

Configuration Guide

Network Monitoring in AOS

This configuration guide describes network monitoring and its use on ADTRAN Operating System (AOS) products. This guide contains information about the AOS Network Monitor feature, including a feature overview, component configuration, common applications and network examples, a command line interface (CLI) command summary, and simple troubleshooting procedures.

This guide consists of the following sections:

- *Network Monitor Overview on page 2*
- *Network Monitoring in the Network on page 5*
- *Hardware and Software Requirements and Limitations on page 6*
- *Configuring Network Monitor on page 6*
- *Using the Network Monitor Wizard on page 43*
- *Network Monitor Common Configurations and Applications on page 48*
- *Network Monitor Command Summary on page 63*
- *Troubleshooting on page 71*
- *Additional Documentation on page 77*

Network Monitor Overview

Network Monitor is a feature composed of multiple parts designed to test and control connectivity routes within a network structure. The primary function of network monitoring is to detect and remove failed routes so that backup routes can take effect, and then restores the failed routes when they are functioning properly again. Network monitoring can also be used to adjust router configurations and permissions as needed. Overall, network monitoring enhances ADTRAN products' capabilities to detect and adapt to changes in the network by monitoring remote sites, detecting link and wide area network (WAN) failures, and monitoring virtual private network (VPN) connections.

Component Parts

The Network Monitor feature is composed of three main mechanisms: probes, tracks, and schedules. Each component works together to ensure proper route connectivity in an automated manner to provide easy network management for network administrators. Each component is described in the following sections, along with additional considerations for configuring network monitoring.

Probes

Probes are objects in the unit's configuration that collect information about network connectivity by sending test traffic across network paths. The probes used by Network Monitor are either in a PASS or a FAIL state at any given time. PASS states indicate that a probe is successfully receiving responses to the test packets sent to the designated destination by the probe, and FAIL states indicate the test traffic is not reaching its destination or that the information was not transmitted successfully. Each probe is defined by a number of parameters. These parameters include:

- Probe Destination: The IP address or host name where the probe test packets are sent (the address to be monitored).
- Probe Period: The time (in seconds) between probe test attempts.
- Probe Source: The source IP address for the probe packets.
- Probe Timeout: The time it takes for a test to be considered as failed.
- Probe Tolerance: The number of tests that must pass or fail before the probe changes states.

Network Monitor supports four types of probes: Internet Control Message Protocol (ICMP) echo (Ping) probes, Hypertext Transfer Protocol (HTTP) request probes, Transmission Control Protocol (TCP) connect probes, and Two-way Active Measurement Protocol (TWAMP) probes.



Although Network Monitor supports TWAMP probes, their purpose is to measure roundtrip IP performance between devices, and their use falls more accurately under Network Quality Monitoring. These probes are mentioned, but not covered, in this document. For more information about TWAMP probes, refer to [Configuring Network Quality Monitoring \(NQM\) in AOS](https://supportforums.adtran.com) available online at <https://supportforums.adtran.com>.

ICMP echo probes are used to test network connectivity by sending packets which either do or do not reach their destination and either do or do not receive a response. Packet size and packet data patterns can be specified for each ICMP echo probe, in addition to configurable parameters common to all probes. Most probes used in network monitoring for testing link connectivity are ICMP echo probes.

HTTP request probes are useful when using Network Monitor to gauge the health of remote sites and applications. HTTP request probes send HTTP requests to Web servers, and determine success or failure of the probe based on the information returned by the server. HTTP request probes can be GET, HEAD, or RAW requests. GET requests are simple Web page requests, HEAD requests only request Web page header information, and RAW requests are custom-configurable HTTP requests. Each HTTP request probe can include command strings, expected statuses or strings in response packets, the server's root path, and the packet's source port.

TCP connect probes are used to test network connectivity by sending packets which either do or do not open a TCP session. The probe's destination TCP port must be specified, and source ports and source addresses for TCP probes can be specified. TCP connect probes are useful in situations when the destination does not listen to ICMP echo probes.

Tracks

Tracks are objects in the unit's configuration created to monitor probes for state changes, and cause other objects to take action based on the probe's state. A probe can monitor network conditions, but cannot take any action on its own. The track is tied to the probe through configuration, and specifies that an object (interface, schedule, etc.) will perform an action based on the probe's (track's) state. Tracks have PASS or FAIL states just as probes do, and use these state changes to cause objects to perform specific actions. Specific actions tracks can make objects perform include removing faulty routes, logging events, executing a tool command language (Tcl) script, controlling individual statements inside access control lists (ACLs), or controlling the availability of crypto maps. These actions are described in more detail in [Track Configuration Using the CLI on page 23](#) and in the configuration examples beginning on [page 48](#).

It is important to understand that although the track monitors the probe's state, and causes an object to perform an action based on the link between the track and the probe, it is not the track's configuration that causes the object to take the action. The action is specified in the object's configuration by specifying the action to be taken when the track changes state. [Table 1](#) gives a few examples of specifying an action in an object's configuration. These examples of using a track in an object's configuration are not all-inclusive, but they should help you to understand that the action actually occurs at the object's configuration level.

Table 1. Tracks and Object Actions

Configuration Command	Command Indication	Further Reading
<code>#run-tcl <filename> track <name></code>	Specifies that a Tcl script is initiated when a track changes states.	Configuring Tcl Scripting in AOS
<code>(config)#ip access-list extended TEST1 (config-ext-nacl)# <action> <protocol> <source> <source port> <destination> <destination port> track <name></code>	Specifies that an entry in an ACL is dependent upon a track, and is either applied or not applied based on the state of the track.	AOS Command Reference Guide

Table 1. Tracks and Object Actions (Continued)

Configuration Command	Command Indication	Further Reading
(config)# ip route <ip address> <subnet mask> <interface> <administrative distance> track <name>	Specifies that the route is dependent upon the track to be functional. If the track enters a FAIL state, the route specified in this command is disabled, and traffic is no longer routed using this route.	<i>AOS Command Reference Guide</i>
(config)# interface <interface> (config-<interface>)# ip address dhcp client-id [<interface> <identifier>] [hostname <"string">] [track <name>] [<administrative distance>]	Specifies that on this interface, the Dynamic Host Configuration Protocol (DHCP) gateway route for this client only resides in the route table while the track is in the PASS state.	<i>AOS Command Reference Guide</i>
(config)# mail-client myagent (config-mail-client-myagent)# capture trigger track <name> [fail pass]	Specifies that the trigger used by the generic mail agent to know when to capture command output is based on whether a track has changed from PASS to FAIL, or from FAIL to PASS.	<i>Generic Mail Agent Quick Configuration Guide</i>
(config)# interface switch <slot/port> (config-sw<slot/port>)# no shutdown track <name>	Specifies the switch interface is up or down depending on the track state. If the track is in a FAIL state, the interface is down. If the track is in a PASS state, the interface is restored.	<i>Port Scheduler</i>

Tracks can be associated with either one or more probes, and are configured to cause specific actions based on probe state changes. When using multiple probes with a single track, tracks can be configured to PASS only when all probes are in a PASS state, or to PASS when any of the probes is in a PASS state.

Schedules

Schedules are objects in the unit's configuration that monitor the time of day and day of the week. Schedules are active or inactive based on their configuration and the time of day. Schedules are used to determine what times during the day and how often tracks and probes are active. They are beneficial in creating reoccurring network monitoring tests. There are three types of schedules used by Network Monitor: absolute, relative, and periodic.

Absolute schedules are active only once, for a specified amount of time. This type of schedule activates at an absolute month, day, year, and time.

Relative schedules become active after a specified delay. The delay begins when the delay command is entered into the unit's configuration. Relative schedules are beneficial in situations such as those where the activation of dial backup interfaces should be delayed until after the primary interface has had time to start up.

Periodic schedules transition from active to inactive at specified periods. These periods can be daily, weekly, on weekdays, or on weekends. Periodic schedules are beneficial in creating reoccurring network monitoring tests.

Special Considerations

Policy based routing (PBR) is often necessary when implementing network monitoring. PBR can route the probe traffic to ensure that the proper connections are being monitored. Without using PBR, probes may egress through a backup connection, rather than through the primary interface. If this occurs, the correct connection is not being monitored, and the track actions based on the probe will be incorrect. This scenario presents a false positive, and reports that the connection is active, when it is the wrong connection being monitored. This situation also can lead to a connection that toggles up and down, causing users to lose their sessions, as tracks reinstate primary routes that are not really connected based on the states of the associated probes which have reached their destination through a backup connection. Using PBR forces probe traffic through the primary interface, and through the correct connection, to ensure that the track is responding to feedback about the correct connection.

This guide does not go into detail about all PBR configurations, but instead focuses on PBR configuration as it relates to Network Monitor. For detailed PBR configuration information, refer to the *Policy Based Routing Configuration Guide* available online at <https://supportforums.adtran.com>.

Tracks can also be associated with other network configurations, such as routes, ACLs, crypto maps, and Tcl scripts. These associations are used to determine how network monitoring behaves on your network.

Network Monitoring in the Network

Network monitoring has many benefits in your network. It can monitor the operational status of static routes, reroute traffic when static routes fail, and even monitor default routes received with a Dynamic Host Configuration Protocol (DHCP) address or a negotiated IP address. Often, the WAN is connected to the Internet through an Ethernet connection and a cable or broadband modem. Without network monitoring, the connection will appear valid even if the modem is down. The connection appears valid because there is no problem with the Ethernet connection. With network monitoring, however, the connection problem at the modem is noticed, and the route through the modem can be removed from the route table and replaced with another backup route so the entire WAN maintains connectivity to the outside world. *Figure 1* and *Figure 2* describe a typical network in which network monitoring is used to change the route in the route table and activate a backup route through the Internet.

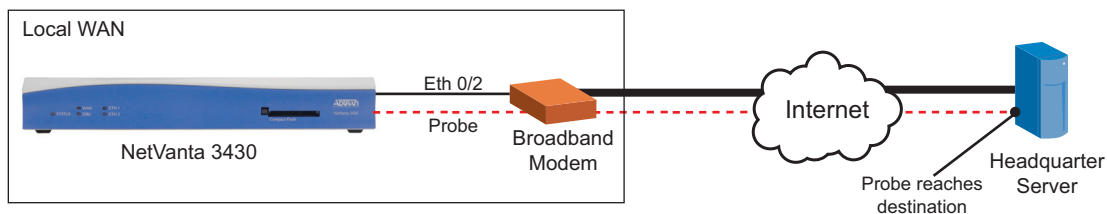


Figure 1. Primary Network Connection

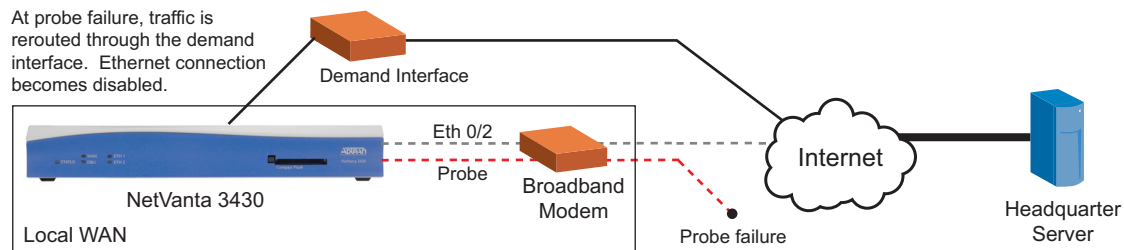


Figure 2. Backup Network Connection after Primary Connection Failure

Hardware and Software Requirements and Limitations

The Network Monitor feature is available on AOS products as outlined in the *Product Feature Matrix* available online at <https://supportforums.adtran.com>.

In AOS firmware release R10.7.0, support for Internet Protocol version 6 (IPv6) was added to the Network Monitor feature. For more information about configuring IPv6, refer to the guide *Using IPv6 in AOS* available online at <https://supportforums.adtran.com>.

In AOS firmware release R10.11.0, support for Y.1731 maintenance endpoints (MEPs) and Ethernet in the First Mile (EFM) bonding groups was added to the Network Monitor feature. For more information about configuring Y.1731, refer to the guide *Configuring Ethernet OAM using Y.1731*, available online at <https://supportforums.adtran.com>.

In AOS firmware release R11.2.0, Network Monitor became virtual routing and forwarding (VRF) aware. Probes that support nondefault VRF functionality are HTTP Request, ICMP Echo, ICMP Timestamp, TWAMP, and UDP Echo responders. Responders that support VRF functionality are ICMP Timestamp, TWAMP, and UDP Echo responders. When a nondefault VRF is specified, the probe or responder service is limited to the specified VRF. If no VRF is specified, the service is limited to the default (unnamed) VRF. When a nondefault VRF is specified for probe responders, multiple source interfaces can be specified, resulting in one source interface for each VRF. For more information about VRF configuration, and its use in AOS products, refer to the guide *Configuring Multi-VRF in AOS*, available online at <https://supportforums.adtran.com>.

In AOS firmware release R11.10.0, support for RapidRoute service assurance was added to the Network Monitor feature. For more information about RapidRoute service assurance, refer to the guide *RapidRoute Service Features in AOS*, available online at <https://supportforums.adtran.com>.

There is a limit of **10** probes that can be configured on the AOS product.

Configuring Network Monitor

To configure Network Monitor, you must:

- Configure one (or more) probes.
- Configure a track and associate it with a probe.
- Specify the track's action.
- Configure a schedule (optional).

- Associate the schedule with a track (optional).
- Configure PBR for probe traffic.

Configuring Probes

To create a Network Monitor probe, you must define the following probe characteristics before activating the probe:

- Name
- Associated VRF (optional)
- Type
- Destination
- Source

You should also be aware of the following probe characteristics, which generally operate with the default value:

- Period
- Timeout
- Tolerance

Additional probe configuration parameters are also available depending on the probe type.

Probe Name

The probe's name is the identifier of the probe. You should name the probe something that makes it easy to identify in your network, such as **INTERCONNECTIVITY**, **BACKUPCONNECTION**, or **REMOTEOFFICE**.

Probe VRF

The probe's VRF is the VRF on which the probe operates. If no VRF is specified, the probe operates on the default (unnamed) VRF. Specifying a VRF for the probe is optional. Refer to *Hardware and Software Requirements and Limitations on page 6* for more information about VRF functionality in the Network Monitor feature.

Probe Type

The probe's type must be defined as either an ICMP echo probe, a TCP connectivity probe, or an HTTP request probe. Each type of probe is better for use in specific situations, and each probe type has configurable parameters not required by all probes. ICMP echo probes are useful in testing network connectivity, and can include packet size and packet data configurations. TCP connect probes are useful in testing network connectivity when the destination does not listen to ICMP echo probes, and can include source port, source address, and destination TCP port configurations. HTTP request probes are useful in monitoring a Web server, and can include command string, response packet, path, and port configurations. The configuration parameters for each type of probe are described in detail in the following sections.

Probe Destination

The probe's destination is the location to which the probe sends test packets. This destination can be defined by a host name or an IP address.



*If a host name is used, a domain naming system (DNS) server should be learned by the AOS device via DHCP or specified in the global configuration with the **ip name-server** command.*

TCP connectivity probes require the destination port be specified along with the host name or IP address. HTTP request probes use port **80** by default, but a different port can be specified if necessary.

Probe Source

The source IP address for probe packets is by default the address of the interface through which the probe packets are transmitted. The address can be specified manually, but because the probe requires a valid address on the unit's local interface to function correctly, manually entering the address may lead to probe failure if the local interface address is changed at another time. This can be problematic in situations where dynamic addresses are given out by an Internet service provider (ISP). Since PBR is used with Network Monitor to force the probes out of a desired interface, there is usually no need to specify the source IP address unless the probe traffic is intended to traverse a VPN tunnel.

TCP connect and HTTP request probes can also specify the source port for sending probe packets. By default, the unit begins sending probes from port **1026**, and moves up one port with each probe packet. Changing the port to a set port (range is **1** to **65535**) allows the probe to successfully pass through a firewall which can filter out certain ports, or keep one probe's traffic separate from another probe's, even when they have the same destination.

Probe Period

The probe's period is the elapsed time (in seconds) between the sending of probe packets. The probe's period must always be greater than the probe's timeout, otherwise the probe will return a false result. The shorter the probe's period, the more detailed the image of network performance and the faster you can detect a downed connection, but shorter periods also add additional traffic to WAN connections. The default probe period is **60** seconds for all probes. To change the period of an ICMP echo probe, enter a value between **1** and **65535** seconds, and for TCP connect or HTTP request probes, enter a value between **60** and **65535** seconds.

Probe Timeout

The probe's timeout value is the determined time (in milliseconds) in which a returned packet must be received before the test is considered to have failed. The timeout value ranges from **1** to **900000** milliseconds. Each probe type has a different default timeout value: ICMP echo probes have a default timeout value of **1500** milliseconds, TCP connect probes have a default timeout value of **10000** milliseconds, and HTTP request probes have a default timeout value of **10000** milliseconds.

Probe Tolerance

Tolerance refers to the number of probe tests that pass or fail before the probe changes states, either from PASS to FAIL or from FAIL to PASS.



There is no default setting for the probe tolerance value. If the tolerance is not specified, the probe will remain in a PASS state.

The tolerance can be specified as either a number of consecutive failures, or a rate of failures. If the tolerance is specified as a number of consecutive failures, then a specified number of consecutive tests must fail (or pass) for the probe to change states. If setting the tolerance consecutively for failures, each time a test passes, the probe resets the count. Conversely, if the tolerance is set consecutively for passes, each time a test fails, the probe resets the count. The valid range of consecutive failures or passes is **1** to **255**. The consecutive failure option is beneficial in cases where connectivity is being tested.



*When using ICMP echo probes to test for connectivity, the consecutive tolerance should be set to at least **3** to compensate for routinely lost packets.*

If the tolerance is specified as a rate of failure, then the probe will change states after tests fail or pass at a certain rate. By default, the valid range for failures in a rate of failure configuration is **1** to **254**. When specifying a rate of failure, a test set size must be defined. The test set size is the number of tests to be considered for determining a rate of failure (for example, a probe can be configured to fail when **20** out of **25** tests fail, or be configured to pass when **8** out of **10** tests pass. In those examples, **25** and **10** are the test set sizes). The range of test set sizes is **1** to **255**. Using the rate of failure tolerance setting is beneficial when monitoring the performance of a server.



*The probe's **tolerance**, **period**, and **timeout** values determine the minimum time required by the probe to detect a failure. The following formulas are beneficial in estimating the length of time a network experiences unacceptable conditions before the probe reports a failure:*

For a probe with consecutive failure tolerance settings:

*Tolerance (# of consecutive failures) * (Period + Timeout (in seconds))*

For a probe with rate of failure tolerance settings:

*Tolerance (minimum allowed failures) * (Period + Timeout (in seconds))*

*Tolerance (maximum set size) * (Period + Timeout (in seconds))*

Determining the length of time before a probe reports a failure is beneficial for probes monitoring a route and additional probe traffic on the network.

Probe Activation

By default, the probe is shut down until it is activated, and must be activated manually to begin running tests. The probe can be activated from either the Web-based graphical user interface (GUI) or CLI, and remains in a PASS state until it is activated. Once the probe is activated, it will begin running tests and consuming bandwidth, but will not have any effect until it is associated with a track. Refer to [Configuring Tracks on page 21](#) for more information on associating tracks and probes.

Additional Probe Configuration Parameters

Some probe types have special considerations and optional parameters that can be configured. This section describes the additional parameters associated with ICMP echo and HTTP request probe types.

Additional ICMP Echo Probe Parameters

In addition to the configurable parameters common to all probes, ICMP echo probe packets can be configured by size and data pattern. The default data size for an ICMP echo packet is **0** bytes, but the data size of the packet can be changed to any size between **0** and **1462** bytes. Changing the packet size, which is at the minimum by default, increases the bandwidth required by the probe traffic. Sometimes changing the packet data size aids in testing for fragmentation, especially on Voice over IP (VoIP) frames.



