Introduction
Introduction
Introduction

Copyright © High-Availability.com. All rights reserved.
Sold under licence by High-Availability.Com Ltd.

Issue 2.2, last updated 10/01/2004 14:30:00

All rights reserved. This product and related documentation are protected by copyright and distributed under licensing restricting their use, copying, distribution and de-compilation. No part of this product or related documentation may be reproduced in any form or by any means without prior written authorisation of High-Availability.Com Ltd. While every precaution has been taken in the preparation of this book, High-Availability.Com Ltd assumes no responsibility for errors or omissions.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. High-Availability.Com Ltd MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCTS(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

RSF-1 is a registered trademark of High-Availability.com.
Sun, SPARC, Solaris, SunOS, Solstice and Java are trademarks of Sun Microsystems, Inc.
AIX and PowerPC are trademarks of International Business Machines Corp.
Linux is a registered trademark of Linus Torvalds.
RedHat is a registered trademark, and RPM is a trademark of RedHat Software, Inc.
VERITAS, Volume Manager and File System are trademarks of VERITAS Software Corp.
FireWall-1 is a trademark of Check Point Software Technologies Ltd.
Oracle is a registered trademark of Oracle Corp.
Sybase is a trademark of Sybase, Inc.
Remedy, Action Request System and AR System are trademarks of Remedy Corp, Mountain View, California, which are registered or pending in certain jurisdictions.
All other products are trademarks of their respective companies.

Typographic Conventions

The majority of this manual is set in Verdana 10pt.

Code examples, program names and file pathnames are set in courier new at 10pt.
Introduction

Table of Contents

1 Introduction ........................................................................................................................................... 7
2 Theory of Operation ............................................................................................................................... 9
  2.1 Network Connectivity ...................................................................................................................... 9
    2.1.1 ARP (Address Resolution Protocol) ....................................................................................... 9
    2.1.2 Ping - Internet Control Message Protocol (ICMP) ............................................................ 10
    2.1.3 UDP Time Protocols ............................................................................................................. 10
    2.1.4 Other Considerations ............................................................................................................ 10
3 Configuration .......................................................................................................................................... 12
4 Detail Configuration .............................................................................................................................. 13
  4.1 Global Settings ............................................................................................................................... 13
    4.1.1 LOG_LEVEL .......................................................................................................................... 13
    4.1.2 DEBUG_LEVEL ..................................................................................................................... 13
    4.1.3 DEFAULT_METHOD ............................................................................................................. 13
    4.1.4 PROCESS_PRIORITY .......................................................................................................... 13
    4.1.5 REPORT <method> ............................................................................................................. 13
    4.1.6 RESMON PIPE <pipe> ........................................................................................................ 14
  4.2 Global or Group options ................................................................................................................ 14
    4.2.1 POLL_TIME <seconds> ...................................................................................................... 14
    4.2.2 ACT_DELAY <seconds> ....................................................................................................... 14
    4.2.3 PKT_TIMEOUT <seconds> .................................................................................................. 14
    4.2.4 MAX_PKTLOSS <losses> ...................................................................................................... 14
    4.2.5 OPTION <name> <value> .................................................................................................... 14
      4.2.5.1 OPTION ARP-SRC-IP or ARP-SOURCE-IP <IP-ADDRESS> ........................................ 14
      4.2.5.2 OPTION TX-PER-POLL <NUMBER> ........................................................................ 14
      4.2.5.3 OPTION PROCESS-PRIORITY <NUMBER> ................................................................. 15
  4.3 Group Options ............................................................................................................................... 15
    4.3.1 INTERFACE <INTERFACE-NAME> .................................................................................... 15
    4.3.2 MACHINE <ADDRESS>@[ROUTER] [METHOD <TYPE>] ................................................ 15
  4.4 Value Formats ............................................................................................................................... 16
  4.5 NetMon status reporting using http ............................................................................................... 16
    4.5.1 WebFront Options .................................................................................................................. 17
      4.5.1.1 OPTION WEBFRONT-PORT <INTEGER> ...................................................................... 17
      4.5.1.2 OPTION WEBFRONT-ENABLE <TRUE|FALSE|ENABLE|DISABLE> .................................. 17
      4.5.1.3 OPTION WEBFRONT-SERVER-NAME <STRING> ........................................................... 17
      4.5.1.4 OPTION WEBFRONT-DEFAULT-REFRESH <INTEGER> ................................................ 17
      4.5.1.5 OPTION WEBFRONT-AS-USER <STRING> .................................................................. 17
      4.5.1.6 OPTION WEBFRONT-AS-UID <INTEGER> .................................................................. 17
      4.5.1.7 OPTION WEBFRONT-AS-GID <INTEGER> ..................................................................... 17
    4.5.2 Notes on Using WebFront ......................................................................................................... 17
    4.5.3 Security Advice When Using WebFront .................................................................................. 18
5 Interface Failover with ResMon .......................................................................................................... 19
  5.1 Example of NetMon and ResMon interaction ................................................................................ 19
6 Licensing and Running NetMon .......................................................................................................... 21
  6.1 Starting and Monitoring NetMon .................................................................................................. 21
  6.2 Command Line Options .................................................................................................................. 21
7 Important usage Notes ......................................................................................................................... 23
  7.1 Actions on Detecting Change ......................................................................................................... 23
8 Pre-Installation Worksheet ................................................................................................................ 24
Appendix A. NetMon Worked Example ................................................................................................. 25
Appendix B. Remote Connectivity (ISP Failover) ................................................................................. 27
Appendix C. Syslog Messages ............................................................................................................... 28
Appendix D. Installation ......................................................................................................................... 37
Introduction
1 Introduction

