, and seconds

As for tracks, you can view the entire configuration of all probes configured on the ProCurve Secure Router:

Syntax: show running-config probe [verbose]

Debugging Network Monitor Probes

Debugging probes gives you a moment-to-moment view of the results of tests. You can see in real time when a test passes or fails, even if the test does not alter the probe's state.

Enter this command from the enable mode context:

Syntax: debug probe

You can also specify a specific probe:

Syntax: debug probe <*name*>

Messages display the results of tests, but not the contents of actual packets. To view packets, enter the **debug** command for the protocol used by that probe: **debug ip icmp** or **debug ip tcp**.

Clearing Statistics

When viewing network monitoring probes and tracks, you often want to view current conditions. Table 9-6 shows the commands used to clear network monitoring counters. When you clear a probe, the probe's statistics clear, but not the count that the probe has been keeping of the number of tests that have failed.

Table 9-6. Network Monitoring clear Commands

Command	Clears
clear counters probe	for all probes: • number of tests run • time in current state
clear counters probe <name>	for the specified probe: • number of tests run • time in current state
clear counters track	for all tracks: • number of times that the track state has changed • time in current state
clear counters track <name>	for the specified track: • number of times that the track state has changed • time in current state

Troubleshooting Network Monitoring

This section explains common causes of general problems with network monitoring:

- The track does *not* take action (remove a route or log a change) when it should.
- The track takes action (removes a route or logs a change) when it should *not*.
- A primary route is removed, but a backup route is not added.
- A primary route fails and is removed from the routing table, but then reappears periodically.

Track Fails to Take Action

You receive complaints that a remote server is down, but fail to see any logs in the event history about this problem. You ping a network, and the ping fails, but your track fails to remove the route. In either case, your track is not taking the action that you expect.

Reasons for this problem include:

- Track not associated with a route—Check which routes the track is monitoring with the **show track *<name>*** command. If the track is not monitoring any routes, it cannot remove a failed route. You must enter the **track** option with one of these commands:
  - **ip route**
  - **ip address dhcp**
  - **ip address negotiated**

  One reason that the **track** option might fail to appear is that you attempted to add it to a route that already exists, but did not enter exactly the same route.
- Track not enabled to log changes—View tracks' configuration (**show running-config track**) and check for the **log-changes** command. If you want to view logs on a remote server, verify that log forwarding is enabled.
- Probes have not been activated—By default, probes are shut down and fixed in the Pass state. Unless you activate the probes associated with the track, the track never changes state and takes action no matter what the network conditions. To activate a probe, move to the network probe configuration mode context and enter **no shutdown**.
- Multiple probes specified with the wrong option—When you want a track to fail when *either* probe fails, remember to use the **and** option when specifying the probes. The track then passes only when both the first *and* second probe pass.
- Track's dampening interval set inappropriately long—Check the dampening interval (in seconds) by entering **show track *<name>***.
- Probe's period or timeout set inappropriately long—Check these settings by entering **show probe *<name>***.

Track Takes an Inappropriate Action

A track that takes action when it should not can be as dangerous as one that fails to take action, particularly when you have tied the track to a route. Even when a track simply logs changes, you want these logs to be accurate.

Check for these misconfigurations:

- Track shutdown—An inactive track automatically fails and, if so configured, removes the associated route. Therefore, if you want to remove a track from the router, delete it entirely (**no track <name>**) instead of shutting it down. If you want to disable the track temporarily, disable the track's *probes*, but not the track itself.

- Incorrect probe destination—View a probe (**show probe <*name*>**) and check for a miskeyed destination address or hostname, which would cause the probe to fail when it should pass.
- Inappropriate timeout and tolerance settings—You might find that you need to adjust a probe's timeout or tolerance settings to account for normal network congestion. If you find that a probe often fluctuates between Pass and Fail, you can increase the tolerance, add a dampening interval to the associated track, or both.
- Misconfigured HTTP probe—You can customize the response required from a Web server, but with this increased control comes increased risk for misconfiguration. These two commands can cause an HTTP probe to fail even when the probe receives a response for the server:
  - **expect status <*value*>**
  - **expect regex <*expression*>**

  View the configuration for the probe associated with the track (**show running-config probe**) and look for those commands. Look for and fix obvious misconfigurations such as typos. If you cannot find a misconfiguration, try expanding the range of allowed statuses or simply removing the commands in question entirely. Then check whether probes more accurately test the Web server's status.