*If the ICMP echo probe packet data size is changed for fragmentation testing, the router must be notified to mark the probe packets with a **don't fragment bit** in the IP header. Refer to the [Policy Based Routing Configuration Guide](#) available online at <https://supportforums.adtran.com>.*

The data pattern of the ICMP echo packet can also be changed if necessary. Normally, the ICMP echo probes are just testing for connectivity, so it is not important what is communicated in the test packets. If that is the case, the data pattern can be left at the default. The default pattern begins at 0x00 and increases along the length of the packet. For example, a packet of 64 bytes would have the following pattern: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11...3a 3b 3c 3d 3e 3f (with 3f as the 64th byte). The data pattern can be changed, however, to any hexadecimal characters, for example, **0FF0**. The new pattern is repeated as many times as necessary to fill out the packet's data bits. Changing the data pattern can be beneficial if a network interface is suspected of malfunctioning when it transmits or receives a certain data string (usually long continuous strings of 1s or 0s). By changing the packet data pattern to a suspect data pattern, it is possible to locate the problem.

HTTP Request Probe Types

You can define the request type of HTTP request probes. Typically, most HTTP request probes are HTTP GET requests. However, HEAD and RAW requests can also be used to demand specific responses from servers. HTTP GET requests are the standard requests sent to Web servers, and are the default request type for HTTP request probes. This request type is good for testing the actual performance of a server, but because the server sends all the data of the Web page in the response packet, this request type can add to network traffic.

HTTP HEAD requests require the same response as a GET request, but only require that the heading of the response packet be sent, thus effectively checking for performance without tying up the network. HEAD requests can also be used to monitor the status of the server response, since this status is included in the header.

HTTP RAW requests are useful if it is necessary to customize the request. RAW requests are generally not necessary for monitoring routes, but they can be beneficial in accessing specific information stored on a server. For example, if a large number of remote units are in your network, they can be configured to periodically send HTTP RAW requests to a centralized HTTP server. A common gateway interface (CGI) script can receive these requests and store them in a file or database, where they can be used by a Web page to display the status of all units in the network. The following are system variables that can be used in the RAW string:

Table 2. String Variables for HTTP RAW Requests

String Variable	Definition
\$SYSTEM_NAME	The host name of the system or unit.
\$SYSTEM_SERIAL_NUMBER	The serial number of the system or unit.
\$SYSTEM_DESCRIPTION	The product name and part number of the system or unit.
\$SYSTEM_SOFTWARE_VERSION	The firmware version of the system or unit.

When configuring HTTP RAW requests through the CLI, there are certain conventions that are followed. These conventions are detailed in *HTTP Request Probe Configuration on page 15*.

Additional HTTP Request Probe Parameters

HTTP request probes can allow you to mandate that Web servers return a particular status (Ok, Forbidden, Not Found, etc.) or particular information in their response packets. Specifying that Web servers return a particular status can aid in determining that the server is both connected and successfully returning correct information. Sometimes problems arise, where servers are not returning the requested information, and monitoring status codes returned by HTTP requests can help to determine the problem. HTTP status codes follow those outlined in RFC 2616, and follow this general format:

Table 3. HTTP Status Codes (RFC 2616)

Status Code Range	Indication
1xx	Informational status: indicates request is received and the server is continuing to process the request.
2xx	Success status: indicates the request was received successfully.
3xx	Redirection status: indicates additional action is required to complete the request.
4xx	Client Error status: indicates the request cannot be fulfilled or contains bad syntax.
5xx	Server Error status: indicates the server failed to fulfill a valid request.

To specify a status code to indicate a successful HTTP request probe, the minimum, or minimum and maximum, status values must be defined. For example, you could specify that only successful requests be accepted by specifying the minimum value as **200**, or by specifying the minimum value as **200** and the maximum value as **299**.

HTTP request probes can also be configured to accept only specific information from the server to be considered as successful. This configuration specifies that not only is a server sending content, but that it is sending the correct content. To specify the desired information to be returned, you enter a set of keywords, or a regular expression. Regular expressions are the text or information you want the server to return in its response message to indicate it is connecting properly.

HTTP request probes can also be configured to follow a specific path to the server. This is beneficial if the server does not follow the default path (/), but instead uses a different path (/home/index.htm), and it keeps the probe from failing due to a faulty request.

Probe Configuration Using the CLI

The following sections detail the CLI configuration of all three types of probes (ICMP echo probe, TCP connect probe, and HTTP request probe).

ICMP Echo Probe Configuration

To configure an ICMP echo probe, follow these steps:

1. Specify the name for the probe and the probe type using the **probe** *<name>* [**http-request** | **icmp-echo** | **tcp-connect**] command from the Global Configuration mode prompt. Enter the command as follows:

```
(config)#probe PRIMARYCONNECTION icmp-echo  
(config-probe-PRIMARYCONNECTION)#
```

In the previous example, the probe's name is **PRIMARYCONNECTION**, the probe type is **icmp-echo**. The probe's configuration mode has been entered, indicated by the **config-probe-PRIMARYCONNECTION** prompt.

2. Optionally specify the VRF associated with the probe using the **vrf** *<name>* command from the probe's configuration mode. If no VRF is specified, the probe operates on the default (unnamed) VRF. Enter the command as follows:

```
(config-probe-PRIMARYCONNECTION)#vrf primary  
(config-probe-PRIMARYCONNECTION)#
```

In the previous example, the probe **PRIMARYCONNECTION** is configured to operate in the previously configured VRF **primary**.

3. Specify the probe's destination using the **destination** *<ip address | hostname>* command from the probe's configuration mode. IP addresses should be expressed in dotted decimal notation (for example, **10.10.10.1**) and host names should be fully qualified (for example, **www.company.com**). Enter the command as follows:

```
(config-probe-PRIMARYCONNECTION)#destination 10.10.10.1  
(config-probe-PRIMARYCONNECTION)#
```

- Specify the probe's period using the **period** *<value>* command from the probe's configuration mode. The period specifies the time (in seconds) between the sending of probe packets. Valid range is **1** to **65535** seconds, with a default value of **60** seconds. For most ICMP echo probes, the period will need to be reduced to much less than 60 seconds. Enter the command as follows:

```
(config-probe-PRIMARYCONNECTION)#period 10
(config-probe-PRIMARYCONNECTION)#
```

- Optional. Specify the probe's source address by using the **source-address** *<ip address>* command from the probe's configuration mode. IP addresses should be expressed in dotted decimal notation. By default, the probe uses the address assigned to the interface through which probe packets are transmitted. You should pay careful attention to the address you enter for the probe if you change it manually, because the probe requires a valid address on the unit's local interface to function correctly. Enter the command as follows:

```
(config-probe-PRIMARYCONNECTION)#source-address 10.10.10.2
(config-probe-PRIMARYCONNECTION)#
```

- Specify the probe's timeout value by using the **timeout** *<value>* command from the probe's configuration mode. The timeout value is the determined time (in milliseconds) in which a returned packet must be received before a test is considered failed. The timeout value ranges from **1** to **900000** milliseconds, with an ICMP echo probe default timeout value of **1500** milliseconds. Enter the command as follows:

```
(config-probe-PRIMARYCONNECTION)#timeout 1200
(config-probe-PRIMARYCONNECTION)#
```

- Specify the probe's tolerance by using the **tolerance consecutive** [**pass** *<number>* | **fail** *<number>*], **tolerance rate** [**fail** *<number>* | **pass** *<number>*] **of** *<set size>*, or **tolerance rate fail** *<number>* **pass** *<number>* **of** *<set size>* commands. The **consecutive** keyword indicates that the probe must either pass or fail a certain number of times in a row to change states. Valid range for consecutive passes is **1** to **255**; valid range for consecutive failures is **1** to **255**. In **consecutive** mode, the default value is **1**. The **rate** keyword indicates that the probe must either fail or pass a certain number of times in a specified number of tests before changing states, or that it must fail a certain number in a specified number of tests to change to FAIL state, and pass a certain number in a specified number of tests to change to a PASS state. Valid ranges for rate passes are **1** to **254** per set; valid ranges for rate failures are **1** to **254** per set. Valid set size range is **1** to **255**. Enter the command from the probe's configuration mode as follows:

```
(config-probe-PRIMARYCONNECTION)#tolerance rate fail 10 pass 5 of 15
(config-probe-PRIMARYCONNECTION)#
```

The proceeding example specifies that the probe **PRIMARYCONNECTION** will change to a FAIL state if **10** of **15** tests in a row fail, and that it will change to a PASS state if **5** of **15** tests in a row pass.

- Optional. Specify the probe packet's data length using the **size** *<payload length>* command. Changing the packet length can aid in testing for fragmentation, but larger sizes increase bandwidth usage. Valid range is **0** to **1462** bytes, and the default value is **0**. Enter the command from the probe's configuration mode as follows:

```
(config-probe-PRIMARYCONNECTION)#size 5
(config-probe-PRIMARYCONNECTION)#
```

- Optional. Specify the probe packet's data pattern using the **data** *<pattern>* command. The packet's data pattern can be used to monitor network malfunctions stemming from receipt or transmission of certain

data strings. The pattern can be specified as any hexadecimal data pattern. By default, the data pattern is a standard ping packet pattern of data values starting with 0x00, incrementing by one for the length of the packet. To change the packet's data pattern, enter the command at the probe's configuration mode as follows:

```
(config-probe-PRIMARYCONNECTION)#data 0FF0
(config-probe-PRIMARYCONNECTION)#
```

10. Activate the probe using the **no shutdown** command. Enter the command at the probe's configuration mode as follows:

```
(config-probe-PRIMARYCONNECTION)#no shutdown
(config-probe-PRIMARYCONNECTION)#
```

TCP Connect Probe Configuration

The steps for configuring a TCP connect probe are very similar to those for ICMP echo probes. For more detailed information about each step, refer to *ICMP Echo Probe Configuration on page 12*. To configure a TCP connect probe, follow these steps:

1. Specify the name for the probe and the probe type using the **probe <name> [http-request | icmp-echo | tcp-connect]** command from the Global Configuration mode prompt. Enter the command as follows:

```
(config)#probe SECONDARYCONNECTION tcp-connect
(config-probe-SECONDARYCONNECTION)#
```

2. Optionally specify the VRF associated with the probe using the **vrf <name>** command from the probe's configuration mode. If no VRF is specified, the probe operates on the default (unnamed) VRF. Enter the command as follows:

```
(config-probe-SECONDARYCONNECTION)#vrf secondary
(config-probe-SECONDARYCONNECTION)#
```

In the previous example, the probe **SECONDARYCONNECTION** is configured to operate in the previously configured VRF **secondary**.

3. Specify the probe's destination using the **destination <ip address | hostname> port <number>** command from the probe's configuration mode. IP addresses should be expressed in dotted decimal notation (for example, **10.10.10.1**) and host names should be fully qualified (for example, **www.company.com**). Valid port range is **1** to **65535**. Enter the command as follows:

```
(config-probe-SECONDARYCONNECTION)#destination www.company.com port 56
(config-probe-SECONDARYCONNECTION)#
```

4. Specify the probe's period using the **period <value>** command from the probe's configuration mode. The period specifies the time (in seconds) between the sending of probe packets. Valid range for TCP connect probes is **60** to **65535** seconds, with a default value of **60** seconds. Enter the command as follows:

```
(config-probe-SECONDARYCONNECTION)#period 70
(config-probe-SECONDARYCONNECTION)#
```

- Optional. Specify the probe's source address by using the **source-address** *<ip address>* command from the probe's configuration mode. IP addresses should be expressed in dotted decimal notation. Enter the command as follows:

```
(config-probe-SECONDARYCONNECTION)#source-address 10.10.10.3
(config-probe-SECONDARYCONNECTION)#
```

- Optional. Specify the probe's source port by using the **source-port** *<port>* command from the probe's configuration mode. Valid port range is **0** to **65535**. By default, the port is set to **0** so that the probe dynamically selects the port number. To specify the probe's source port, enter the command as follows:

```
(config-probe-SECONDARYCONNECTION)#source-port 1010
(config-probe-SECONDARYCONNECTION)#
```

- Specify the probe's timeout value by using the **timeout** *<value>* command from the probe's configuration mode. The timeout value ranges from **1** to **900000** milliseconds, with a TCP connect probe default timeout value of **10000** milliseconds. Enter the command as follows:

```
(config-probe-SECONDARYCONNECTION)#timeout 8000
(config-probe-SECONDARYCONNECTION)#
```

- Specify the probe's tolerance by using the **tolerance consecutive** [**pass** *<number>* | **fail** *<number>*], **tolerance rate** [**fail** *<number>* | **pass** *<number>*] **of** *<set size>*, or **tolerance rate fail** *<number>* **pass** *<number>* **of** *<set size>* commands. Valid range for consecutive passes is **1** to **255**; valid range for consecutive failures is **1** to **255**. Valid ranges for rate passes are **1** to **254** per set; valid ranges for rate failures are **1** to **254** per set. Valid set size range is **1** to **255**. Enter the command from the probe's configuration mode as follows:

```
(config-probe-SECONDARYCONNECTION)#tolerance rate fail 20 pass 15 of 25
(config-probe-SECONDARYCONNECTION)#
```

The proceeding example specifies that the probe **SECONDARYCONNECTION** will change to a FAIL state if **20** of **25** tests in a row fail, and that it will change to a PASS state if **15** of **25** tests in a row pass.

- Activate the probe using the **no shutdown** command. Enter the command at the probe's configuration mode as follows:

```
(config-probe-SECONDARYCONNECTION)#no shutdown
(config-probe-SECONDARYCONNECTION)#
```

HTTP Request Probe Configuration

HTTP request probes are configured in much the same way as TCP connect probes. There are additional optional parameters for HTTP request probes, detailed in the following section. To configure HTTP probes, follow these steps:

- Specify the name for the probe and the probe type using the **probe** *<name>* [**http-request** | **icmp-echo** | **tcp-connect**] command from the Global Configuration mode prompt. Enter the command as follows:

```
(config)#probe BACKUPSERVER http-request
(config-probe-BACKUPSERVER)#
```

2. Optionally specify the VRF associated with the probe using the **vrf** *<name>* command from the probe's configuration mode. If no VRF is specified, the probe operates on the default (unnamed) VRF. Enter the command as follows:

```
(config-probe-BACKUPSERVER)#vrf server
(config-probe-BACKUPSERVER)#
```

In the previous example, the probe **BACKUPSERVER** is configured to operate in the previously configured VRF **server**.

3. Specify the probe's destination using the **destination** *<ip address | hostname>* **port** *<port>* command from the probe's configuration mode. IP addresses should be expressed in dotted decimal notation (for example, **10.10.10.1**) and host names should be fully qualified (for example, **www.company.com**). Valid port range is **1** to **65535**. By default, HTTP request probes use port **80**. Enter the command as follows:

```
(config-probe-BACKUPSERVER)#destination www.companyserver.com port 2025
(config-probe-BACKUPSERVER)#
```

4. Specify the probe's period using the **period** *<value>* command from the probe's configuration mode. The period specifies the time (in seconds) between the sending of probe packets. Valid range for HTTP request probes is **60** to **65535** seconds, with a default value of **60** seconds. Enter the command as follows:

```
(config-probe-BACKUPSERVER)#period 70
(config-probe-BACKUPSERVER)#
```

5. Optional. Specify the probe's source address by using the **source-address** *<ip address>* command from the probe's configuration mode. IP addresses should be expressed in dotted decimal notation. Enter the command as follows:

```
(config-probe-BACKUPSERVER)#source-address 10.10.10.4
(config-probe-BACKUPSERVER)#
```

6. Optional. Specify the probe's source port by using the **source-port** *<port>* command from the probe's configuration mode. Valid port range is **0** to **65535**. By default, the port is set to **0** so that the probe dynamically selects the port number. To specify the probe's source port, enter the command as follows:

```
(config-probe-BACKUPSERVER)#source-port 200
(config-probe-BACKUPSERVER)#
```

7. Specify the probe's timeout value by using the **timeout** *<value>* command from the probe's configuration mode. The timeout value ranges from **1** to **900000** milliseconds, with an HTTP request probe default timeout value of **10000** milliseconds. Enter the command as follows:

```
(config-probe-BACKUPSERVER)#timeout 7000
(config-probe-BACKUPSERVER)#
```


8. Specify the probe's tolerance by using the **tolerance consecutive [pass <number> | fail <number>]**, **tolerance rate [fail <number> | pass <number>] of <set size>**, or **tolerance rate fail <number> pass <number> of <set size>** commands. Valid range for consecutive passes is **1 to 255**; valid range for consecutive failures is **1 to 255**. Valid ranges for rate passes are **1 to 254** per set; valid ranges for rate failures are **1 to 254** per set. Valid set size range is **1 to 255**. Enter the command from the probe's configuration mode as follows:

```
(config-probe-BACKUPSERVER)#tolerance rate fail 20 pass 15 of 25
(config-probe-BACKUPSERVER)#
```

The proceeding example specifies that the probe **BACKUPSERVER** will change to a FAIL state if **20** of **25** tests in a row fail, and that it will change to a PASS state if **15** of **25** tests in a row pass.

9. Optional. Specify the HTTP request probe type using the **type [get | head | raw]** command. By default, HTTP request probes send GET requests, but you may specify GET, HEAD, or RAW requests. To change the HTTP request probe type, enter the command as follows:

```
(config-probe-BACKUPSERVER)#type head
(config-probe-BACKUPSERVER)#
```

10. Optional. Specify the raw string in an HTTP RAW request using the **raw-string** command. A raw string is the series of HTTP commands placed in the data portion of the probe packet. This type of request can be useful in accessing specific information stored on a server. To enter the **raw-string** command correctly, you must follow a few conventions. After entering **raw-string** in the CLI, the CLI will prompt you for the HTTP commands. You begin these commands with this command: **GET/** followed by the HTTP commands (see *String Variables for HTTP RAW Requests on page 11* for more HTTP command information). When you have completed entering the HTTP commands, you must specify the correct HTTP version (for example, **HTTP/1.0**) followed by two carriage returns (**\r\n\r\n**). Then enter **exit** to exit the HTTP command prompt. Enter the command from the probe's configuration mode as follows:

```
(config-probe-BACKUPSERVER)#raw-string
    GET/update/php?hostname=$SYSTEM_NAME&uptime=$SYSTEM_UPTIME HTTP/1.0
    \r\n
    \r\n
    exit
(config-probe-BACKUPSERVER)#
```

11. Optional. Specify that the HTTP request probe will only accept a particular status from the Web server as passing using the **expect status <minimum> <maximum>** command. This command is useful in verifying that a Web server is returning Web pages to users that request them. The **<minimum>** parameter refers to the minimum HTTP status code acceptable (status codes are detailed on *page 11*). The **<maximum>** parameter refers to the maximum status code acceptable to create a valid range. Both the minimum and maximum codes range from **0 to 999**. Only the minimum parameter is necessary to make this command work, but using the maximum parameter creates a range of valid HTTP status codes acceptable to the probe. Enter the command as follows:

```
(config-probe-BACKUPSERVER)#expect status 200 299
(config-probe-BACKUPSERVER)#
```

In the proceeding example, the expected status was the range from **200 to 299**, indicating that the HTTP probe will fail if a response other than a successful HTTP response is received.

- Optional. Specify that the HTTP request probe return specific content using the **expect regex** *<expression>* command. This command allows the probe to verify not only that information is being sent, but that the correct information is being sent, by configuring the probe to expect a regular expression inside the contents of the HTTP response message. Enter the command as follows:

```
(config-probe-BACKUPSERVER)#expect regex successful
(config-probe-BACKUPSERVER)#
```

In the previous example, the HTTP request probe will now expect the word **successful** in an HTTP response message, or it will fail.

- Optional. Configure the HTTP request probe to follow a specific path using the **absolute-path** *<name>* command. This command is useful in making sure the request goes to a specific location (for example, **/home/index.html**) rather than just a default location (for example, the forward slash /). Enter the command as follows:

```
(config-probe-BACKUPSERVER)#absolute-path /home/index.html
(config-probe-BACKUPSERVER)#
```

- Activate the probe using the **no shutdown** command. Enter the command at the probe's configuration mode as follows:

```
(config-probe-BACKUPSERVER)#no shutdown
(config-probe-BACKUPSERVER)#
```

Probe Configuration Using the GUI

To configure probes using the GUI, follow these steps:

- Open a new Web page in your Internet browser.
- Enter your AOS product's IP address in the Internet browser's address field in the following form:

http://<ip address>. For example:

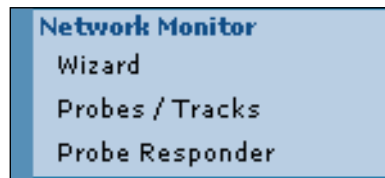
http://65.162.109.200

- At the prompt, enter your user name and password and select **OK**.