This document describes the installation, configuration and operation of NetMon and the associated product ResMon, both for Unix-based systems.

NetMon is a program that runs on a host and continually monitors all network interfaces on that host for reliable connectivity to the outside world. To do this NetMon must have one or more external network enabled devices to be able to contact, for each interface being monitored (although external network devices can be shared amongst interfaces being monitored). Network devices are typically (but not exclusively) other hosts and routers.

NetMon has no kernel module components and runs completely in user space. However, as NetMon requires access to the low level network drivers it is run with root privileges. NetMon requires this low level access because it is completely responsible for the creation of all outbound packets to ensure no operating system defaults (for example source IP address, system default routes) are included in those packets and the decisions on which interface to send them out on (which could potentially result in unreliable or false connectivity information).

NetMon can be configured to operate at different speeds, with detection and switchover of traffic occurring in less than five seconds. In some installations it is safe to configure NetMon to respond in sub-second times, however, this should be used with caution as it increases the likelihood of a false failure detection on a machine with diminishing resources (i.e. heavily loaded CPU etc). See section 4.2.5.2 (OPTION TX-PER-POLL <NUMBER>) for a discussion on the metrics of polling intervals. NetMon detects and reports network failures.

However, should NetMon detect a failure, action taken is typically done by an accompanying program called the Resource Allocation Manager (ResMon). ResMon is supplied with NetMon and is installed and configured as part of the NetMon installation procedure – details of which are also covered in this document.

ResMon can be considered as the decision based component of NetMon. Whereas NetMon spends its time monitoring and reporting the availability (or not) of interfaces, ResMon spends its time analysing this information on a whole and performing actions based on that analysis.

In a simple configuration (which we define in more detail further on in this manual), ResMon is responsible for a single virtual IP address made available on one of the host machines interfaces. ResMon knows the state of all interfaces being monitored from the information it receives from NetMon. Initially, ResMon will assign a virtual IP to the first available interface. However, should NetMon report that an interface has failed ResMon will then select the next available interface (again using the information being supplied by NetMon) and move the virtual IP over to it. Should the failed interface become available again, ResMon will add it to the list of available interfaces but take no action.

Finally, for this action to be useful, the host usually has a service associated with the virtual IP. So, for example, if a WEB server is running and accessed through this IP,
Introduction

any interface failure will only cause a momentary loss of service as ResMon reassigns the address to the next available interface.
2 Theory of Operation

2.1 Network Connectivity

NetMon verifies network connectivity by contacting a specified remote node or list of nodes on attached networks. It maintains contact to these nodes using a single or combination of available configurable protocols (methods). The methods available are now described.