Backup Route Fails to Be Added

Network monitoring tracks can remove failed routes, but they cannot add backup routes. You must configure those routes in advance. Simply create a route to the network in question through the backup interface (typically a demand interface), and assign the route a higher administrative distance than the primary route.

Make sure that a demand interface is ready to initiate a call. It should be spoofing an up status. See the *Basic Management and Configuration Guide*, *Chapter 8: Configuring Demand Routing for Primary ISDN Modules* and the *Advanced Management and Configuration Guide*, *Chapter 3: Configuring Backup WAN Connections* for more information on configuring demand routing.

Failed Primary Route Periodically Reappears in the Routing Table

A primary connection fails; network monitoring detects the failure and removes the primary route. A backup connection becomes active, and traffic once again reaches its destination. A minute or two later, the router begins dropping traffic again. Then after several minutes, traffic can again reach its destination. Users complain that their connections keep going up and down.

The problem is that PBR has not been configured for probe traffic, so the probe begins testing the backup route rather than the primary route. The track, on the other hand, is still monitoring the primary route. Because the backup route is good, the track believes incorrectly that the primary route is valid and should be added to the routing table.

To fix this problem, make sure that you have correctly configured a PBR route map for the probe. See “Implementing PBR to Route Probe Traffic” on page 9-34. Check that:

- The route map has been applied to router traffic as the local policy.
- A route map entry matches the correct ACL configured for the probe.
- The destination address in the ACL matches the probe’s destination. For TCP and HTTP probes, the destination port must also match.

Quick Start

This section contains the commands that you must enter to quickly configure network monitoring to control static and DHCP routes.

Only minimal explanation is provided. If you need additional information about any of these options, check "Contents" on page 9-1 to locate the section that contains the explanation you need.

First, configure probes to test a remote server or a connection to a remote network:

1. Create the probe and select its type.

   Syntax: probe <*name*> [icmp-echo | tcp-connect | http-request]

2. Specify the probe's destination.
   - Specify an ICMP probe's destination:

     Syntax: destination <*hostname* | *A.B.C.D*>
   - Specify a TCP or HTTP probe's destination:

     Syntax: destination <*hostname* | *A.B.C.D*> port <*value*>

3. Specify the probe's tolerance.
   - Set the probe to fail after a certain number of consecutive failures:

     Syntax: tolerance consecutive-failures <*failures*>
   - Set the probe to fail after certain rate of failure (for example, 10 out of 15):

     Syntax: tolerance rate-of-failure <*failures*> <*set size*>

4. Specify the test's timeout or retain the default timeout (1.5 seconds for ICMP probes and 10 seconds for TCP and HTTP probes).

   Syntax: timeout <*milliseconds*>

5. Specify the probe's period or accept the default (60 seconds):

   Syntax: period <*seconds*>

6. Optionally, repeat these steps to create another probe.

Next configure a track:

7. Create the track and specify its name.

   Syntax: track <*name*>

8. Specify the probe for the track.
   - Specify one probe.

     Syntax: test probe <*name*>
   - Specify two probes, both of which must pass for the track to pass.

     Syntax: test probe <*name*> and probe <*name*>
   - Specify two probes, either of which can pass for the track to pass.

     Syntax: test probe <*name*> or probe <*name*>

9. Optionally, specify a dampening interval.

   Syntax: dampening-interval <*seconds*>

10. Configure the track's action.
    - You can configure the track to remove a static route when the track fails. This command is entered from the global configuration mode context:

      Syntax: ip route <*destination network A.B.C.D*> <*subnet mask | /prefix length*> <*next-hop A.B.C.D | forwarding interface ID*> [<*administrative distance*>] track <*name*>
    - You can configure the track to remove a DHCP default route when the track fails. This command is entered from an interface configuration mode context:

      Syntax: ip address dhcp {client-id [ethernet 0/<*port*> | HH:HH:HH:HH:HH:HH:HH] | hostname <*hostname*>} track <*name*> [<*administrative distance*>]

      Syntax: ip address dhcp [hostname <*hostname*> | no-domain-name | no-nameservers] track <*name*> [<*administrative distance*>]
    - You can configure the track to remove a negotiated default route when the track fails. This command is entered from an interface configuration mode context:

      Syntax: ip address negotiated track <*name*> [<*administrative distance*>]
    - You can enable the track to log changes in its state. This command is entered from the network monitor track configuration mode context:

      Syntax: log-changes

Next, configure PBR to specify the route for probe packets. The ProCurve Secure Router should, of course, always forward these packets along the route that the probe is testing:

11. Create an extended ACL to select traffic associated with each probe:

    Syntax: ip access-list extended <*listname*>

12. Permit the probe traffic:

    Syntax: permit tcp any [host <*destination A.B.C.D*> | hostname <*hostname*>] [eq <*destination port*>]

    Syntax: permit icmp any [host <*destination A.B.C.D*> | hostname <*hostname*>]

13. Create a route map entry.

    Syntax: route-map <*mapname*> <*index number*>

14. Match the route map entry to the ACL created for the probe.

    Syntax: match ip address <*listname*>

15. Set the forwarding interface or next-hop address to match the route that the probe tests. The route map forces probes along the specified route independent of the routing table.

    Syntax: set [interface <interface ID> | ip next-hop <*A.B.C.D*>]

16. Configure the route map to drop packets if it cannot force them along the route specified in step 15.

    Syntax: set interface null 0

17. Apply the route map to local router traffic. Exit to global configuration mode and enter:

    Syntax: ip local policy route-map <*mapname*>

Configure NAT, if you are using that feature. When you use NAT with network monitoring, you must create a NAT statement that translates source addresses to each possible forwarding interface. You must also create ACPs for the WAN interfaces, which help the router determine which NAT statement to apply.

18. Create an ACL to select local traffic to be translated. From the global configuration mode context, enter:

    Syntax: ip access-list standard <*listname*>
    Syntax: [permit | deny] [any | host {<*A.B.C.D*> | hostname <*hostname*>} | <*A.B.C.D*> <*wildcard bits*>]

19. Create an ACL to select incoming traffic permitted on the primary connection. From the global configuration mode context, enter:

    Syntax: ip access-list standard <*listname*>
    Syntax: [permit | deny] [any | host {<*A.B.C.D*> | hostname <*hostname*>} | <*A.B.C.D*> <*wildcard bits*>]
    Syntax: ip access-list extended <*listname*>
    Syntax: [permit | deny] <*protocol*> <*source address*> <*source port*> <*destination address*> <*destination port*> [<*packet bits*>] [log | log-input]

20. Create another ACL to select incoming traffic permitted on the backup connection.

21. Create an ACP to allow the primary connection ACL. Starting from the global configuration mode context, enter:

    Syntax: ip policy-class <*policyname*>
    Syntax: allow list <*listname*>

22. Create a second ACP to allow the secondary connection ACL.

23. Apply the ACPs to the corresponding interfaces. Starting from the global configuration mode context, enter these commands:

    Syntax: interface <*primary interface ID*>
    Syntax: access-policy <*primary policyname*>
    Syntax: interface <*backup interface ID*>
    Syntax: access-policy <*backup policyname*>

24. Create an ACP for NAT.

    Syntax: ip policy-class <*policyname*>

25. Create a NAT statement:
    - Specify the ACL that you configured for local traffic.
    - Specify the primary WAN interface or an IP address valid for the primary connection.
    - Specify the ACP for traffic on the primary interface.

    Syntax: nat source list <*listname*> [address <*A.B.C.D*> | interface <*interface*>] overload policy <*policyname*>

26. Create a second NAT statement:
    - Specify the ACL that you configured for local traffic.
    - Specify the backup WAN interface or an IP address valid for the backup connection.
    - Specify the ACP for traffic on the primary interface.

    Syntax: nat source list <*listname*> [address <*A.B.C.D*> | interface <*interface*>] overload policy <*policyname*>

27. Disable the RPF check on the ACPs applied to the WAN interfaces. Enter this global configuration mode command:

    Syntax: no ip policy-class <*policyname*> rpf-check

28. Make sure that the firewall is enabled. From the global configuration mode context, enter:

    Syntax: ip firewall

Finally, activate the probes:

29. Access a probe's configuration mode context.

    Syntax: probe <*name*>

30. Activate the probe.

    Syntax: no shutdown

31. Repeat to activate other probes.