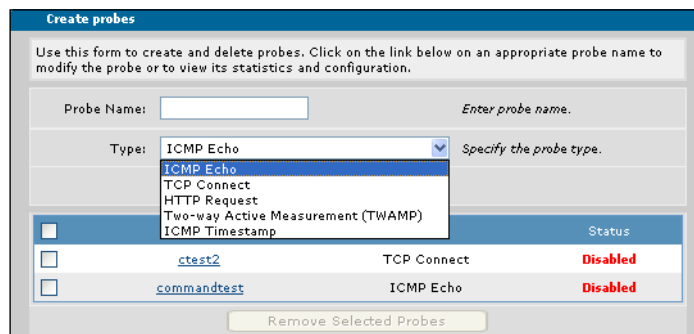


NOTE *The default user name is **admin** and the default password is **password**.*

- Navigate to **Data > Network Monitor > Probes/Tracks**.



- Enter the probe's name and select the probe type from the drop-down menu.



6. Select **Create**.

Use this form to create and delete probes. Click on the link below on an appropriate probe name to modify the probe or to view its statistics and configuration.

Probe Name: Enter probe name.

Type: Specify the probe type.

<input type="checkbox"/>	Probe	Type	Status
<input type="checkbox"/>	ctest2	TCP Connect	Disabled
<input type="checkbox"/>	commandtest	ICMP Echo	Disabled

7. Enable the probe using the check box, enter the **Probe Period**, **Timeout**, and **Destination Hostname**, select the **Tolerance Mode** from the drop-down menu, and optionally enter the probe's **Source IP address**, **Data size**, and **Data pattern**. When all required fields are completed, select **Apply**.

Edit the information for probe Backup Connection below.

Enable: Enable the probe.

Probe Period: (secs) Time between probe test attempts in seconds. (ICMP range 1-65535), (TCP & HTTP range 60-65535)

Timeout: (msecs) Time to wait before declaring test failed (msec). Must be less than the probe-period. (250-4,294,967,295)

Tolerance: Mode: Configure the tolerance and its specifications for probe state transitions.

Destination hostname: Enter destination hostname or an IP address. (required)

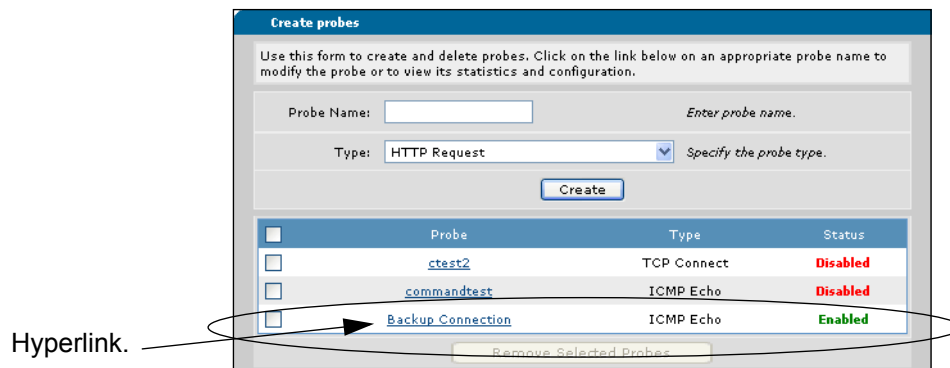
Source IP address: . . . Enter source IP address. (optional)

Data size: Length of the ICMP packet (0-1462)

Data pattern: Hexadecimal pattern for the ICMP packet (without the 0x)

NOTE *TCP connect and HTTP request probes will have additional configurable parameters on this menu. Fill in the required and desired optional fields and select **Apply**.*

8. The probe has been created and appears in the list of probes. The probe's configuration can be edited by selecting the probe's name hyperlink.



Configuring Tracks

Tracks are objects that monitor probes, and take actions based on the probe's state. Actions performed by tracks include removing faulty routes (both static and negotiated) and logging changes. To create a network monitor track, you must define the following track parameters:

- Name
- Track/probe associations and track test conditions
- Dampening interval
- Logging preferences
- Activation/deactivation

Track Name

The track's name is the identifier of the track. You should name the track something that makes it easy to identify on your network, such as naming a track to monitor your Web server **mywebserver**. Multiple tracks can be created and configured, but you should be aware the more active tracks and probes you have running, the more bandwidth is consumed.

Track/Probe Associations and Track Test Conditions

Tracks must be associated with specific probes, in order to monitor their states and take actions accordingly. Tracks can use one or more probes to monitor the network. If only one probe is used, the track will fail if the probe fails. If more probes are used, the track will change states when either one of the probes changes states, or when all of the probes change states. Using multiple probes can be beneficial in instances where there are two default gateways being monitored, when more than one server is being monitored, or when a single server is being monitored in more than one way. For example, in monitoring a network with two default gateways, the track should be configured to pass when either of two probe passes to avoid removing a valid route. In monitoring multiple servers, the track should be set to pass only if all probes pass.

Track test conditions are based in Boolean logic, and are used to monitor objects associated with the track. The tracks can test a single item, whether it is an interface, probe, or a schedule. If the tracks are only testing a single item, their state is wholly dependent on the state of the object they are testing. By negating the results of a single test, the track can be configured to be the opposite state of the object it monitors. Interfaces can be monitored by tracks to test their ability to perform IP routing and their line protocol states. Probes are monitored so that tracks can take action based on their states, and schedules are monitored so that tracks can operate at specified times. Most common network monitoring applications involve tracks monitoring probes and schedules.

Tracks can also test multiple items at once. If tracks are testing multiple items, they rely on the Boolean AND, OR, and on weighted tests. If the track is using AND logic, all tests in the test list must be successful for the track to remain in a PASS state. If the track is using OR logic, any successful test in the list will cause the track to pass. If the track is using weighted logic, each entry in the list is given a weight. A threshold weight is then specified, and the track will be in a PASS state when the sum of the weights of the successful tests in the list is above the threshold.

Track Dampening Interval

The track dampening interval configures the track to delay any state changes in response to changes in an associated probe's state. The dampening interval is necessary to keep the track from making changes that could affect an entire network (like removing a vital route) and serves as a safety precaution against the track taking actions when the probe state change may be faulty. For example, a dampening interval of 10 seconds forces a track to stay in the PASS state for 10 seconds after its associated probe has changed to a FAIL state. The track takes no action during this delay, and if the probe returns to a PASS state in that time, no action is taken by the track. In essence, the dampening interval prevents fluctuations in the routing table, the creation of excess logs, and severe track actions based on probes that report false positives. The dampening interface ranges from 1 to 4294967295 seconds, with a default value of 1 second.

Track Logging Preferences

Tracks can create a log, recording every time they change states. This log is helpful in alerting you that some network conditions have changed. It is beneficial to have a log because it can tell you when a network has become unavailable and a route is removed. These logs are different from debug output because they appear in the configuration of the unit, outlast a terminal session, and can be saved to persist across a restart. The logs can also be forwarded to a syslog server.

Track Activation/Deactivation

Unlike probes, tracks are activated when they are created. A track must be manually shut down, which forces it into a FAIL state. Tracks should not be shut down solely to stop them from monitoring the network. If the track is monitoring a route, shutting it down could result in the removal of the route. If you want the track to stop monitoring the network, shut down the probes associated with the track.

Track Configuration Using the CLI

To configure a track using the CLI, follow these steps:

1. Create and name the track using the **track** *<name>* command from the Global Configuration mode prompt. Enter the command as follows:

```
(config)#track BACKUPROUTE
(config-track)#
```
2. Associate the appropriate test objects with the track and specify the track's testing conditions using the **test if**, **test list**, or **test list weighted** commands.

Track test if Command

Using the **test if** command allows you to select one object to be associated with and tested by the track. These objects can be interfaces, probes, schedules, Y.1731 MEPs, RapidRoute flow entries, or voice users. The basic syntax for this command is **test if [not]** *<object>* *<test parameter>*. The *<object>* parameter is the object to be tested, and *<test parameter>* is the parameter(s) of the object you want to test. The **not** parameter allows the results of the object test to be negated, using Boolean NOT logic. Using this logic may be beneficial in an instance where a track should be in a PASS state when an associated schedule (or other object) is inactive. To associate a single probe with the created track, enter the command as follows:

```
(config-track)#test if probe BACKUPCONNECTION
```

In the previous example, the track will mirror the state of probe **BACKUPCONNECTION**. If the probe is in a PASS state, the track is in a PASS state. Conversely, if the probe is in a FAIL state, the track is in a FAIL state. This command functions in the same manner no matter the object you are testing.

To take advantage of the Boolean **not**, enter the command as follows:

```
(config-track)#test if not probe BACKUPCONNECTION
```

In this example, the track state is the opposite of the probe state. If the probe is in a PASS state, the track is in a FAIL state. Conversely, if the probe is in a FAIL state, the track is in a PASS state.



*There are several objects that can be tested and several test parameters that accompany these objects. For a complete listing of available **test if** commands, refer to [Track test if Command Syntax on page 27](#).*

Track test list Commands

Using the **test list** command allows you to select multiple objects to be associated with and tested by the track. These objects can also be interfaces, probes, schedules, Y.1731 MEPs, RapidRoute flow entries, or voice users. This command can also employ both Boolean logic and weighted logic, using the Boolean AND or OR, as well as weighted. The AND logic specifies the relationship between all objects placed in the list. This means that all objects in the list must be in the PASS state for the track to pass, or at least one object to be in a FAIL state for the track to fail. The OR logic also specifies the relationship between all objects in the list, but means that only one of the objects must be in the PASS state for the track to pass, and all objects in the FAIL state for the track to fail. Weighted logic gives each list entry a specified weight, and creates a threshold weight. The track remains in a PASS state

when the sum of the weights of the successful tests in the list is above the threshold weight. There is no limit to how many objects can be tested within a single test list; however, only one type of test list (**and**, **or**, or **weighted**) can exist on a track at any given time. The full syntax for this command is: **test list [and | or | weighted]**. When the **test list** command is entered, you must select whether you will be using **and**, **or**, or **weighted** logic.

test list [and | or] Command (Boolean Track Test List Configuration Mode)

If you select **and** or **or**, you will enter the Boolean Track Test List Configuration mode and will be prompted to add specific tests to the list. The basic syntax for the command to test objects in this mode is **if [not] <object> <test parameter>**. The **<object>** parameter is the object to be tested, and the **<test parameter>** is the parameter(s) of the object you want to test. The **not** parameter allows the results of the object test to be negated. Once you have added all the objects you wish to track to the test list, enter **exit** at the prompt. To associate multiple probes (or other objects) with a created track, using **AND** or **OR** logic, enter the command as follows:

```
(config-track)#test list and
(config-track-test)#if not probe PRIMARYCONNECTION
(config-track-test)#if probe BACKUPSERVER
(config-track-test)#exit
(config-track)#
```

In the previous example, the track **BACKUPROUTE** will be in a PASS state when both the probe **PRIMARYCONNECTION** is in a FAIL state (indicated by the **not** parameter) and the probe **BACKUPSERVER** is in a PASS state. What this indicates is that the track **BACKUPROUTE** will not become active until the primary connection has failed (indicated by the probe associated with the primary connection) and until the backup server is available (indicated by the probe associated with the backup server). This configuration associates the two probes with the track, and governs its actions based on the states of the two probes. A route will be associated with this track later (refer to [Track/Route Associations on page 39](#)) and the route will be placed in the routing table when the track becomes active (in a PASS state). The route is removed when the track becomes inactive (in a FAIL state). This is a simplified example of using tracks and probes to govern primary and backup connections.



*There are several objects that can be tested and several test parameters that accompany these objects. For a complete listing of available **test list [and | or]** commands, refer to [Track test list \[and | or\] Command Syntax on page 29](#).*

test list weighted Commands (Weighted Track Test List Configuration Mode)

If you select **weighted** as your logic option, you will enter the Weighted Track Test List Configuration mode and will be prompted to add specific tests to the list, as well as specify weights or thresholds for each test. The basic syntax for the command to test objects in this mode is **if [not] <object> <test parameter> weight <value>**. The **<object>** parameter is the object to be tested, and the **<test parameter>** is the parameter(s) of the object you want to test. The **weight <weight>** parameter specifies the weight value to use if the test is successful. Valid weight range is **1** to **65535**.

You can optionally choose to not specify a weight, but rather enter the **threshold <number>** or **threshold pass <number> fail <number>** commands in addition to the **if [not] <object> <test**

parameter> command(s). If you only use the **threshold** <number> command, you will have a single baseline weight that all successful test weights must reach for the track to change to a PASS state. If the combined weight of all successful test weights in the list does not reach the baseline threshold, the track is in a FAIL state. If you use the **threshold pass** <number> **fail** <number> command, you are specifying a specific weight for the track to change to a PASS state, and a specific weight for the track to change to a FAIL state. In specifying weights and thresholds, it is important to remember that the threshold value must be met for the track to be in a PASS state. Therefore, you should configure your list's item weights according to whether one item or more will have to be successful for your track to be in a PASS state. Once you have added all the objects you wish to track, enter **exit** at the prompt.

To associate multiple items with a track using **weighted** logic, enter the command as follows:

```
(config-track)#test list weighted
(config-track-test)#if probe BACKUPSERVER 20
(config-track-test)#if probe SECONDARYCONNECTION 10
(config-track-test)#if not probe PRIMARYCONNECTION 30
(config-track-test)#threshold pass 35 fail 30
(config-track-test)#exit
```

In the previous example, the track **BACKUPROUTE** gains its state from the three associated probes, **BACKUPSERVER**, **SECONDARYCONNECTION**, and **PRIMARYCONNECTION**. The pass threshold in this example is set to **35**. The sum of all the weights must meet or exceed 35 before the track will transition to a PASS state. The fail threshold in this example is set to **30**. If the sum of all weights meets or falls below the value of 30, the track will transition to a FAIL state. This configuration has the same end as the AND/OR test list in the previous example, where the **BACKUPROUTE** track will only pass when the probe monitoring the primary connection is down, and either the backup server or the secondary connection is up. Because the combined weights of the **PRIMARYCONNECTION** probe and one of the other probes exceeds the pass threshold of **35**, the track will pass when these conditions are met. Conversely, the track fails if the combined weight equals or falls below the fail threshold of **30**, which happens if the probe monitoring the primary connection is in a PASS state (note: the **not** keyword negates the result of the **PRIMARYCONNECTION** probe test, meaning that if the probe passes, this test negates the result, reports the test as unsuccessful, and does not count the probe's weight). This configuration allows the backup route to take effect if the primary connection fails.



*There are several objects that can be tested and several test parameters that accompany these objects. For a complete listing of available **test list weighted** commands, refer to [Track test list weighted Command Syntax on page 30](#).*

- Specify the track's dampening interval using the **dampening-interval** [<value> | **fail** <value> | **pass** <value>] command. This command specifies the amount of time (in seconds) that the track will wait before allowing an associated probe's state change to trigger the track's state change. The <value> parameter indicates the time in seconds and ranges from **1** to **4294967295** seconds. The **fail** keyword indicates that the delay will occur following pass-to-fail transitions. The **pass** keyword indicates the delay will occur following fail-to-pass transitions. If neither **fail** nor **pass** is selected, the entered value will apply to both conditions. By default, the dampening interval is set to **1** second. This command can be beneficial in preventing network fluctuations, the creation of excess logs, and severe track actions based on probes that report false positives. To specify the track's dampening interval, enter the command as follows:

```
(config-track)#dampening-interval fail 30
(config-track)#
```

When configuring the track's dampening interval, it is important to remember the period and timeout for any probes associated with the track. The dampening interval should be long enough that the probe can get a successful response and change state before the track takes action.



*The **tolerance consecutive** command used in the probe configuration can also help prevent flapping tracks and probes by requiring a particular number of consecutively successful or failed probe tests before changing the state of the probe. This command can be used in conjunction with, or as an alternative to, the **dampening-interval** command depending on the probe's application and configuration.*

- Specify the track's logging behavior by using the **log-changes** command. This command enables the logging of track state changes to the terminal screen or event history. Enter the command as follows:

```
(config-track)#log-changes
(config-track)#
```

These logs can also be forwarded to a syslog server using the **logging forwarding** commands. To enable logging forwarding, enter these commands from the Global Configuration mode:

```
(config)#logging forwarding on
(config)#logging forwarding receiver-ip 172.5.67.99
(config)#logging forwarding priority-level notice
```

These commands enable logging forwarding, specify the IP address of the receiving server (IP address expressed in dotted decimal notation), and specify that the events logged are at **notice** priority level. Track logs have a priority of **notice** (level 3), so you must make sure the priority for logged events or forwarded logs is set at this level or above.

You can also receive track logs via email, using ADTRAN's generic mail agent. For more information about configuring the mail agent, refer to the *Generic Mail Agent Quick Configuration Guide* available online at <https://supportforums.adtran.com/>.

- Optional. Deactivate the track. Unlike probes, tracks are active as soon as they are created. You can manually shut down the track, using the **shutdown** command from the track's configuration mode, which forces the track into a permanent FAIL state. To shut down the track manually, enter the command as follows:

```
(config-track)#shutdown
(config-track)#
```



If the track is monitoring a route, the route will be removed if the track is shut down. This can cause network issues if the route is a vital one. If you want to stop the track from monitoring the network, shut down the probes associated with the track, not the track itself.

Track Test Commands Complete Command Syntax

The three track test commands (**test if**, **test list [and |or]**, and **test list weighted**), have several objects and parameters that they can test. The following section outlines the available options for each track test type.

Track test if Command Syntax

The Track Test Configuration mode is entered by issuing the **test if [not] <object> <test parameter>** command from the track's configuration mode (refer to [Track test if Command on page 23](#)). The *<object>* parameter is the object to be tested, and *<test parameter>* is the parameter(s) of the object you want to test. Variations of the test commands within this command set include:

1. The testing of Y.1731 MEP objects using the **test if [not] ethernet y1731 meg [char-string <name> | icc-umc <name>] <level> <id> [loc | rdi]** command. Syntax description for this command is described in [Table 4](#).

Table 4. Syntax Description for Testing Y.1731 MEPs

Syntax	Description
ethernet y1731 meg	Specifies a Y.1731 MEP as the object to be tested.
char-string <name>	Specifies a Y.1731 maintenance entity group (MEG) name using a character string format. Maximum length is 45 ASCII characters.
icc-umc <name>	Specifies a Y.1731 MEG name using the ITU-Carrier Code Unique MEG ID Code MEG (ICC-UMC) format. Maximum length is 13 ASCII characters.
<level>	Specifies the MEG level. Valid range is 0 to 7 .
<id>	Specifies the MEP ID. Valid range is 1 to 8191 .
loc	Specifies that the track will inspect the specified MEP for indication of loss of continuity (LOC). The test will report TRUE if the specified MEP reports an LOC condition.
rdi	Specifies that the track will inspect the specified MEP for remote defect indication (RDI). The test will report TRUE if the specified MEP reports an RDI condition.
not	Optional. Negates the test results when specifying a single object to be tested.