2.1.1 ARP (Address Resolution Protocol)

Using ARP, NetMon sends a broadcast request intended to resolve the node's IP address into an Ethernet MAC address\(^3\) – NetMon is not actually interested in the contents of the reply, rather just the fact that it has received a reply, thus implying that the remote node is reachable, and thus secondly inferring that the interface the response was sent on is up and running.

The above paragraph introduces an important concept in understanding how NetMon operates. With all packet types it sends out, NetMon is not interested in the contents of the response received, rather it is only concerned that it received a response at all. NetMon maintains an internal table that records all packets sent out. These packets are removed from this table in one of two ways, 1) a valid response packet is received and 2) a timeout occurs indicating that no response was received.

NetMon also understands that a group of nodes can be monitored through a single interface, and thus will only mark an interface as failed when all nodes in the group fail to respond. The general rule of thumb here is that the more nodes you monitor through each interface, the better the indication of the true state of that interface; monitoring just one node through a particular interface makes the state of the interface dependent on the availability of a single node, and is thus intrinsically more likely to mark an interface as down when it is also conceivable that the remote node is at fault. There are situations where this may be acceptable, for instance when a single node is monitored via two separate interfaces, where a single failure can be configured to cause an acceptable re-routing of packets.

Finally, nodes being monitored will learn your own address from the ARP request, thereby implying that if two interfaces share the same IP address, the source address for the ARP request should be changed to avoid erroneous entries in the remote ARP cache (or the remote host continually tries to update it's cache to reflect the ever changing MAC to IP address mapping). NetMon gets round this issue by allowing you to specify the source address (see section 4.2.5.1) for the outbound ARP packet, either globally or on a per interface basis; so for instance using the reserved 192.168.*.* style addresses ensures that machines being monitored do not have problematical ARP cache entries (it may also be set to the subnet zero address should the device being polled support this style of addressing). The setting of the ARP source address MUST be done when polling Cisco routers. The use of source addressing in ARP polling is considered a best practice.

For monitoring purposes, ARP is considered the most preferable option as it is lightweight protocol (and therefore most efficient for monitoring purposes), and must be implemented by any device supporting the TCP/IP protocol suite. However, one drawback with ARP is that the packets are not routable, and therefore monitoring is limited to the local network only. In order to provide a mechanism that allows

---

\(^3\) A node can be host, a router, a bridge or anything that correctly implements the ARP protocol

\(^4\) Note, this method uses broadcast frames which will be received by all nodes on the local network
monitoring of nodes on remote networks, two other protocols (**ping** and **time**) are supported. These are now described.

**2.1.2 Ping - Internet Control Message Protocol (ICMP).**
The ‘ping’ method sends ICMP echo requests to the monitored host, and waits for an echo response. Since this operates at the OS stack level, there is no guarantee that the remote host is running applications properly (the same is true of ARP). Any host that is running an IP stack should respond to these requests without any extra configuration. However, there is an added advantage that ping (and the other UDP protocols described below) have over ARP in that they can be routed. This means that you can optionally tell NetMon to use a gateway when sending out individual packets. Furthermore, it is possible to specify different gateways for different ping requests within the same group of nodes being monitored, thereby allowing as diverse a range of monitored hosts as is required. Note it is possible that a network filter or firewall may be configured to drop ping requests.

**2.1.3 UDP Time Protocols**
NetMon also supports two time based protocols, **time** on port 27 and the **Network Time Protocol** (NTP) on port 123, both of which also support routing through a gateway. It may be necessary to configure or enable these protocols on a node before it will respond; refer to vendor specific documentation for details. NTP software (**xntp**) can also be downloaded from the Internet, go to the NTP home site [http://www.ntp.org/](http://www.ntp.org/) for details.

Although ARP and ICMP are the preferred methods for monitoring, the UDP time protocols are supported for the following reasons:

- The ARP protocol only works on locally connected hosts and therefore cannot be used where a monitored node is on a remote network.
- Many people firewall out ICMP ping requests as a result of the widely publicised ‘ping of death’ denial of service attack, but leave time and xntp open.