2. The testing of interfaces, including the system control and system management EVCs, using the **test if [not] interface [<interface> | system-control-evc | system-management-evc] [downspeed <speed> | ip-routing | ipv6-routing | line-protocol | upspeed <speed>]** command. Syntax description for this command is described in [Table 5 on page 28](#)

Table 5. Syntax Description for Testing Interfaces

Syntax	Description
interface <interface>	Specifies an interface as the object to be tested. Specify an interface in the format <interface type [slot/port slot/port.subinterface id interface id interface id.subinterface id]>. For example, for an Ethernet subinterface, use eth 0/1.1 ; for a PPP interface, use ppp 1 ; for an ATM subinterface, use atm 1.1 ; and for a virtual local area network interface, use vlan 1 . Type test if interface ? for a complete list of valid interfaces. Included as an interface are the system-control-evc and system-management-evc parameters, which specify the system control EVC and system management EVC respectively. Enter these commands as interface system-control-evc or interface system-management-evc .
downspeed <speed>	Specifies the downstream speed (in kilobits per second) will be tested. The downstream speed is the receive speed of an interface from the perspective of the unit. The test will report TRUE if the interface downstream speed is greater than or equal to the specified downstream speed.
ip-routing	Specifies the interface's ability to perform Internet Protocol version 4 (IPv4) routing will be tested.
ipv6-routing	Specifies the interface's ability to perform Internet Protocol version 6 (IPv6) routing will be tested.
line-protocol	Specifies the line-protocol state of an interface will be tested.
upspeed <speed>	Specifies the upstream speed (in kilobits per second) will be tested. The upstream speed is the transmit speed of an interface from the perspective of the unit. The test will report TRUE if the interface upstream speed is greater than or equal to the specified downstream speed.
not	Optional. Negates the test results when specifying a single object to be tested.

- The testing of probes using the **test if [not] probe** <name> command. The <name> parameter is the name of the probe to be tested. The optional **not** parameter negates the test results.
- The testing of schedules using the **test if [not] schedule** <name> command. The <name> parameter is the name of the schedule to be tested. The optional **not** parameter negates the test results.
- The testing of RapidRoute flow entries using the **test if [not] [ip | ipv6] ffe** [<ingress interface>] **entries less-than** <number> command. This command tests that the number of RapidRoute flow entries is less than the specified number for IPv4 or IPv6 traffic. Syntax for this command is described in [Table 6](#).

Table 6. Syntax Description for Testing RapidRoute Flow Entries

Syntax	Description
ip	Specifies that IPv4 RapidRoute is tested.
ipv6	Specifies that IPv6 RapidRoute is tested.

Table 6. Syntax Description for Testing RapidRoute Flow Entries (Continued)

Syntax	Description
<i><ingress interface></i>	Optional. Specifies that only a single ingress interface is tested for RapidRoute entries. Specify an interface in the format <i><interface type [slot/port slot/port.subinterface id interface id interface id.subinterface id]></i> . For example, for an Ethernet subinterface, use eth 0/1.1 ; for a PPP interface, use ppp 1 ; for an ATM subinterface, use atm 1.1 ; and for a virtual local area network interface, use vlan 1 . If no interface is specified, the global RapidRoute entry count is tested.
entries less-than <i><number></i>	The specified maximum number of RapidRoute flow entries for the test. Valid range is 1 to 500000 entries.
not	Optional. Negates the test results when specifying a single object to be tested.

- The testing of voice users using the **test if [not] voice user <extension> registered** command. This command tests the Session Initiation Protocol (SIP) registration status of the specified voice user. The *<extension>* parameter is the extension of the SIP user to test. The optional **not** parameter negates the test results.

Track test list [and | or] Command Syntax

The Boolean Track Test List Configuration mode is entered by issuing the **test list [and | or]** command from the track's configuration mode (refer to *Track test list Commands on page 23*). Once in this mode, specify the objects to be tested using the **if [not] <object> <test parameter>** command. The *<object>* parameter is the object to be tested, and *<test parameter>* is the parameter(s) of the object you want to test. Variations of the test commands within this command set include:

- The testing of Y.1731 MEP objects using the **if [not] ethernet y1731 meg [char-string <name> | icc-umc <name>] <level> <id> [loc | rdi]** command. Syntax description for this command is described in *Table 4 on page 27*.
- The testing of interfaces, including the system control and system management EVCs, using the **test if [not] interface [<interface> | system-control-evc | system-management-evc] [downspeed <speed> | ip-routing | ipv6-routing | line-protocol | upspeed <speed>]** command. Syntax description for this command is described in *Table 5 on page 28*.
- The testing of probes using the **if [not] probe <name>** command. The *<name>* parameter is the name of the probe to be tested. The optional **not** parameter negates the test results.
- The testing of schedules using the **if [not] schedule <name>** command. The *<name>* parameter is the name of the schedule to be tested. The optional **not** parameter negates the test results.
- The testing of RapidRoute flow entries using the **if [not] [ip | ipv6] ffe [<ingress interface>] entries less-than <number>** command. This command tests that the number of RapidRoute flow entries is less than the specified number for IPv4 or IPv6 traffic. Syntax for this command is described in *Table 6 on page 28*.

6. The testing of voice users using the **if [not] voice user** *<extension>* **registered** command. This command tests the Session Initiation Protocol (SIP) registration status of the specified voice user. The *<extension>* parameter is the extension of the SIP user to test. The optional **not** parameter negates the test results.

Track test list weighted Command Syntax

The Weighted Track Test List Configuration mode is entered by issuing the **test list weighted** command from the track's configuration mode (refer to *Track test list Commands on page 23*). Once in this mode, specify the objects to be tested using the **if [not]** *<object>* *<test parameter>* command. The *<object>* parameter is the object to be tested, and *<test parameter>* is the parameter(s) of the object you want to test. Variations of the test commands within this command set include:

1. The testing of Y.1731 MEP objects using the **if [not] ethernet y1731 meg [char-string** *<name>* | **icc-umc** *<name>*] *<level>* *<id>* **[loc | rdi] weight** *<value>* command. Syntax description for this command is described in *Table 4 on page 27*.
2. The testing of interfaces, including the system control and system management EVCs, using the **if [not] interface** [*<interface>* | **system-control-evc** | **system-management-evc**] **[downspeed** *<speed>* | **ip-routing** | **ipv6-routing** | **line-protocol** | **upspeed** *<speed>*] **weight** *<value>* command. Syntax description for this command is described in *Table 5 on page 28*.
3. The testing of probes using the **if [not] probe** *<name>* **weight** *<value>* command. The *<name>* parameter is the name of the probe to be tested. The optional **not** parameter negates the test results.
4. The testing of schedules using the **if [not] schedule** *<name>* command. The *<name>* parameter is the name of the schedule to be tested. The optional **not** parameter negates the test results.
5. The testing of RapidRoute flow entries using the **if [not] [ip | ipv6] ffe** [*<ingress interface>*] **entries less-than** *<number>* **weight** *<value>* command. This command tests that the number of RapidRoute flow entries is less than the specified number for IPv4 or IPv6 traffic. Syntax for this command is described in *Table 6 on page 28*.

Track Configuration Using the GUI

To configure tracks using the GUI, follow these steps:

1. Open a new Web page in your Internet browser.
2. Enter your AOS product's IP address in the Internet browser's address field in the following form:

http://<ip address>. For example:

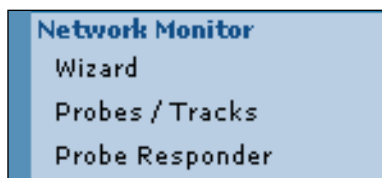
http://65.162.109.200

3. At the prompt, enter your user name and password and select **OK**.

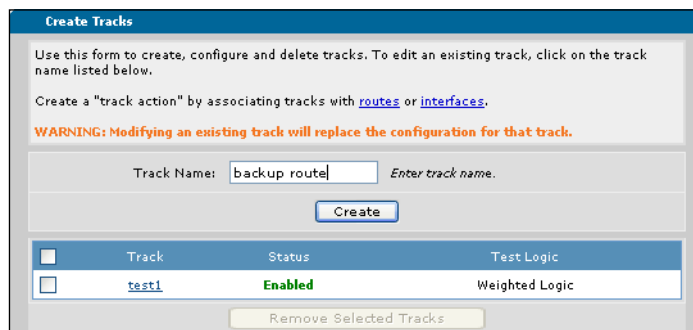


*The default user name is **admin** and the default password is **password**.*

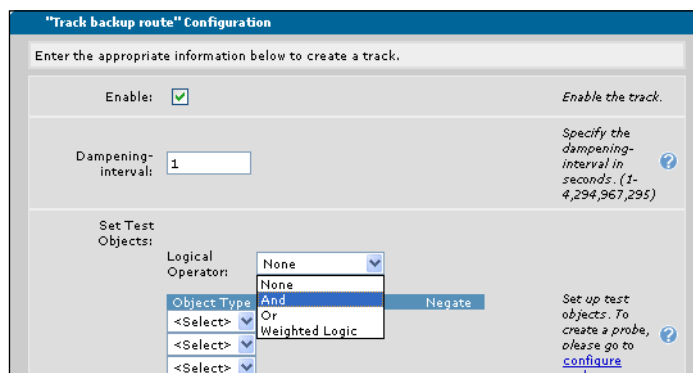
4. Navigate to **Data > Network Monitor > Probes/Tracks**.



5. Scroll to the bottom of the menu, enter the track's name in the appropriate field, and select **Create**.



6. Enable the track using the check box, enter the desired **Dampening-interval**, and select the **Logical Operator** from the drop-down menu. If you are only monitoring one object with the track, select **None**. If you are monitoring more than one object, select **And**, **Or**, or **Weighted Logic**. For more information on the types of logic, refer to the logic descriptions in Step 2 of *Track Configuration Using the CLI on page 23*.



7. Once you have selected the type of logic for monitoring the objects, select the objects you want the track to monitor from the drop-down menus. You can choose from **probes** or **schedules**. Select one, and from the next drop-down menu, select the probe or schedule name. You can also negate the state of the object

by selecting the **Negate** box. The negation works in the same way the **not** operator does in the CLI (see [page 23](#)).

"Track backup route" Configuration

Enter the appropriate information below to create a track.

Enable: *Enable the track.*

Dampening-interval: *Specify the dampening-interval in seconds. (1-4,294,967,295)*

Set Test Objects:

Logical Operator:

Object Type	Object	Negate
<input type="text" value="Probe"/>	<input data-bbox="711 598 836 625" type="text" value="Backup Conn..."/>	<input type="checkbox"/>
<input type="text" value="Probe"/>	<input type="text" value="test"/>	<input type="checkbox"/>
<input type="text" value="<Select>"/>		
<input type="text" value="<Select>"/>		
<input type="text" value="<Select>"/>		

Set up test objects. To create a probe, please go to [configure probe](#).

Execute TCL: *Check to execute a TCL script when the track state transitions.*

- After entering all the objects you wish to associate with this track, you can also choose to have the track run a Tcl script upon state change by selecting the appropriate box.

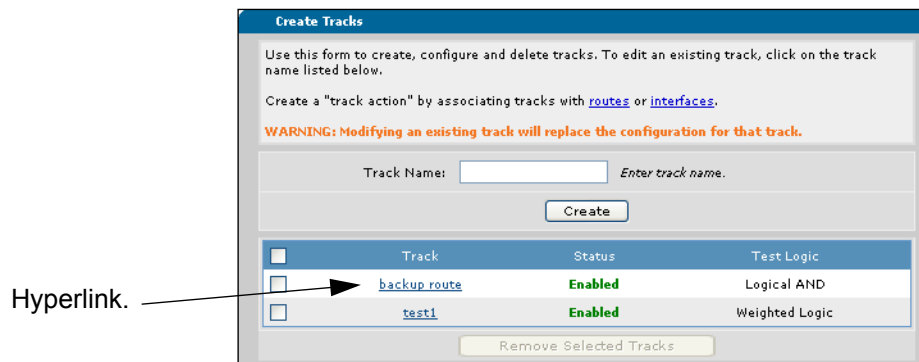
Execute TCL: *Check to execute a TCL script when the track state transitions.*

Track State:

Script:

You must have already configured a Tcl script before selecting this option. The benefits of this option are discussed in more detail on [page 52](#). For more information on creating a Tcl script, refer to the *Tcl Scripting in AOS* configuration guide available online at <https://supportforums.adtran.com>.

- Once you have entered all the correct information in the required fields, select **Apply** to add the information to the track. The track is then created and appears in the track list, which displays the name of the track, the track's status, and the test logic the track uses. To modify the track's configuration, select the track's name hyperlink.



Configuring Schedules

Schedules monitor the time of day and day of the week, and are used to determine what times during the day and how often tracks and probes are active. They are beneficial in creating reoccurring network monitoring tests, and can be used to schedule any object that can be tracked (for example, ACLs, crypto maps, static routes, and Tcl scripts).

Configurable Schedule Parameters

To configure a schedule, you must define the following schedule characteristics:

- Schedule type
- Schedule duration
- Schedule time

Schedule Type

Three types of schedules are available: absolute, relative, and periodic. Absolute schedules are active only once, for a specified amount of time. These schedules begin at an absolute month, day, year, and time. These schedules are beneficial for running a one-time network monitoring test, perhaps scheduled after peak network usage hours to avoid overloading the network. Absolute schedules have start and end times and dates expressed in the format `<time> <day> <month> <year>` (for example, **08:15 2 February 2008**).

Relative schedules become active after a specified delay. The delay begins when the delay command is entered into the unit's configuration. Relative schedules are beneficial in situations where the track or probe's action needs to be delayed (for example, in delaying the activation of a backup route until after the primary route has had time to connect). Valid range for delay times is **1** to **65535** seconds.

Periodic schedules transition from active to inactive at configured periods. These periods can be daily, weekly, on weekdays, or on weekends. Periodic schedules are useful in creating reoccurring network monitoring tests. Periodic schedules can be configured to begin at a certain date and time and run for a specified duration, or they can be configured to start and end at specific times.

Schedule Duration

Schedule duration is the length of time the schedule is active. Schedules can be active for as long as you need them to be, up to 24 hours. Duration is expressed in the 24-hour format of hours:minutes (hh:mm).

Schedule Time

A schedule's time is the start or end time or date for the schedule. Time is expressed in the 24-hour format of hours:minutes (hh:mm). Dates are expressed using day, month, and year values. The day of the month (for absolute schedules) is expressed with a number, ranging from 1 to 31. The day of the week (for periodic schedules) can be spelled out or abbreviated. The month can also be spelled out or abbreviated. The year is expressed in 4-digit format (yyyy).



To use a schedule effectively, it is important that the time of day on the AOS product be set correctly. AOS has a built-in Network Time Protocol (NTP) client to accurately set the clock. Most schedule applications will require that the NTP client is configured correctly.

Schedule Configuration Using the CLI

Schedules can only be configured using the CLI. A schedule is configured by specifying the following criteria:

- Schedule name
- Schedule type
- Schedule start/end and duration

To configure a schedule using the CLI, follow these steps:

1. Create a schedule using the **schedule** <name> command entered from the Global Configuration mode prompt. Creating a schedule enters the Schedule Configuration mode. Enter the command as follows:

```
(config)#schedule RUNTEST
(config-schedule-RUNTEST)#
```

Using the **no** form of this command removes the specified schedule from the unit's configuration.

2. Specify the schedule type, start time, and end time using the **absolute start**, the **relative start-after**, or the **periodic** commands.

To create an absolute schedule, you must enter the **absolute start** command and specify the time, day, month, and year of the schedule's start and end times. The command follows this syntax: **absolute start** <time> <day> <month> <year> **end** <time> <day> <month> <year>. Time is expressed in the 24-hour format (hh:mm). The day is the day of the month, expressed with a number between 1 and 31. The month is expressed by name, and can be spelled out or abbreviated. The year is in four-digit format (yyyy). Enter the command from the schedule's configuration mode as follows:

```
(config-schedule-RUNTEST)#absolute start 17:30 1 November 2008 end 18:00 1 November 2008
```

The absolute schedule in the previous example will run one time only, from 5:30 p.m. to 6:00 p.m. on November 1, 2008.

To create a delayed schedule, you must enter the **relative start-after** *<delay>* command. The delay parameter is the time (in seconds) that you want the schedule to delay before becoming active. Valid range for the delay is **1** to **65535** seconds. Enter the command from the schedule's configuration mode as follows:

```
(config-schedule-RUNTEST)#relative start-after 20
```

The delayed schedule in the previous example will wait **20** seconds before becoming active.

To create a periodic schedule, you must enter the **periodic** command. You can choose to configure a periodic schedule using a variety of command variations. To specify a periodic schedule that occurs on a specific day (or range of days) and begins and ends at specified times, you would use the **periodic** *<day>* *<time>* **to** *<time>* command. In this case, the day parameter can be one day of the week (**monday**, for example) or could be up to seven days of the week (**monday tuesday thursday**, for example). The **to** parameter denotes that you are configuring the schedule's start and end times. Enter the command as follows:

```
(config-schedule-RUNTEST)#periodic monday wednesday friday 17:30 to 18:00
```

The previous periodic schedule is configured to run three days a week (**monday**, **wednesday**, and **friday**) from 5:30 p.m. to 6:00 p.m.

To specify a periodic schedule that occurs on a specific day (or range of days) and begins at a certain time, but runs for a duration of time, you would use the **periodic** *<day>* *<time>* **for** *<time>* command. The **for** parameter denotes that you are configuring the schedule's duration. The duration is entered in the hh:mm format. To create a schedule that runs every Monday, beginning at 6:00 p.m., that runs for two hours, enter the command as follows:

```
(config-schedule-RUNTEST)#periodic monday 18:00 for 02:00
```

Periodic schedules can also be configured to run **daily**, to run Monday through Friday (**weekday**), or to run on the **weekend**. These schedules are created by using the **periodic** [**daily** | **weekday** | **weekend**] *<time>* [**to** | **for**] *<time>* command. Again, the **to** parameter denotes that you are specifying a start and end time for the schedule, and the **for** parameter denotes that you are specifying a duration for the schedule to be active. To create a schedule that runs Monday through Friday, beginning each day at 5:00 a.m. and running for an hour, enter the command as follows:

```
(config-schedule-RUNTEST)#periodic weekday 05:00 for 01:00
```

Associating Schedules with Tracks

Schedules are associated with tracks when it is beneficial for tracks to begin monitoring probes at specified times. Tracks may not need to be active right away if, for example, they are monitoring a primary connection that takes a few minutes to connect on startup. Such a track may have a delayed schedule associated with it, which allows the track to wait for the primary connection to get connected.

There are two ways to associate schedules and tracks: through the track's CLI configuration, and through the GUI. Each method is detailed in the following sections.

Scheduling Using the Track's CLI Configuration

There are two methods of associating tracks and schedules in the track's CLI configuration. The two commands for this purpose are: **test if schedule** <name> (or **test list**) and **time-schedule** <name>.



*For more information on how the **test if** and **test list** commands operate, review [Track/Probe Associations and Track Test Conditions on page 21](#).*

Use the **test if schedule** or the **test list** commands to specify that the track's state will be dependent on the state of the schedule. For example, the schedule may not be active (in a FAIL state), and depending on how you have the track configured, the track may or may not be in a FAIL state also based on the schedule. The schedule is entered in the same configuration area as the probes that are to be associated with the track (see [page 21](#)), and are entered in the same manner. The logic that applies to probe and track associations also apply to schedule and track associations. The example below builds on the example given on [page 21](#), and includes a preconfigured schedule named **DELAY**. The schedule was configured to delay the activation of the track for 30 seconds after boot to give the primary connection time to connect. Adding the **DELAY** schedule to the track's configuration specifies that the track will not be in a PASS state unless the schedule is active (PASS), and the two probes are in their correct states. To have a track's condition (PASS or FAIL) defined by the state of its associated probes and a schedule, enter the commands from the track's configuration mode as follows:

```
(config-track)#test list and
(config-track-test)#if not probe PRIMARYCONNECTION
(config-track-test)#if probe BACKUPSERVER
(config-track-test)#if schedule DELAY
(config-track-test)#exit
(config-track)#
```

In the previous example, the track **BACKUPROUTE** will not be in a PASS state until these conditions occur: the probe **PRIMARYCONNECTION** is in a FAIL state (indicated by the **not** keyword), the probe **BACKUPSERVER** is in a PASS state, and the schedule **DELAY** is active (PASS state). The association between the track and the schedule can be configured similarly to probes in all aspects. Refer to [Track/Probe Associations and Track Test Conditions on page 21](#).

Use the **time-schedule** <name> command to specify the time period a track is in effect. This method allows you to define when the track is monitoring the probes or other objects, rather than specifying that the track's state is dependent upon the schedule's state (as with the **test if** or **test list** commands). Using this command allows the track to be either active or inactive, rather than only passing or failing. It is possible to specify that the track be in a certain state (PASS or FAIL) when the schedule is inactive, if necessary. The **time-schedule** command provides a way in which routes or other objects being tracked are not affected when the track is inactive by either keeping the track from monitoring the objects when the schedule is inactive, or by specifying the state of the track when the schedule is inactive. The full syntax of the command is as follows: **time-schedule** <name> [**pass** | **fail**]. As with the **test if** and **test list** commands, the schedule must already be configured before attempting to associate it with a track. To associate the track **BACKUPROUTE** with the schedule **DELAY**, and specify that the track is not monitoring any probes when the schedule is inactive, enter the command as follows:

```
(config-track)#time-schedule DELAY
```

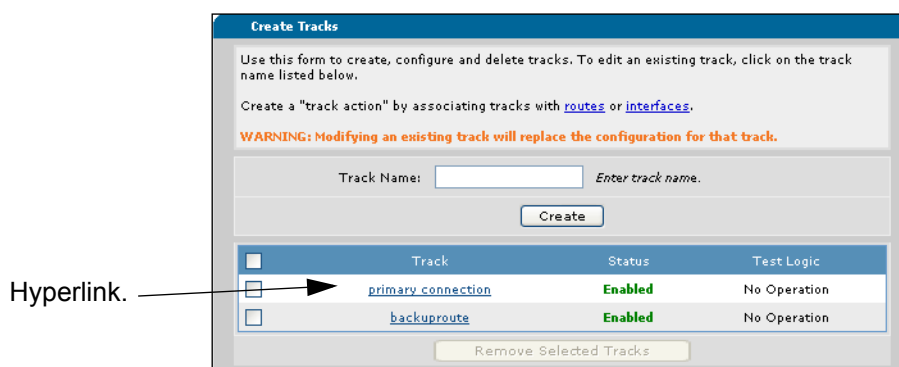
To associate the same track and schedule, and specify that the track must be in a PASS state when the schedule is inactive, enter the command as follows:

(config-track)#time-schedule DELAY pass

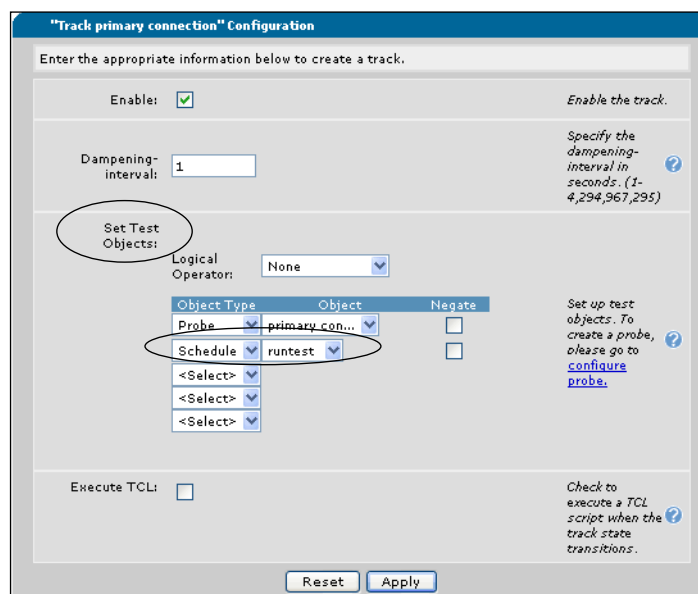
Scheduling Using the Track's GUI Configuration

Using the GUI, the track can be configured to monitor schedules, and therefore, be associated with a schedule. In this configuration, the track will be in a PASS or FAIL state depending on the logic chosen for the track and the state of the schedule and other objects the track is monitoring. To add a schedule to the track's monitoring list, follow these steps:

1. After connecting to the GUI, navigate to **Data > Network Monitoring > Probes/Tracks**. Select the track to which you wish to add the schedule, and follow the track's hyperlink (the track name).



2. Once in the track's configuration menu, under **Object Type** select **Schedule** from the drop-down menu. Then select the schedule's name from the drop-down menu, and whether to negate the schedule using the check box. After all the parameters are entered, select **Apply**. The schedule is now associated with the track.



Track/Route Associations, Policy Based Routing, and Other Track Associations

Tracks are associated with routes (both static and default) because tracks can be configured to remove faulty routes from the route table. PBR is used to specify the route that probes are using and monitoring, and is necessary to ensure that Network Monitor is monitoring the correct parts of your network. Tracks can also be associated with other network configurations, such as ACLs, crypto maps, and Tcl scripts through specific application configurations (refer to *Network Monitor Common Configurations and Applications on page 48*).

Track/Route Associations

Tracks can monitor static routes, default routes received from a DHCP server, and default routes received with a negotiated address. This association is necessary if you are using Network Monitor to control routes.

To associate a track with a route using the CLI, follow these steps:

1. Determine which type of route you are associating with the track. If you are associating a static route, then the route's presence in the routing table is dependent upon the track's state. If the track is in a PASS state, the route remains in the table and is used to forward traffic. When the track changes to a FAIL state, the route is removed. When the track changes back to a PASS state, the route is reinstated. If you are associating a DHCP default route or another default route, the route behaves like a static route, and its presence in the routing table depends on the track's state.
2. Associate the track and the route through the route's configuration.

If you are associating a track and a static route, use the **ip route** command from the Global Configuration mode. It is easiest to associate the track at the time you create the route. The most common variation of the command to use with network monitoring is **ip route <ip address> <subnet mask> <ip address> <administrative distance> track <name>**. You can use other variations in your network configuration. These variations are detailed in the *AOS Command Reference Guide* available online at <https://supportforums.adtran.com/>. The first *<ip address>* parameter specifies the destination to add to the route table (destination is first instance). The second specifies the next hop IP/far end IP address. IP addresses are expressed in dotted decimal notation. The *<subnet mask>* parameter specifies the subnet mask that corresponds to a range of IP addresses (network) or a specific host. Subnet masks can be expressed in dotted decimal notation (for example, **255.255.255.0**) or as a prefix length (for example, **/24**). The *<administrative distance>* parameter specifies an administrative distance associated with a particular router used to determine the best route when multiple routes exist to the same destination. The lower the administrative distance, the higher the priority of the route. By default, the route has an administrative distance of **1**. To specify a route as a backup route, enter a higher administrative distance (range is **1** to **255**). The *<name>* parameter is the name of the track you wish to associate with this route. To specify that the track **BACKUPROUTE** monitor this route with an administrative distance of **100**, you would enter the command as follows:

```
(config)#ip route 10.220.0.0 255.255.0.0 192.22.45.254 100 track BACKUPROUTE
```

If you are associating the track with a DHCP default route, use the **ip address dhcp track <name> <administrative distance>** command from the configuration mode of the interface through which the route travels. Valid interfaces for DHCP routes include Frame Relay, asynchronous transfer mode (ATM), Ethernet, virtual local area network (VLAN), and Point-to-Point Protocol (PPP) interfaces. The same considerations of administrative distance apply to these routes as with other routes.

To add monitoring to a DHCP route through the Ethernet interface, you would enter the command as follows:

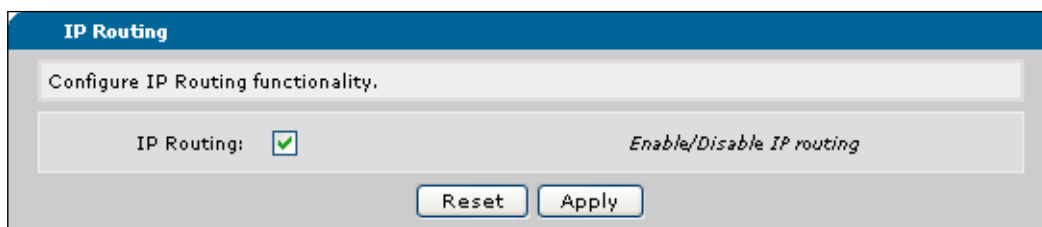
```
(config)#interface eth 0/1
(config-eth 0/1)#ip address dhcp track BACKUPROUTE 100
(config-eth 0/1)#
```

If you are associating the track with another type of default route with a negotiated address, use the **ip address negotiated track <name> <administrative distance>** command from the configuration mode of the interface through which the route travels. Valid interfaces for negotiated routes include PPP and demand routing interfaces. The same considerations of administrative distance apply to these routes as with other routes. To add monitoring to a negotiated route through a PPP interface, you would enter the command as follows:

```
(config)#interface ppp 1
(config-ppp1)#ip address negotiated track BACKUPROUTE 100
(config-ppp1)#
```

To associate a track with a route using the GUI, follow these steps:

1. Connect to the GUI and navigate to **Data > Router/Bridge > Routing**. Ensure IP routing is enabled. If not, enable **IP Routing** using the check box and select **Apply**.



2. To add a track to a **static** route, navigate to **Data > Router/Bridge > Route Table**. Enter the route's configuration information in the appropriate fields (**Destination Address**, **Destination Mask**, **Gateway Address**, and **Administrative Distance**). Select the track name from the drop-down menu, and select **Add**. The route is now monitored by a track, and will be added to the route table.

- To add a track to a **DHCP default** route or a default route received with a **negotiated** address, navigate to **Data > Router/Bridge > IP Interfaces**. Select the appropriate interface from the list.

Name	IP Address	Netmask	Type
eth 0/2	0.0.0.0	255.255.255.255	Ethernet
demand 1	0.0.0.0	255.255.255.255	Demand
eth 0/1	10.200.1.134	255.255.0.0	Ethernet

- In the interface's configuration, scroll to **IP Settings**. Select the appropriate track's name from the drop-down menu and select **Apply**.

- Tracks are now monitoring the routes you have associated with them. As tracks reach a FAIL state, the routes will be removed from the route table. As tracks reach a PASS state, the routes will be reinstated to the route table.

PBR and Network Monitoring

PBR is necessary when using Network Monitor to monitor routes. PBR ensures that probe packets are always forwarded over the right route. If you do not use PBR, the routing table determines where the packets are sent. This can be a problem because you always want probes to send packets over the route you configured them to test, and not along a route determined by a changing routing table. Using PBR allows the probe packets to always use the route they were configured to use, even though network traffic is routed according to the routing table.

To configure PBR for use with network monitoring, you must complete the following steps:

- Create an ACL
- Set the ACL's permissions
- Create a route map entry (one for each probe)
- Match the route map entries to the ACL
- Force the probes out a static, Internet Protocol Control Protocol (IPCP), or DHCP interface
- Apply the route map to a local interface

These steps can be completed using the CLI. To configure PBR for network monitoring using the CLI, follow these steps:

1. Create an ACL using the **ip access-list extended** *<name>* command. This command creates an empty ACL and enters the ACL configuration mode. The *<name>* parameter is the name of the ACL. Enter the command from the Global Configuration mode prompt as follows:

```
(config)#ip access-list extended SECONDARYCONNECTION
Configuring New Extended ACL "SECONDARYCONNECTION"
(config-ext-nacl)#
```

2. Set the ACL's permissions using the **permit** command from the ACL configuration mode. The ACL should be configured to permit packets of the probe's protocol type and packets with the probe's destination address or host name, as well as the destination port if the probe is an HTTP request or TCP connect probe. For ICMP echo probes, the command uses the following syntax: **permit icmp any [host <ip address> | hostname <hostname>]**. For HTTP request or TCP connect probes, the command uses the following syntax: **permit tcp any [host <ip address> | hostname <hostname>] eq <port>**. Enter the command as follows to permit ICMP echo probe packets with the destination of **10.10.10.1**:

```
(config-ext-nacl)#permit icmp any host 10.10.10.1
```

3. Create a route map entry using the **route-map** command. The **route-map** command creates a route map and enters the route map configuration mode. A route map entry should be created for each probe used to test different routes. Syntax of this command for use with PBR and Network Monitor is as follows: **route-map <name> permit 10**. The *<name>* parameter is the name of the route map entry, and **permit 10** indicates that the packets that will match this entry (specified in Step 4) will use the **set** commands (configured in Step 5). Enter the command as follows from the Global Configuration mode prompt:

```
(config)#route-map SECONDARYPROBE1 permit 10
(config-route-map)#
```

4. Match the route map entry to the ACL created for the specific probe using the **match ip address** *<name>* command from the route map's configuration mode. The *<name>* parameter is the ACL name associated with the probe. Enter the command as follows:

```
(config-route-map)#match ip address SECONDARYCONNECTION
```

5. The probes must now be forced out the proper interfaces, using the **set** commands from the route map's configuration mode. To force the probe out a static interface, use the **set ip next-hop <ip address>** and **set interface null 0** commands. The *<ip address>* is the destination address of the route. To force the probe out a PPP or DHCP interface, use the **set interface <interface> null 0** command. The *<interface>* parameter specifies the interface that forces the probe. The **null 0** parameters specify that the packet is dropped if either the next hop is unreachable or the interface is down.



*Only use the **set interface** command for the Ethernet interface if the Ethernet interface learned the default route from DHCP. Otherwise, use the **set ip next-hop** command.*

Enter the commands as follows for a static interface:

```
(config-route-map)#set ip next-hop 10.10.10.2
(config-route-map)#set interface null 0
```

Enter the commands as follows for an PPP interface:

```
(config-route-map)#set interface ppp 1 null 0
```

Enter the commands as follows for a DHCP interface:

```
(config-route-map)#set interface eth 0/1 null 0
```

6. Apply the route map to the local interface by using the **ip local policy route-map** *<name>* command from the Global Configuration mode. The *<name>* parameter here indicates the name of the route map you just configured. The local interface means the interface through which traffic that is generated by the TCP/IP stack in the router travels, not the packets that pass through the router. This means that the route map will only affect performance of the traffic generated by the router itself. To apply the route map to the local interface, enter the command as follows:

```
(config)#ip local policy route-map SECONDARYPROBE1
```

PBR is now configured for network monitoring. In all, the PBR configuration for an ICMP echo probe created to test connectivity to a remote network (address **10.5.1.1**) reached through ATM interface **1.1** would look like this:



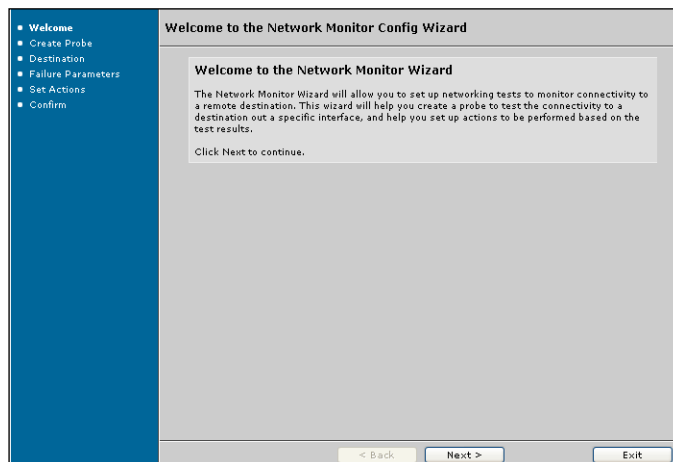
This configuration assumes the ICMP echo probe has a destination specified of 10.5.1.1.

```
(config)#ip access-list extended PROBE1
(config-ext-nacl)#permit icmp any host 10.5.1.1
(config-ext-nacl)#exit
(config)#route-map PROBES 10
(config-route-map)#match ip address PROBE1
(config-route-map)#set interface atm 1.1 null 0
(config-route-map)#exit
(config)#ip local policy route-map PROBES
```

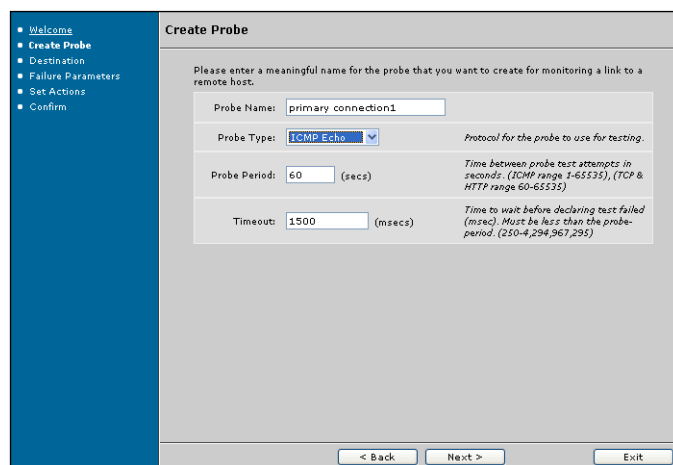
Using the Network Monitor Wizard

The Network Monitor wizard is a guided method of creating probes to test connectivity to a destination out a specific interface. The GUI wizard aids in creating probes and specifying actions to be performed based on the test results. The wizard also automatically creates and associates a track to the probe that is created. To use the Network Monitor wizard to create a probe, follow these steps:

1. Connect to the GUI and navigate to **Data > Network Monitor > Wizard**. You are greeted with the Network Monitor wizard welcome menu. Select **Next** to begin configuring the probe.



2. To create the probe, enter the probe's name in the appropriate field, select the **Probe Type** from the drop-down menu, and enter the **Probe Period** and **Timeout** values. The default values for each probe type are automatically displayed. After the information is completed, select **Next** to continue.



- Enter the probe's **Destination** IP address or host name, and select the **Interface** used to monitor the destination connectivity. Select **Next** when all the information is completed.

Set up a destination for monitor.

Enter an IP address or a hostname of a destination that you want to monitor.

Destination:

Select an interface to use to monitor the destination connectivity.

Interface:

< Back Next > Exit

- Enter the failure mode of the probe. The failure mode is the probe's tolerance for failures and is used to monitor the destination connectivity. For a more detailed description of probe failure modes, refer to *Probe Tolerance on page 9*. When the necessary information is complete, select **Next**.

Set up failure mode.

Select the mode of failure to determine if the link to the destination is failing so that appropriate actions can be taken.

None

Consecutive *The probe will allow the specified number of consecutive test passes and failures before declaring probe state.*

Number of Failures:

Number of Successes:

Rate *The probe will allow the rate of test failures or successes before declaring probe state (X of Y).*

Number of Failures (X):

Number of Successes (X):

Total number of Tests (Y):

< Back Next > Exit

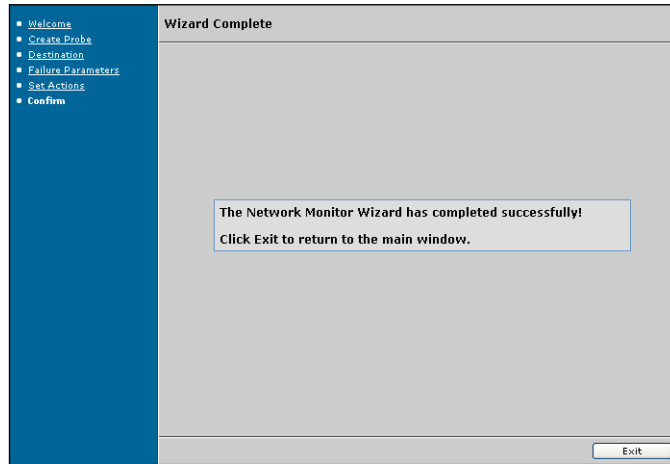
5. Enter the action to be taken if the probe fails. Selections include none or disabling a static route based on the next-hop IP address or interface. This step defines the action to be taken by a created and associated track when the probe fails (the track is created automatically by the wizard). Select **Next** when all information has been entered.

The screenshot shows the 'Set up actions based on probe failure' step of the wizard. On the left is a blue sidebar with a navigation menu containing: Welcome, Create Probe, Destination, Failure Parameters, **Set Actions**, and Confirm. The main panel has the title 'Set up actions based on probe failure.' and a sub-header 'Set up an action to be performed once the probe reports failure (destination is unreachable over the link specified).' There are two radio button options: 'None' (with the note 'Do not perform any action.') and 'Disable static route' (which is selected). Under 'Disable static route', there are two sub-options: 'Next Hop IP:' and 'Next Hop Interface:'. The 'Next Hop IP:' option is selected and has a text input field containing '10 . 200 . 1 . 136'. To the right of this field is a note: 'Create a default static route with the next hop IP or an interface you provide that becomes inactive when the probe fails.' The 'Next Hop Interface:' option is unselected and has a dropdown menu showing 'eth 0/2'. At the bottom of the panel are three buttons: '< Back', 'Next >', and 'Exit'.