**2.1.4 Other Considerations**
As stated before, the monitored nodes should preferably be grouped, typically by the interface through which they are reached. If all nodes in a group fail to respond, NetMon marks that group **DOWN** and reports **fail interface-name** to ResMon. If one or more nodes begin to respond again, NetMon then reports **resume interface-name** to ResMon. It is ResMon that makes the decision on what to do when these events occur.

Should connectivity to several groups of nodes be lost simultaneously, NetMon can be configured to ignore the failures and take no action (say for example where a complete network outage is experienced).

If you have multiple interfaces on the same network, you can perform interface failover and yet ignore network failures by defining a different group for each interface. This means that netmon will send out packets on **each** interface, and by inference test each interface, rather than the standard system behaviour of routing all packets destined for the same network through the first available interface. Without this facility, NetMon could not differentiate between interface and network failure.

---

5 It should be noted that the so called ping of death exploit (packets with more than the allowed 65507 octets of data), is actually not just restricted to ICMP packets but TCP and UDP as well, so the true fix is to apply vendor supplied patches specifically aimed at fixing this type of attack.

6 Via a Unix named pipe
Theory of Operation

NetMon logs its status and all state changes, including script output, to Unix standard output along with certain messages to the syslog facility. The amount of information reported can be changed via command line options or within the configuration file.
3 Configuration

Part of the standard installation is an example configuration file (located as /opt/HAC/NetMon/etc/config.example) that implements the example configuration shown in Figure 1.

Figure 1 example configuration supplied with NetMon

Copy the example config file to /opt/HAC/NetMon/etc/config and edit it to reflect interfaces installed along with machines and their IP addresses to monitor – the example config file is well documented and should be edited in conjunction with the section on Detail Configuration.

Once NetMon is configured using the example file, start the server process:

```
/bin/sh netmonrc start
```

To test that everything is working as expected, use the Unix tail command to view the output of NetMon:

```
tail -f /opt/HAC/NetMon/etc/pipe
```

This should produce output of the form unknown, resume or fail <interface name>. Once you are happy with this stage, stop the tail command using <CTRL>-C and move onto the next section, detailed configuration, to further refine your setup.
4 Detail Configuration

This section documents all the options that are available to configure NetMon. Options described here are declared in /opt/HAC/NetMon/etc/config, which is loaded only at NetMon start up.

Options can have one of two scopes in the configuration file, global or local. Global options are declared before any interface group definitions, and likewise local options are specified within an interface group definition. Local options override options set globally (for the scope of the interface group). Most entries have a default value, documented below.

Note that for interface monitoring, multiple physical network interfaces must be on the same network. (Consult your OS documentation or vendor for assistance with this.)

4.1 Global Settings

The following settings are specified globally only and apply to general NetMon behaviour irrespective of any interface group. Please note that all these settings have default values given in the individual description.

4.1.1 LOG_LEVEL
Controls verbosity of logging up-to a maximum setting of 9. Higher levels cause more log messages to appear with extra detail, but will cause more activity on the system. Levels above 1 will cause a lot of output, most of it periodic information sent to the logs. See section Syslog Messages for a detailed description of the message format and levels. The default value is 1.

4.1.2 DEBUG_LEVEL
Controls the amount of debugging information logged, up-to a maximum setting of 9. Logging is done on stdout and is set in the netmonrc file to /opt/HAC/NetMon/etc/netmon.log. The default debug level is 1.

4.1.3 DEFAULT_METHOD
Default protocol used when connecting to remote nodes. Set this to either arp, time, ping or ntp. The protocol can also be specified set for each node being monitored. The default method is time.

4.1.4 PROCESS_PRIORITY
This has the same effect as the nice command, allowing the process runtime priority to be raised or lowered. For more information see vendor specific documentation on the nice(1) command. The default priority is unset.