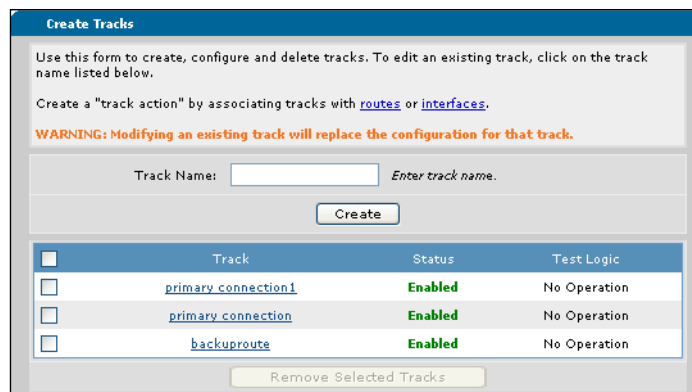
6. Confirm the network monitor settings and select **Finish**.

The screenshot shows the 'Confirm Settings' step of the wizard. On the left is the same blue sidebar with the navigation menu. The main panel has the title 'Confirm Settings' and a sub-header 'Please review the settings that you have configured for network monitoring. You may use the "Back" to change any incorrect settings or "Finish" to complete your setup.' Below this is a list of settings: 'Probe Name: primary connection1', 'Probe Type: ICMP Echo', 'Probe Period: 60', 'Timeout: 1500', 'Destination (Interface): 10.200.1.123 (eth 0/1)', 'Type of Failure: Consecutive Failures', 'Number of Consecutive Failures: 10', 'Number of Consecutive Successes: 5', and 'Action: Over-ride Static Route'. At the bottom of the panel are three buttons: '< Back', 'Finish', and 'Exit'.

7. The wizard is complete. Select **Exit** to exit the wizard.



8. You can now edit the new probe and track created by the wizard by navigating to **Data > Network Monitor > Probes/Tracks**. The newly created track is visible in the track list, and has the same name as the probe created by the wizard.



Network Monitor Common Configurations and Applications

The following sections describe typical network monitoring applications in real-world settings. All configurations are done using the CLI, though most probe and track configurations can be achieved using the GUI as described on [page 18](#) and [page 31](#). The configuration parameters entered in these examples are sample configurations only. You should configure these applications in a manner consistent with the needs of your particular network. CLI prompts have been removed from the configuration examples to provide you with a method of copying and pasting configurations directly from this guide into the CLI. You should not copy these configurations without first making the necessary adjustments to ensure they will function properly in your network.

Configuring Network Monitor for Multi-Homing

In multi-homing applications, Network Monitor is used for link failure detection. When the probe using the primary connection fails, the track monitoring the probe fails, and the primary connection route is removed from the routing table. The backup connection with a higher administrative distance is inserted into the routing table and is used until the primary connection connects again. The primary connection probe becomes successful again, and the track monitoring the probe transitions from a FAIL to a PASS state. The route used by the primary connection is reinserted into the routing table and is used by network traffic.

[Figure 3](#) and [Figure 4](#) display the network configuration for multi-homing.

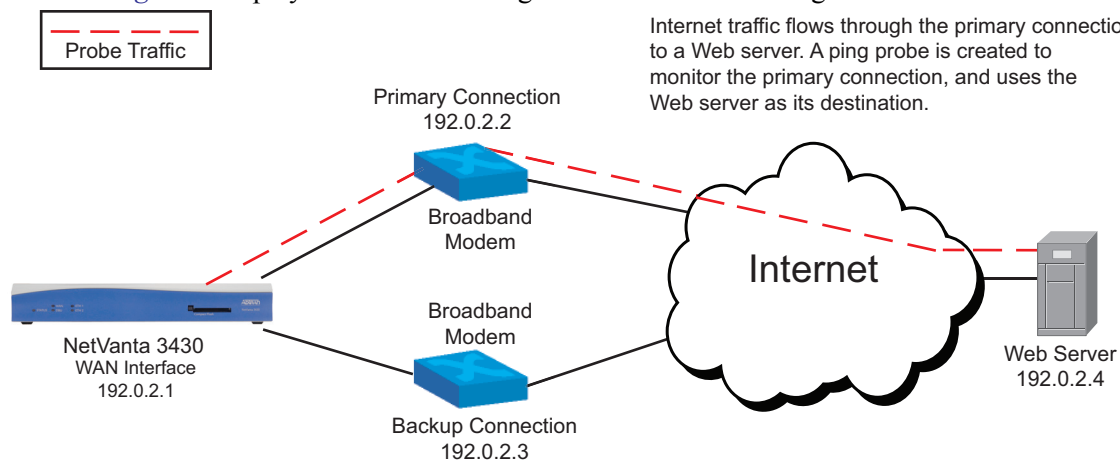


Figure 3. Multi-Homing with Functional Primary Connection

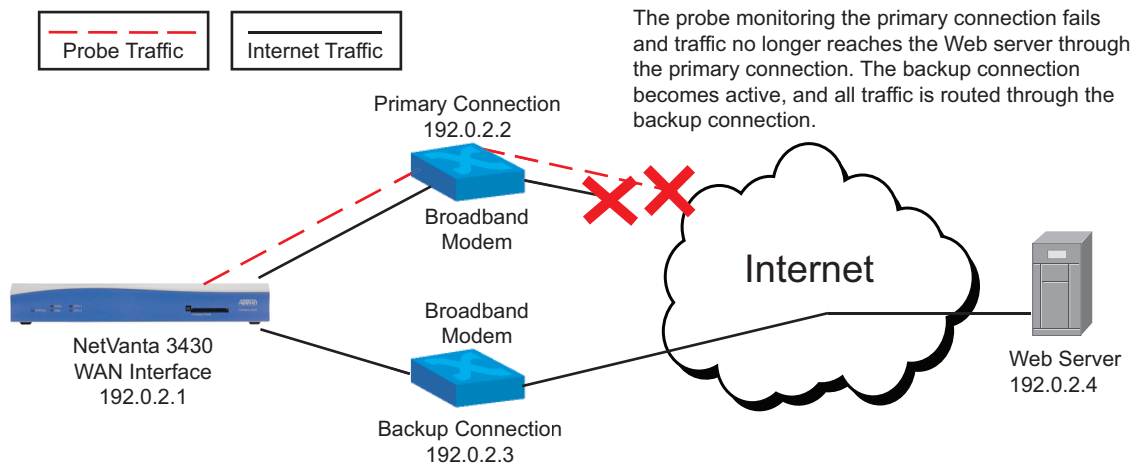


Figure 4. Multi-Homing Using Backup Connection

To configure Network Monitor for this application, you must complete these steps:

- Configure the probe
- Configure the track
- Configure the administrative distance of the secondary link
- Configure PBR
- Associate the track with the primary route

Enter the CLI commands for each step as follows:

1. Configure the probe:

```
probe PING1 icmp-echo
  destination 192.0.2.4
  period 10
  tolerance consecutive fail 3
  no shutdown
```

2. Configure the track:

```
track PING1
  test ip probe PING1
```

3. Configure the administrative distance of the secondary link:

```
ip route 0.0.0.0 0.0.0.0 192.0.2.3 100
```

4. Configure PBR:

```
ip access-list extended PING1
  permit icmp any host 192.0.2.4
!
route-map PROBEPING1 permit 10
  match ip address PING1
  set ip next-hop 192.0.2.2
  set interface null 0
!
```

ip local policy route-map PROBEPING1

5. Associate track with primary route:

```
ip route 0.0.0.0 0.0.0.0 192.0.2.2 track PING1
```

Configuring Network Monitor for Virtual Private Network (VPN) Tunnel Fail-Over

Network Monitor can trigger a backup VPN to become effective in case of primary VPN failure. A probe pings the primary VPN peer, and a track monitors the probe. Backup crypto maps, Internet key exchange (IKE) policies, and remote IDs are created with higher sequence numbers and different peers than the primary VPN. The primary crypto map is associated with the track monitoring the primary VPN. *Figure 5* describes the network configuration for a typical real-world application of VPN connections.

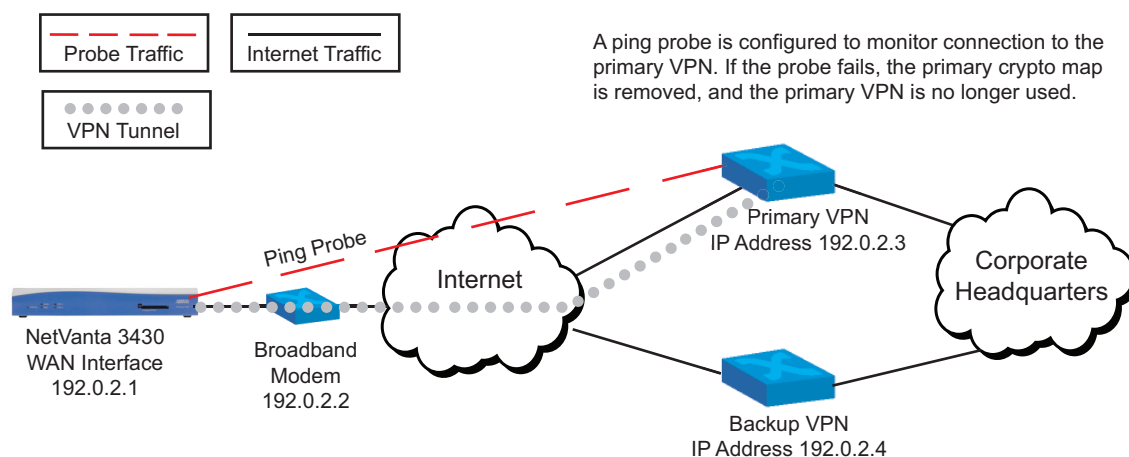


Figure 5. Typical Remote VPN Connection to Corporate Headquarters

In this example, the ADTRAN unit has control of the VPN configuration for both the primary corporate VPN gateway and the backup corporate VPN gateway. In these configurations, only one VPN tunnel is active at a given time. This works as a probe monitors the availability of the primary VPN gateway, and a track monitors that probe. A VPN tunnel is then associated with the primary VPN gateway, which is monitored by the track. The VPN tunnel to the backup VPN gateway is configured with a crypto map that has a higher sequence number than the crypto map associated with the primary VPN. If the probe monitoring the primary VPN fails, and subsequently the track fails, the VPN tunnel to the primary VPN gateway is removed. The primary VPN crypto map is removed when the track fails, and the unit creates the backup VPN tunnel. *Figure 6* illustrates these actions.

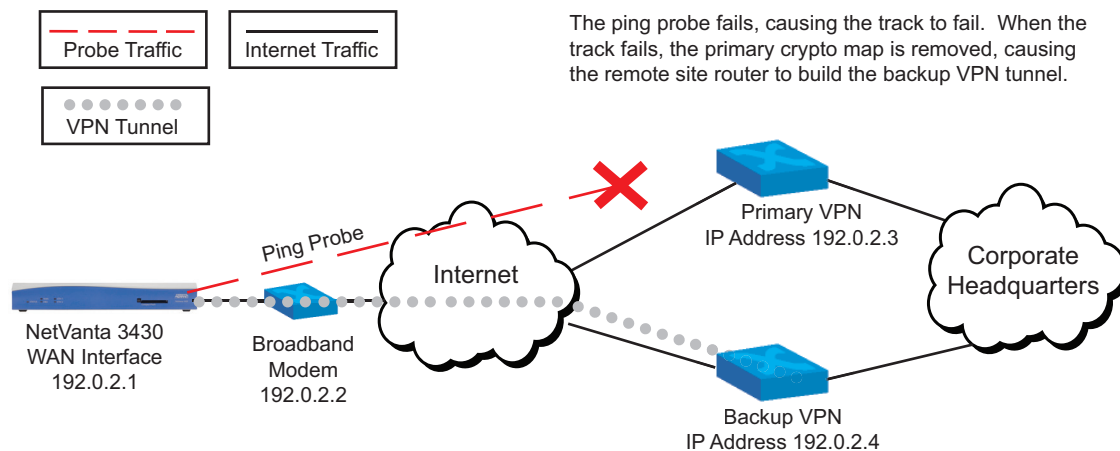


Figure 6. Ping Failure and Backup VPN Tunnel Creation

When the probe returns to a PASS state, the track returns to a PASS state, and the primary VPN tunnel is restored using the primary VPN crypto map. To configure network monitor to create VPN failover, you must complete these steps:

- Configure the primary and secondary VPN tunnels
- Configure the probe
- Configure the track
- Associate the track with the primary crypto map

Enter the CLI commands as follows:

1. Configure primary and secondary VPN tunnels (remote ID, IKE policies, and crypto maps):

```
crypto ike remote-id address 192.0.2.3 preshared-key adtran
crypto ike remote-id address 192.0.2.4 preshared-key adtran
crypto ike policy 10
  initiate aggressive
  peer 192.0.2.3
  attribute 1
  encryption 3des
!
!
crypto ike policy 20
  initiate aggressive
  peer 192.0.2.4
  attribute 1
  encryption 3des
!
!
crypto ipsec transform-set 3des-sha esp-3des esp-sha-hmac
mode tunnel
```

```
!  
ip access-list extended CORPORATE-VPN  
  permit ip 192.0.2.1 any  
!  
crypto map VPN 10 ipsec-ike  
  match address CORPORATE-VPN  
  set peer 192.0.2.3  
  set transform-set 3des-sha  
!  
crypto map VPN 20 ipsec-ike  
  match address CORPORATE-VPN  
  set peer 192.0.2.4  
  set transform-set 3des-sha  
!  
interface eth 0/1  
  crypto map VPN  
!
```

2. Configure the probe:

```
probe VPN-PING icmp-echo  
  destination 192.0.2.3  
  period 5  
  tolerance consecutive fail 3  
  no shutdown  
!
```

3. Configure the track:

```
track VPN-TRACK  
  test if probe VPN-PING  
!
```

4. Associate the track with the primary crypto map:

```
crypto map VPN 10  
  match track VPN-TRACK  
!
```

Configuring Network Monitor to Accommodate a Slow-Starting Primary Connection

Sometimes a site that has a primary Internet connection that is either DHCP Ethernet through a broadband modem, or an ADSL connection, has a primary interface that takes time to negotiate an IP address and default route. If this site also has a backup connection configured to dial an Internet provider if the primary connection fails, the dial-up connection may try to dial before the primary connection has had time to get the address and the route upon initial bootup. This behavior is often undesirable, and can be prevented by using a relative schedule that is tied to a track. The track is then associated with a route. The track becomes active based on the schedule, and then removes or reinstates the route accordingly. Using Network Monitor in this way allows the primary interface to become active without a backup connection dialing unnecessarily. In this example, it takes up to three minutes for the primary connection to become active.

To configure Network Monitor to create a delay, you must do the following:

- Create a schedule that passes only after a delay
- Create a track to monitor the schedule
- Configure a route, and associate it with the track

Enter the CLI commands as follows:

1. Create a schedule that passes only after a delay:
schedule DELAY-AFTER-BOOT
relative start-after 180
2. Create a track to monitor the schedule:
track DELAY
test if schedule DELAY-AFTER-BOOT
3. Configure a route, and associate the route with the track:
ip route 0.0.0.0 0.0.0.0 demand 1 200 track DELAY

Network Monitor and Tcl Scripts

Network Monitor can also be used to execute Tcl scripts. For example, a restaurant has complimentary WiFi for its customers. The WiFi configuration uses virtual access points (VAPs) for customer access, using an encrypted VAP for employees and an open VAP for guests. If the restaurant wants to restrict guest Internet access to business hours, a Tcl script that is activated by a schedule and a track allows them to do so. Two Tcl scripts are created prior to associating them with the schedule and track. The Tcl script, **DisableOpenVAP**, disables the VAP, and the script, **EnableOpenVAP**, enables the VAP. This example assumes that the VAPs for both employees and guests have already been configured.

To configure the schedule and track, and associate them with the correct Tcl scripts, you must do the following:

- Configure a schedule for business hours
- Configure a track for the schedule
- Associate the Tcl scripts with the track

Enter the CLI commands as follows:

1. Configure a schedule for business hours:
schedule BUSINESS
periodic daily 11:00 to 21:00
2. Configure a track for the schedule:
track FREEWIFI
time-schedule BUSINESS fail
no shutdown
3. Associate the Tcl scripts with the track:
run-tcl DisableOpenVAP.tcl track FREEWIFI on-fail
run-tcl EnableOpenVAP.tcl track FREEWIFI on-pass

Network Monitor and ACLs

Network Monitor can also be used to monitor and direct traffic based on an ACL. For example, a business only wants to provide Internet access during business hours. To achieve this, a schedule, **BUSINESS**, is created and configured to be active every weekday from 8 a.m. until 5 p.m. The schedule is then associated with the track, **INTERNET**. An ACL, **wizard-ics**, is created. This ACL allows many-to-one network address translation (NAT) to occur, establishing Internet access, and the **INTERNET** track is applied to the **permit any** statement of the **wizard-ics** ACL. When the **BUSINESS** schedule is active, the **INTERNET** track will be in a PASS state and allow Internet access. When the **BUSINESS** schedule transitions to an inactive state, the **INTERNET** track also transitions to a FAIL state and removes the **permit any** entry in the **wizard-ics** ACL. When the **permit any** entry is removed from the ACL, all communication sessions with the Internet are prevented.

To configure network monitor to track an ACL, you must do the following:

- Set the clock on the AOS device.
- Enable the firewall.
- Create a schedule for business hours.
- Configure a track for the schedule.
- Configure the necessary interfaces.
- Create an ACL for Internet sharing.
- Create a private ACL.
- Configure the ACLs for NAT.
- Configure IP routing.
- Configure the SNTP server.

Enter the CLI commands as follows:

1. Set the clock:

```
clock timezone-6-Central-Time  
clock auto-correct-DST
```

2. Enable the firewall:

```
ip firewall
```

3. Create a schedule for business hours:

```
schedule BUSINESS  
periodic weekday 08:00 to 17:00
```

4. Configure a track for the schedule:

```
track INTERNET  
test if schedule BUSINESS  
no shutdown
```

5. Configure the necessary interfaces:

```
interface eth 0/1  
ip address 192.168.5.43 255.255.255.0  
access-policy Private  
no shutdown
```

```
!  
interface t1 1/1  
  tdm-group 1 timeslots 1-24 speed 64  
  no shutdown  
!  
interface ppp1  
  ip address 65.162.109.202 255.255.255.252  
  access-policy Public  
  no shutdown  
  cross-connect 1 t1 1/1 1 ppp 1
```

6. Create an ACL for Internet sharing:

```
ip access-list standard wizard-ics  
  remark Internet Connect Sharing  
  permit any track INTERNET  
  deny any
```

7. Create a private ACL:

```
ip access-list extended self  
  remark Traffic to NetVanta  
  permit ip any any log
```

8. Configure the ACLs for NAT:

```
ip policy-class Private  
  allow list self self  
  nat source list wizard-ics interface ppp 1 overload  
!  
ip policy-class Public  
  implicit discard
```

9. Configure IP routing:

```
ip route 0.0.0.0 0.0.0.0 ppp 1
```

10. Configure the SNTP server:

```
sntp server pool.ntp.org version 3
```



It is recommended that a Simple Network Time Protocol (SNTP) server is configured with proper time zone settings to ensure the clock on the AOS unit is set properly.



Clock auto-correct-DST is a default setting, and will not be displayed in the output of the show run command.

Network Monitoring and SNMP

Simple Network Management Protocol (SNMP) can be used with network monitoring to report changes in track states, whether from a PASS to a FAIL state, or a FAIL to a PASS state. When used with network monitoring, SNMP can alert you when the track has changed states and makes network management of monitored devices and the condition of the track more easily monitored. Using SNMP with network monitoring is optional. To use SNMP with network monitoring, you must enable SNMP, specify the type of SNMP used, enable network monitoring SNMP traps (either all network monitoring traps or traps for a specific host), and add SNMP trap capabilities in the network monitoring track.

To configure SNMP network monitoring traps using the CLI, follow these steps:



For more detailed information about the configuration of SNMP, refer to [Configuring SNMP in AOS](https://supportforums.adtran.com) available online at <https://supportforums.adtran.com>.

1. Enable the SNMP agent by entering the **ip snmp agent** command from the Global Configuration mode prompt. Enter the command as follows:

```
(config)#ip snmp agent
```

2. Specify that SNMP track traps are enabled by either enabling all network monitor traps (using the **snmp-server enable traps track** command from the Global Configuration mode prompt) or by enabling network monitoring traps only for a specified SNMP destination (using the **snmp-server host <ip address> traps version <1 | 2c | 3> <community> track** command from the Global Configuration mode prompt). Using the **no** form of either command disables the SNMP traps for network monitoring. By default, SNMP traps for network monitoring are disabled. To enable all network monitoring SNMP traps, enter the command as follows:

```
(config)#snmp-server enable traps track
```

To enable network monitoring SNMP traps for a specified host and destination, use the **snmp-server host** command. The *<ip address>* parameter is the destination address that receives the SNMP notifications. Enter IP addresses in the dotted decimal notation (for example, **10.10.10.1**). The **version** parameter of the command specifies the SNMP version and security model used. The *<community>* parameter is the community name used for the SNMP traps. Enter the command as follows:

```
(config)#snmp-server host 10.23.1.181 traps version 1 Public track
```

3. Enable SNMP traps on the tracks you want to monitor using the **snmp trap state-change** command from the track's configuration mode prompt. You will need to enable SNMP traps on each track that you want to monitor. Using the **no** form of this command disables SNMP traps on the track. By default, SNMP traps for tracks are disabled. For example, to enable SNMP traps on the **STANDBY** track, enter the command as follows from the track configuration mode:

```
(config)#track STANDBY
(config-track)#snmp trap state-change
```



For more information about the configuration and use of SNMP, refer to [Configuring SNMP in AOS](https://supportforums.adtran.com) available online at <https://supportforums.adtran.com>.

Tracking and Responding to NNI and UNI Defects

Tracks that monitor for network-to-network interface (NNI) and user-to-network interface (UNI) defects can be defined on the AOS unit and used to dynamically alter the operation of the unit in response to the defect. These tracks allow the unit to change its behavior in response to LOC and RDI conditions, decreased upstream and downstream bandwidth of interfaces or EFM groups, and the line protocol status of the MEN port.



The following sections only provide the configurations required for track operation. For more information on Y.1731 configuration, refer to [Configuring Ethernet OAM Using Y.1731](#), available online at <https://supportforums.adtran.com>.

Loss of Continuity (LOC)

An LOC defect occurs when a MEP no longer receives continuity check message (CCM) frames from a remote MEP. These failures can be caused by both soft failures (misconfigurations) and hard failures (link or device failures). The LOC condition is declared by a MEP when it receives no CCM frames from the remote MEP during an interval equal to 3.5 times the CCM transmission period. The LOC condition is cleared when the MEP receives 3 or more CCM frames from the peer MEP during an interval equal to 3.5 times the CCM transmission period.

Remote Defect Indication (RDI)

Upon encountering a defect condition, such as signal failure or alarm indication signal (AIS), a MEP will send an RDI upstream in the opposite direction of the defect to its peer MEPs to inform them that a failure has occurred downstream. Because RDIs report far-end defect conditions, they can be used for both single-ended fault management and performance monitoring of the far end.

Monitoring UNI and NNI Defect Conditions Using Tracks

UNI and NNI defect conditions can be monitored on MEPs using tracks. The following section gives three examples of tracks used to monitor LOC conditions, upstream and downstream bandwidth, and line protocol states of MEPs, EFM groups, or Ethernet Virtual Connections (EVCs). The state of any created track can be viewed using the **show track** *<name>* command, entered from the Enable mode.

Using a Track to Monitor LOC Conditions on a MEP

Specify a MEP to monitor for LOC conditions by creating a track, using the **track** *<name>* command from Global Configuration mode, and specifying the MEP and LOC conditions in the track's configuration using the **test if ethernet y1731 meg char-string** *<name>* *<level>* *<mep id>* **loc** command from the track's configuration mode. In the following example, the track **EXAMPLETRACK** is created to monitor the LOC conditions of MEP **102**:

```
(config)#track EXAMPLETRACK
```

```
(config-track)#test if ethernet y1731 meg char-string EXAMPLEMEG 7 102 loc
```

Using a Track to Monitor Upstream or Downstream Bandwidth on an EFM Group

Specify an EFM group to monitor for upstream or downstream bandwidth by creating a track, using the **track <name>** command from Global Configuration mode, and specifying the EFM group to monitor for upstream or downstream bandwidth using the **test if efm-group <slot/port> [downspeed <speed> | upstream <speed>]** command from the track's configuration mode. In the following example, the track **EXAMPLETRACK** is created to monitor the upstream bandwidth of EFM group 1/1:

```
(config)#track EXAMPLETRACK
(config-track)#test if efm-group 1/1 upstream 20000
```

Using a Track to Monitor the Line Protocol of an Interface

Specify an interface or EVC to monitor for line protocol state by creating a track, using the **track <name>** command from Global Configuration mode, and specifying the interface or EVC to monitor for line protocol state using the **test if interface <interface> line-protocol** command from the track's configuration mode. In this type of configuration, the track will PASS while the line protocol of the specified interface is UP, and will FAIL when the line protocol state is DOWN. In the following example, the track is configured to monitor the line protocol state of the system control EVC:

```
(config)#track EXAMPLETRACK
(config-track)#test if interface system-control-evc line-protocol
```

Creating Tracks to Respond to NNI and UNI Defects

Not only can tracks be used to monitor multiple NNI and UNI defects, but they can also respond to these defects by disabling the UNI/NNI interface, stopping CCM transmission on the associated MEP, or sending RDIs to the MEPs remote peers. This ability can be used to automatically disable a UNI/NNI interface and stop CCM transmission on the associated MEP in response to defect conditions and automatically enable the interface once the defect has been cleared. Additionally, it can be used to force RDIs to be sent if an NNI or UNI defect is detected. Defect conditions that can be monitored include: LOC and RDI on MEPs, upstream and downstream bandwidth on interfaces, and line protocol status of interfaces.

To use tracks to not only monitor, but also respond to NNI and UNI defects, you will need to create two tracks. One to monitor the administrative status of the UNI (**GIG_3_ADMIN_STATUS**), and one to control the operational status of the MEP (**GIG_3_OPER_STATUS**). The following actions are needed to complete these tasks:

- Create a track to control the administrative state of the UNI/NNI interface associated with the MEP. This track is used to control whether the interface is administratively UP or DOWN based on the presence of a defect. In the following examples, this track is named **GIG_3_ADMIN_STATUS**.
- Specify the multiple objects to be tested by track **GIG_3_ADMIN_STATUS**. Using the Boolean AND function, configure the track so that all objects in the test list must be in the TRUE state for the track to PASS, or at least one object in the FALSE state for the track to FAIL.
- Specify the test conditions of track **GIG_3_ADMIN_STATUS**. The track should be configured to monitor the LOC or RDI conditions of the MEP, the EFM group for a decrease in downstream or upstream bandwidth, and the system control EVC for line protocol state changes. These configurations are similar to those covered in *Monitoring UNI and NNI Defect Conditions Using Tracks* on page 57.

- Associate track **GIG_3_ADMIN_STATUS** with the appropriate UNI interface. In addition, enable the track on the interface.
- Create a track (**GIG_3_OPER_STATUS**) to control the operational status of the MEP. This track is used to control whether the MEP sends CCMs based on the administrative state of the UNI/NNI interface (controlled by the **GIG_3_ADMIN_STATUS** track).
- In track **GIG_3_OPER_STATUS**, specify the interface that will monitor the line protocol state of the UNI/NNI interface. The track will PASS while the line protocol of the specified UNI/NNI interface is UP, and will FAIL when the line protocol state is DOWN.
- Associate track **GIG_3_OPER_STATUS** with the MEP on which CCMs should be disabled if a defect condition is encountered. In addition, specify the RDI transmission behavior when the track is in a FAIL state.

Enter the CLI commands for each step as follows:

1. Create the **GIG_3_OPER_STATUS** track to control the administrative state of the UNI/NNI interface associated with the MEP:

```
(config)#track GIG_3_ADMIN_STATUS
(config-track)#
```

2. Specify that multiple objects will be tested by track **GIG_3_ADMIN_STATUS** using the Boolean AND function:

```
(config-track)#test list and
(config-track-test)#
```

3. Specify the test conditions of track **GIG_3_ADMIN_STATUS**. The track should be configured for several test conditions:

- To monitor the MEP for RDI conditions. This is completed using the **if not** parameter in the configuration so that the track stays in the PASS state while RDI conditions are not present. Conversely, the track will fail when RDI conditions are present.
- To monitor the upstream bandwidth on the EFM group. The track will FAIL if the bandwidth speed falls below the specified kbps.
- To monitor the MEN port (**system-control-ewc**) for line protocol state. The track will PASS while the line protocol of the interface is UP, and will FAIL when the line protocol state is DOWN.

```
(config-track-test)#if not ethernet y1731 meg char-string EXAMPLE_MEG 7 102 rdi
(config-track-test)#if efm-group 1/1 upstream 20000
(config-track-test)#if interface system-control-ewc line-protocol
```

4. Enter the UNI interface's configuration mode (in this example, Gigabit Ethernet 0/3) from the Global Configuration mode and associate the **GIG_3_ADMIN_STATUS** track to the interface. This specifies that the interface should be administratively UP while the track is in a PASS state, and should be administratively DOWN when the track changes to a FAIL state, thus ensuring the interface remains UP under normal conditions and goes DOWN when one of the defects specified in the track is encountered.

```
(config)#interface gigabit-ethernet 0/3
(config-giga-eth 0/3)#no shutdown track GIG_3_ADMIN_STATUS
```

5. Create a track (**GIG_3_OPER_STATUS**) to control the operational status of the MEP. This track is used to control whether the MEP sends CCMs based on the administrative state of the UNI/NNI interface.

```
(config)#track GIG_3_OPER_STATUS
(config-track)#
```

6. In the configuration of track **GIG_3_OPER_STATUS** specify the interface that will monitor the line protocol state of the UNI/NNI interface (in this example, Gigabit Ethernet 0/3). The track will PASS while the line protocol of the UNI interface is UP, and will FAIL when the line protocol state is DOWN. Remember that this interface is UP or DOWN based on the state of track **GIG_3_ADMIN_STATUS** (applied in Step 4).

```
(config-track)#test if interface gigabit-ethernet 0/3 line-protocol
(config-track)#
```

7. From the Global Configuration mode, enter the Y.1731 MEG Configuration mode for the MEG to which the monitored MEP belongs. From the MEG configuration mode, access the Local MEP Configuration mode for the MEP on which the CCM transmissions should be disabled if a defect condition is encountered (in this example, MEP 102).

```
(config)#ethernet y1731 meg char-string EXAMPLE_MEG level 7
(config-y1731-meg EXAMPLE_MEG)#local-mep 102
(config-y1731-mep102)#
```

8. Specify that the MEP (in this example, MEP 102) should transmit CCMs while the track is in the PASS state, and should cease CCM transmission when the track FAILS. In addition, specify that the MEP should force the transmission of RDIs while the track is in the PASS state, and should transmit RDIs according to the rules of Y.1731 when the track is in the FAIL state. Both specifications associate the MEP with the track **GIG_3_OPER_STATUS**, and connect the MEP's CCM and RDI transmissions to the administrative state of the UNI/NNI interface (Gigabit Ethernet 0/3).

```
(config-y1731-mep102)#ccm-enabled track GIG_3_OPER_STATUS
(config-y1731-mep102)#force-rdi track GIG_3_OPER_STATUS
```

Example NNI/UNI Defect Tracking Configuration

In the configuration example below, a track is configured to monitor for the following conditions:

- Line protocol DOWN on system control EVC
- LOC conditions on MEP 102
- RDI conditions on MEP 102
- Upstream speed lower than 20000 kbps on EFM group 1/1

If one or more of the above conditions are experienced, track **GIG_3_ADMIN_STATUS** will FAIL, causing the Gigabit Ethernet 0/3 (UNI port) to go DOWN. The DOWN state of Gigabit Ethernet 0/3 will cause track **GIG_3_OPER_STATUS** to FAIL, stopping CCM transmission on MEP 102. Once the defect condition is cleared, the track will once again PASS, the UNI port will reactivate, and the MEP will begin CCM transmission.

```
track GIG_3_ADMIN_STATUS
test list and
if interface system-control-vec line-protocol
```

```

if not ethernet y1731 mep char-string "EXAMPLE_MEG" 7 102 loc
if not ethernet y1731 mep char-string "EXAMPLE_MEG" 7 102 rdi
if interface efm-group 1/1 upspeed 20000
no shutdown
!
track GIG_3_OPER_STATUS
test if interface gigabit-ethernet 0/3 line-protocol
no shutdown
!
ethernet y1731 mep char-string "EXAMPLE_MEG" level 7
service single-tagged s-tag 3001
remote-mep 201
local-mep 102
direction down
set interface gigabit-ethernet 0/5
priority 6
ccm-enabled track GIG_3_OPER_STATUS
no shutdown
!
evc "VLAN_3001"
s-tag 3001
connect men-port gigabit-ethernet 0/5
no shutdown
!
interface gigabit-eth 0/3
no shutdown track GIG_3_ADMIN_STATUS

```

Viewing CCM and RDI Transmission of MEPs

The `show ethernet y1731 mep local` command can be used to determine if CCM transmission has been disabled by a track failing or if RDI transmission has been forced by a track passing. If CCM transmission has been disabled by a track, AOS will display a **CCM transmission** line indicating the CCM transmission status of the MEP. If RDI transmission has been forced by a track, AOS will display an **RDI transmission** line indicating the RDI transmission status of the MEP. If CCM or RDI transmission is operating normally, no **CCM transmission** or **RDI transmission** line is displayed.

The following shows sample output for the `show ethernet y1731 mep local` command:

show ethernet y1731 mep local

```

MEP 100 in MEGID "MEG ID" is IS and up
  Status                : Running
  CCM transmission      : Disabled by track ADMIN_STATUS
  RDI transmission      : Forced by track ADMIN_STATUS
  Interface              : Gigabit-Ethernet 0/5
  Direction              : Down
  MEG Level              : 7
  Service                : S-TAG (1000)
                        Receive    Transmit
  Total CCM Frames      : 17005    17006

```

Total 1DM Frames	: 17006	17005
Total DMM Frames	: 85030	85025
Total DMR Frames	: 85025	85030
Total LBM Frames	: 0	0
Total LBR Frames	: 0	0
Total SLM Frames	: 85030	85025
Total SLR Frames	: 85025	85030
Total LTM Frames	: 0	0
Total LTR Frames	: 0	0
Active Continuous 1DM Sessions	: 1	
Active On-demand 1DM Sessions	: 0	
Active Continuous DMM Sessions	: 5	
Active On-demand DMM Sessions	: 0	
Active Continuous SLM Sessions	: 5	
Active On-demand SLM Sessions	: 0	

	Alarm status
CCM Loss of Continuity Alarm	: Clear
CCM Unexpected Level Alarm	: Clear
CCM Mismatch Alarm	: Clear
CCM Unexpected MEP Alarm	: Clear
CCM Unexpected Period Alarm	: Clear
CCM RDI Alarm	: Clear

Determining if the EVC Has Been Disabled by a Track Failure

The **show evc** *<name>* command can be used to determine if the EVC has been disabled by a track failure. If the EVC has been disabled by a track failure, the **Admin State** field will indicate which track disabled the EVC. The optional *<name>* parameter specifies the name of the EVC for which information should be displayed. If no name is specified, information for all EVCs is displayed.

The following shows example output for the **show evc** *<name>* command:

show EVC EXAMPLE_EVC

All EVC Tags Available in MEN

EVC EXAMPLE_EVC

S-TAG	: 1000
Admin State	: Disabled by track TRK_LOC
EVC Status	: Not Running - Disabled
MEN-Port	: gigabit-ethernet 0/3
CE-VLAN Preservation	: Enabled

Network Monitor Command Summary

The following tables summarize the commands used in different CLI configurations of network monitoring. Command summaries include the commands for probe configurations, track configurations, schedule configurations, and PBR for network monitoring configurations. For more details about any of these commands or other commands that may be used in conjunction with network monitoring, refer to the *AOS Command Reference Guide* available online at <https://supportforums.adtran.com>.

Table 7. NM Probe Configuration Command Summary

Access Prompt	Command	Description
(config)#	probe <name> [http-request icmp-echo tcp-connect]	Creates a probe, indicates the probe's name and type, and enters the probe's configuration mode.
(config-probe-<name>)#	vrf <name>	Specifies the VRF within which the probe operates. If no VRF is specified, the probe operates within the default (unnamed) VRF.
(config-probe-<name>)#	destination <hostname ip address> port <number>	Specifies the location to which the probe sends test packets. Ports are specified for TCP connect and HTTP request probes. HTTP request probes use port 80 by default.
(config-probe-<name>)#	period <value>	Specifies the time (in seconds) between probe tests. Valid range for ICMP echo probes is 1 to 65535 . Valid range for TCP connect and HTTP request probes is 60 to 65535 . By default, all probes have a default period of 60 seconds.
(config-probe-<name>)#	source-address <ip address>	Specifies the source IP address for probe test packets. By default, the IP address of the originating interface is used.
(config-probe-<name>)#	source-port <port>	Specifies the source port for TCP connect and HTTP request probe packets. Valid range is 0 to 65535 . By default, the port is set to 0 , which allows the probe to dynamically select the port number.

Table 7. NM Probe Configuration Command Summary (*Continued*)

Access Prompt	Command	Description
(config-probe- <i><name></i>)#	timeout <i><value></i>	Specifies the time (in milliseconds) in which a returned packet must be received before the test is considered failed. Range is 1 to 90000 milliseconds. Default value for ICMP echo probes is 1500 milliseconds, for TCP connect probes is 10000 milliseconds, and for HTTP request probes is 10000 milliseconds.
(config-probe- <i><name></i>)#	tolerance consecutive [pass <i><number></i> fail <i><number></i>]	Specifies the probe will change states after a certain number of probe tests have failed or passed. Fail range is 1 to 255 and pass range is 1 to 255 . By default, consecutive tolerance is set to 1 .
(config-probe- <i><name></i>)#	tolerance rate [fail <i><number></i> pass <i><number></i>] of <i><set size></i>	Specifies the probe will change states after a certain number of probe tests have passed or failed in a specified number of tests. Fail range is 1 to 254 , pass range is 1 to 254 , set size range is 1 to 255 . Default rate tolerance is $1+(\text{set size}) - (\text{value of other state})$.
(config-probe- <i><name></i>)#	tolerance rate fail <i><number></i> pass <i><number></i> of <i><set size></i>	Specifies the probe will change to FAIL state or PASS state when the respective number of tests pass or fail in a specified number of tests. Fail range is 1 to 254 , pass range is 1 to 254 , set size range is 1 to 255 . Default rate tolerance is $1+(\text{set size}) - (\text{value of other state})$.
(config-probe- <i><name></i>)#	no shutdown	Activates the probe.

Table 8. Optional Probe Configuration Command Summary

Access Prompt	Command	Description
(config-probe- <i><name></i>)#	size <i><payload length></i>	Specifies the length of an ICMP echo probe test packet. Range is 0 to 1462 bytes. By default, packet length is 0 bytes.

Table 8. Optional Probe Configuration Command Summary (Continued)

Access Prompt	Command	Description
(config-probe- <i><name></i>)#	data <i><pattern></i>	Specifies the data pattern of an ICMP echo probe test packet. Pattern can be any hexadecimal data pattern. Default pattern is a standard ping packet pattern, beginning with 0x00, incrementing by one for the length of the packet.
(config-probe- <i><name></i>)#	type [get head raw]	Specifies the type of HTTP request probe. HTTP request probes are GET requests by default.
(config-probe- <i><name></i>)#	raw-string	Specifies the text in the data portion of an HTTP RAW request. Not configured by default.
(config-probe- <i><name></i>)#	expect status <i><minimum></i> <i><maximum></i>	Specifies that the HTTP request probe will expect a specific status code in response to an HTTP request message. If another status is returned, the probe fails. Minimum status code range is 0 to 999 . Maximum status code range is 0 to 999 . Maximum status codes are optional, and only serve to create a valid range of acceptable status codes. Not configured by default.
(config-probe- <i><name></i>)#	expect regex <i><expression></i>	Specifies the HTTP request probe will expect a regular expression inside the contents of the HTTP response message. If the regular response does not match anything, the probe fails. Not configured by default.
(config-probe- <i><name></i>)#	absolute-path <i><name></i>	Specifies the path of the HTTP request probe to the server. By default, the server's root path is the forward slash symbol (/).

Table 9. NM Track and Single Object Test Configuration Command Summary

Access Prompt	Command	Description
(config)#	track <i><name></i>	Creates and names a new track. Also, enters the track configuration mode.

Table 9. NM Track and Single Object Test Configuration Command Summary (Continued)

Access Prompt	Command	Description
(config-track)#	test if [not] ethernet y1731 meg [char-string <name> icc-umc <name>] <level> <mep id> [loc rdi]	Specifies a Y.1731 MEP is the object to be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description.
(config-track)#	test if [not] interface <interface> [ipv6-routing ip-routing line-protocol downspeed <speed> upspeed <speed>]	Specifies an interface is the object to be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description.
(config-track)#	test if [not] probe <name>	Specifies a probe is the object to be tested. The not parameter negates the results of the test.
(config-track)#	test if [not] schedule <name>	Specifies a schedule is the object to be tested. The not parameter negates the results of the test.
(config-track)#	test if [not] [ip ipv6] ffe [<ingress interface>] entries less-than <number>	Specifies that the number of RapidRoute flow entries will be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description.
(config-track)#	test if [not] voice user <extension> registered	Specifies that the SIP registration status of the specified voice user will be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description.
(config-track)#	dampening-interval [<value> fail <value> pass <value>]	Specifies the amount of time to wait before allowing a new probe status to trigger a track action. Value range is 1 to 4294967295 seconds. Default delay is 1 second.
(config-track)#	log-changes	Enables the logging of track state changes to the terminal screen, the event history, and to a syslog server if logging forwarding has been configured.
(config-track)#	time-schedule <name> [pass fail]	Associates a track and schedule, so that the track is active based on the schedule. Keywords pass and fail specify what state the track is in when the schedule is inactive.

Table 10. NM Test List Command Summary

Access Prompt	Command	Description
(config-track)#	test list [and or]	Specifies that the track use Boolean logic to monitor multiple objects and enters the Boolean Track Test List Configuration mode.
(config-track-test)#	if [not] ethernet y1731 meg [char-string <name> icc-umc <name>] <level> <mep id> [loc rdi]	Specifies a Y.1731 MEP is the object to be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description.
(config-track-test)#	if [not] interface <interface> [downspeed <speed> ip-routing ipv6-routing line-protocol upspeed <speed>]	Specifies an interface is the object to be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description.
(config-track-test)#	if [not] probe <name>	Specifies a probe is the object to be tested. The not parameter negates the results of the test.
(config-track-test)#	if [not] schedule <name>	Specifies a schedule is the object to be tested. The not parameter negates the results of the test.
(config-track-test)#	if [not] [ip ipv6] ffe [<ingress interface>] entries less-than <number>	Specifies that the number of RapidRoute flow entries will be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description.
(config-track-test)#	test if [not] voice user <extension> registered	Specifies that the SIP registration status of the specified voice user will be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description.

Table 11. NM Test List Weighted Command Summary

Access Prompt	Command	Description
(config-track)#	test list weighted	Specifies the track uses weighted logic to monitor multiple objects and enters the Weighted Track Test List Configuration mode.

Table 11. NM Test List Weighted Command Summary (Continued)

Access Prompt	Command	Description
(config-track-test)#	if [not] ethernet y1731 meg [char-string <name> icc-umc <name>] <level> <mep id> [loc rdi] weight <value>	Specifies a Y.1731 MEP is the object to be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description. Weight range is 1 to 65535 .
(config-track-test)#	if [not] interface <interface> [downspeed <speed> ip-routing ipv6-routing line-protocol upspeed <speed>] weight <value>	Specifies an interface is the object to be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description. Weight range is 1 to 65535 .
(config-track-test)#	if [not] probe <name> weight <value>	Specifies a probe is the object to be tested. The not parameter negates the results of the test. Weight range is 1 to 65535 .
(config-track-test)#	if [not] schedule <name> weight <value>	Specifies a schedule is the object to be tested. The not parameter negates the results of the test. Weight range is 1 to 65535 .
(config-track-test)#	if [not] [ip ipv6] ffe [<ingress interface>] entries less-than <number> weight <value>	Specifies that the number of RapidRoute flow entries will be tested. The not parameter negates the results of the test. Refer to Track Test Commands Complete Command Syntax on page 26 for syntax description. Weight range is 1 to 65535 .
(config-track-test)#	threshold <number>	Specifies a baseline weight for track state transitions. Range is 1 to 4294967295 .
(config-track-test)#	threshold pass <number> fail <number>	Specifies the number which, if reached or exceeded, will change the state of the track to PASS or FAIL. Range is 1 to 4294967295 .

Table 12. Schedule Command Summary

Access Prompt	Command	Description
(config)#	schedule <name>	Creates a schedule and enters the schedule configuration mode.

Table 12. Schedule Command Summary (Continued)

Access Prompt	Command	Description
(config-schedule- <i><name></i>)#	absolute start <i><time></i> <i><day></i> <i><month></i> <i><year></i> end <i><time></i> <i><day></i> <i><month></i> <i><year></i>	Specifies the schedule has an absolute start and end time. Time is expressed in hh:mm 24-hour format. The day of the week is a number, ranging from 1 to 31 . The year is expressed in the yyyy format.
(config-schedule- <i><name></i>)#	relative start-after <i><delay></i>	Specifies the schedule is a delayed schedule. Delay range is 1 to 65535 seconds.
(config-schedule- <i><name></i>)#	periodic <i><day></i> <i><time></i> to <i><time></i>	Specifies the schedule reoccurs on specific days with specified start and end times. Days are expressed in name format, and can be a single day or up to seven days of the week. Time is expressed in hh:mm 24-hour format.
(config-schedule- <i><name></i>)#	periodic <i><day></i> <i><time></i> for <i><time></i>	Specifies the schedule reoccurs on specific days at a specified time, and runs for a specified length of time. Days are expressed in name format, and can be a single day or up to seven days of the week. Time is expressed in hh:mm 24-hour format.
(config-schedule- <i><name></i>)#	periodic [daily weekday weekend] <i><time></i> [to for] <i><time></i>	Specifies the schedule reoccurs either daily, during the week, or on weekends.

Table 13. Commands to Associate Tracks and Routes

Access Prompt	Command	Description
(config)#	ip route <ip address> <subnet mask> <ip address> <administrative distance> track <name>	Creates a static route and specifies the track associated with the route. Default administrative distance is 1.
(config)#	interface <interface>	Specifies an interface to configure for associating tracks with DHCP routes or default routes with negotiated addresses.
(config-<interface>)#	ip address dhcp track <name> <administrative distance>	Associates a track with an interface that uses DHCP default routes.
(config-<interface>)#	ip address negotiated track <name> <administrative distance>	Associates a track with an interface that uses default routes with a negotiated address.

Table 14. Network Monitor and PBR Command Summary

Access Prompt	Command	Description
(config)#	ip access-list extended <name>	Creates an empty ACL for PBR configuration, and enters the extended ACL configuration mode.
(config-ext-nacl)#	permit [icmp tcp] any [host <ip address> hostname <hostname>] [eq <port>]	Specifies the permit parameters for the ACL. If using TCP, the destination port must be specified.
(config)#	route-map <name> permit 10	Creates a route map and enters the route map configuration mode.
(config-route-map)#	match ip address <name>	Specifies the ACL used by the route map for IP address matching.
(config-route-map)#	set ip next-hop <ip address>	Specifies the path of the route map for static routes.
(config-route-map)#	set interface null 0	Specifies that a static interface force the route.
(config-route-map)#	set interface <interface> null 0	Specifies the interface that forces the route for PPP or DHCP routes.
(config-route-map)#	ip local policy route-map <name>	Applies the route map to the local interface.

Table 15. SNMP Configuration Command Summary

Prompt	Command	Description
(config)#	ip snmp agent	Enables the SNMP agent used for SNMP traps.
(config)#	snmp-server enable traps track	Enables all SNMP network monitoring traps. By default, network monitoring SNMP traps are disabled.
(config)#	snmp-server host <ip address> traps version <1 2c 3> <community> track	Enables SNMP network monitoring traps for a specified destination. The <ip address> parameter is the destination address that receives SNMP notifications. Enter IP addresses in dotted decimal notation (for example, 10.10.10.1). The <1 2c 3> parameter specifies the security model version used by SNMP. The <community> parameter is the community name used by SNMP. By default, network monitoring SNMP traps are disabled.
(config-track-Primary)#	snmp trap state-change	Enables SNMP traps for the track. SNMP traps must be enabled on the track for network monitoring SNMP traps to be used. By default, SNMP traps are disabled on the track.

Troubleshooting

The probe, track, and schedule parameters of Network Monitor can be viewed using either the CLI or GUI. The CLI **show** and **debug** commands, and the GUI **View Statistics** options, aid in troubleshooting as they allow a quick picture of Network Monitor component configurations. The following sections describe the CLI troubleshooting commands and how to access the GUI statistics.

CLI Troubleshooting

The following table provides a quick look at the Network Monitor CLI troubleshooting commands.

Table 16. Network Monitor Troubleshooting Commands


Access Prompt	Command	Description
#	show track [<name> [realtime]	Displays track object configuration and statistics.
#	show schedule	Displays schedule configuration information.
#	show probe [<name> [realtime statistics history]	Displays probe configuration and statistics.

Table 16. Network Monitor Troubleshooting Commands (Continued)

Access Prompt	Command	Description
#	debug probe [<name>]	Displays probe event messages or activates debug messages associated with activities performed by the named probe object.
#	debug track [<name>]	Activates debug messages associated with activities performed by the named track object.

Show Commands

Show commands are issued from the Enable mode prompt, and display configuration information and statistics for tracks, probes, and schedules. The <name> parameter for tracks and probes displays only the information about a specific track or probe, rather than all configured tracks or probes. Additional parameters are available with the track and probe **show** commands. The optional **realtime** keyword displays full-screen output in real time.

 **NOTE** Using the *realtime* argument for this command can adversely affect the performance of your unit.

The **history** keyword displays the history of all measured probe statistics. The **statistics** keyword displays measured probe statistics.

The following is sample output from the **show probe** command:

```
>enable
#show probe probe_A
Current State: PASS Admin. Status: DOWN
  Type: ICMP Echo Period: 30 sec Timeout: 500 msec
  Source: 10.200.1.134 Destination: 10.200.1.123
  Data size: 0
  Tracked by: track_1
  Tests Run: 121 Failed: 0
  Tolerance: 0 consecutive failures currently, 10 needed to FAIL
  Time in current state: 25 days 2 hours, 34 minutes, 32 seconds
```

The following is sample output from the **show schedule** command:

```
>enable
#show schedule
Schedule entry: DELAY-AFTER-BOOT (active)
Schedule entry: DELAY (inactive)
```


The following is sample output from the **show track** command:

```
>enable
#show track track_1
Current State: PASS (Admin: UP)
  Testing:
    probe probe_A (PASS)
  Dampening Interval: 30 seconds
  Time in current state: 25 days 2 hours, 34 minutes, 32 seconds
  Track State Changes: 3
  Tracking:
    ip route 0.0.0.0. 0.0.0.0 10.200.1.136
```

Debug Commands

Debug commands are issued from the Enable mode prompt, and display information associated with activities performed by both probes and tracks. The *<name>* parameter in both debug commands displays only the information about a specific probe or track. Without this parameter, all track or probe configurations are displayed.



Turning on a large amount of debug information can adversely affect the performance of your unit.

Enter the command as follows to enable debug messages for probe **primary connection1**:

```
>enable
#debug probe primary connection1
14:44:16: NETMON.PROBE primary connection1: Reply from 10.200.1.123: bytes=28 time=1ms (passed)
```

Contained in the sample output are the following:

- The time of the debug message (**14:44:16**) (**hh:mm:ss** format)
- The type of object (**NETMON.PROBE**)
- The name of the object (**primary connection1**)
- The object's destination (**Reply from 10.200.1.123**)
- The data size of the probe packet (**bytes=28**)
- The packet travel time (**1ms**)
- The state of the object (**passed**)

Enter the command as follows to enable debug messages for the track **BACKUPCONNECTION**:

>enable

#debug track BACKUPCONNECTION

```
15:39:15: NETMON.TRACK BACKUPCONNECTION: Static Route 10.10.20.0 255.255.255.0 10.10.10.3 disabled
```

```
2008.09.03 15:39:15 NETMON.TRACK BACKUPCONNECTION changed from pass to fail.
```

```
15:39:32: NETMON.TRACK BACKUPCONNECTION: Static Route 10.10.20.0 255.255.255.0 10.10.10.3 enabled
```

```
2008.09.03 15:39:32 NETMON.TRACK BACKUPCONNECTION changed from fail to pass.
```

The sample output for the **debug track** command is similar to that of the **debug probe** command. The sample output contains the following information:

- The time of track test (**15:39:15** and **15:39:32**) (**hh:mm:ss**)
- The type of object (**NETMON.TRACK**)
- The name of the object (**BACKUPCONNECTION**)
- The routes monitored by the track (**10.10.20.0 255.255.255.0 10.10.10.3**)
- The state of the monitored route (**disabled** or **enabled**)
- The state change of the track (**changed from pass to fail** or **changed from fail to pass**)
- The date and time of the track state change (**2008.09.03 15:39:15**) (**year.month.day hh:mm:ss** format)

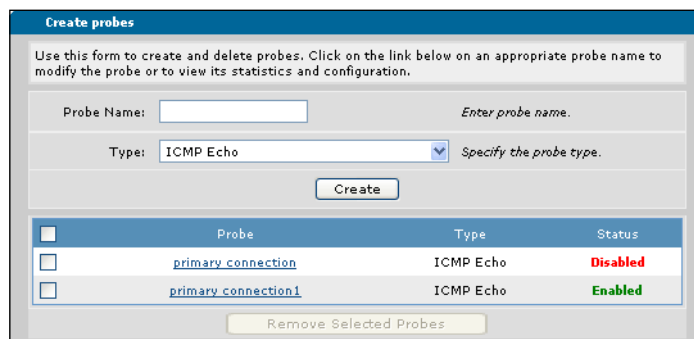
GUI Troubleshooting

Probe and track configuration and statistical information is also available through the GUI. To access this information, connect to the GUI and follow the steps outlined in the following sections.

Viewing Probe Statistics

To view probe statistical and configuration information, follow these steps:

1. Navigate to **Data > Network Monitoring > Probes / Tracks**. Select the probe name from the list.



2. Scroll to the bottom of the new menu, and review the statistics for the selected probe.

Status for "probe primary connection1"

Admin Status	Enabled
Current Test State	PASS
Time in Current State	2 days, 23 hours, 19 minutes, 10 seconds
Total Tests	4550
Tests Failed	21

Refresh in 2 seconds...

Clear Statistics

Viewing Track Statistics

To view probe statistical and configuration information, follow these steps:

1. Navigate to **Data > Network Monitoring > Probes/Tracks**. Select the track name from the list.

Create Tracks

Use this form to create, configure and delete tracks. To edit an existing track, click on the track name listed below.

Create a "track action" by associating tracks with [routes](#) or [interfaces](#).

WARNING: Modifying an existing track will replace the configuration for that track.

Track Name: Enter track name.

Create

<input type="checkbox"/>	Track	Status	Test Logic
<input type="checkbox"/>	primary_connection1	Enabled	No Operation
<input type="checkbox"/>	primary_connection	Enabled	No Operation
<input type="checkbox"/>	backuproute	Enabled	No Operation

Remove Selected Tracks

2. Scroll to the bottom of the new menu, and review the statistics for the selected track.

Status for Track primary connection1

Current State	PASS
Admin Status	Enabled
Number of Track Changes	0
Time in Current State	3 days, 0 hours, 28 minutes, 14 seconds

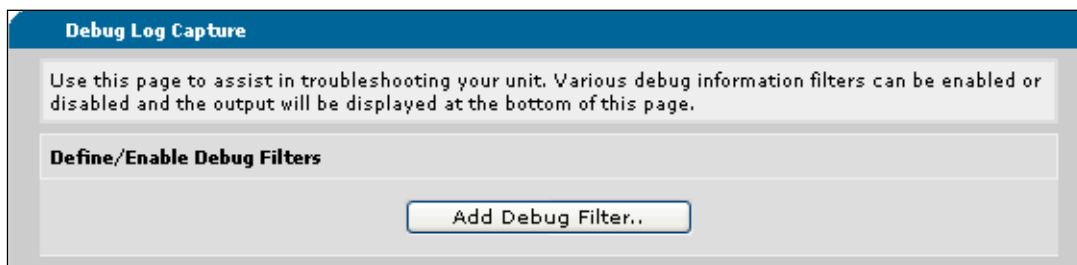
Refresh in 1 seconds...

Clear Statistics

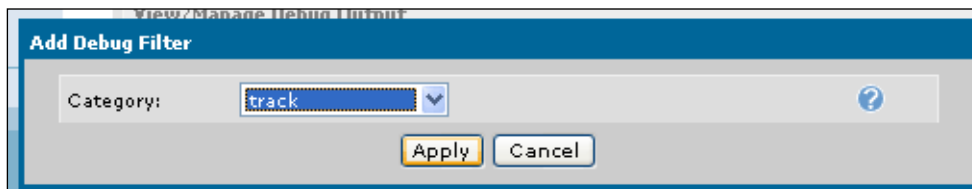
Debugging Probes and Tracks

Debugging features can also be activated for both probes and tracks using the GUI. To use the GUI for these features, follow these steps:

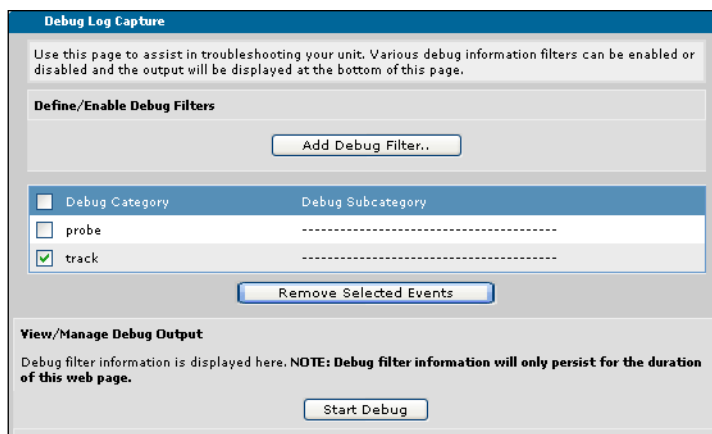
1. Navigate to **Utilities > Debug Unit**. Select **Add Debug Filter**.



2. Select **track** or **probe** from the drop-down menu and select **Apply**.



3. The **track** or **probe** keyword appears in the list of debug filters. Select the check box next to **track** or **probe**, and select **Start Debug**.



4. Debug messaging is now activated.

NOTE *Turning on a large amount of debug information can adversely affect the performance of your unit.*

Additional Documentation

Additional information about Network Monitor, its applications, and related network configurations is available in the following documents:

Table 17. Additional Documentation

Document Title	Document Type
<i>AOS Command Reference Guide</i>	Command reference guide
<i>Configuring Redundant VPN Tunnel Fail-Over in AOS</i>	Common application guide
<i>WAN Failover Using Network Monitoring</i>	Common application guide
<i>Configuring WAN Fail-Over in AOS</i>	Technical note
<i>Tcl Scripting in AOS</i>	Configuration guide
<i>Generic Mail Agent</i>	Quick configuration guide
<i>Policy Based Routing</i>	Configuration guide
<i>Configuring Network Quality Monitoring (NQM) in AOS</i>	Configuration guide
<i>Configuring Multi-VRF in AOS</i>	Configuration guide
<i>Configuring Ethernet OAM using Y.1731</i>	Configuration guide
<i>Configuring IPv6 in AOS</i>	Configuration guide
<i>RapidRoute Service Features in AOS</i>	Configuration guide

These documents are available online at <https://supportforums.adtran.com>.