4.1.5 REPORT <method>
Instructs NetMon how to report the status of the interfaces being monitored. This can either be continually to ResMon via a named pipe, or via scripts on state change. A setting of REPORT RESMON configures continual status updates to ResMon via the named pipe /opt/HAC/NetMon/etc/pipe; REPORT SCRIPTS will send notifications of state change to the script located at /opt/HAC/NetMon/etc/interface.%d. The ‘%d’ will be replaced by a number identifying the group being monitored. The script is called with three arguments, the first is UP or DOWN, depending upon the state change, the second is the group name and the third is the interface name – currently the group and interface arguments are the same.
The limited functionality provided by the interface script can be handled with much more flexibility by using the Resource Allocation Manager – please see section 5 for more details. Logging output from the scripts is sent to stdout and should be redirected to a suitable log file (by default, netmonrc sends it to /opt/HAC/NetMon/log/netmon.log). Both methods can be used at the same time. The default setting is REPORT RESMON.

4.1.6 RESMON PIPE <pipe>
Sets the name of the pipe to use in communication with ResMon, default is /opt/HAC/NetMon/etc/pipe.

4.2 Global or Group options
The following options may be set on a global or group-by-group basis:

4.2.1 POLL_TIME <seconds>
Interval in seconds at which NetMon connects to remote nodes in interface groups. NetMon contacts one node in each group on every poll. The default poll time interval is 4 seconds.

4.2.2 ACT_DELAY <seconds>
Time to leave for state to settle after a change (up/down) before doing the associated action. This should be at least bigger than the POLL_TIME and PKT_TIMEOUT settings. Ideally, a value should be chosen so that at least two packets are considered lost before any action is taken to stop erroneous failures being reported.

4.2.3 PKT_TIMEOUT <seconds>
How long till a packet is considered lost. PKT_TIMEOUT should be smaller than POLL_TIME, otherwise it is possible that transmissions may be missed whilst waiting for a host to timeout.

4.2.4 MAX_PKTLOSS <losses>
How many consecutive packets sent to a host are lost before the host is considered down.

4.2.5 OPTION <name> <value>
A number of options are available which are specific to the polling methods currently being deployed (on a host basis). These options are:

4.2.5.1 OPTION ARP-SRC-IP or ARP-SOURCE-IP <IP-ADDRESS>
Sets the source IP address for sending out ARP requests. The parameter is either a dotted-quad IP address, or a valid hostname. When not specified the ARP source address defaults to using the interface's primary IP address, setting this option means that the interfaces primary address is unused in outbound packets, see the section on Theory of Operation: ARP (Address Resolution Protocol) for an explanation of why you may want to do this. The option is applicable on a group basis only.

4.2.5.2 OPTION TX-PER-POLL <NUMBER>
Set the number of hosts in each group to send to at every poll interval. The default is 1. This option can be applied at a group or global level. This setting is used to control overall the amount of packets being sent at every monitoring interval. To show the significance of this setting consider the following example:

1. We are monitoring 4 interfaces.
2. For each interface being monitored there are 4 remote nodes packets are being set to.
3. The poll interval is .5 of a second.

This means that for every poll interval, if we monitored all nodes, NetMon would be sending out $4 \times 4 = 16$ packets or 32 packets a second, or 1920 packets a minute. Although the packets sent are small, overall this is a not so insignificant amount. Factor into this that NetMon also has to process the (hopefully) returned packets, then every minute NetMon is processing close to 4000 packets.

In order to reduce this (unnecessary) load, by default, NetMon only sends out a single packet per group per poll cycle. In this example the packet and work load is reduced by a factor of 4. NetMon works its way round the hosts to monitor for each given group in a round robin fashion, thereby spreading the workload on the monitored nodes.

Finally, NetMon understands that all the nodes of a monitored interface have to fail (and therefore by implication be tested) before it is an indication of true interface failure. With a \texttt{TX-PER-POLL} setting of 1 in our example it takes 4 poll cycles (with associated IP timeout) before an interface that has failed will be marked as down. To reduce the poll cycles required, and therefore the speed of interface failure detection, increase \texttt{TX-PER-POLL}.

For our example to detect after a single poll cycle the setting would be 4 (anything higher defaults down to the maximum nodes being monitored through any single interface). However, please note that this is a trade off between network traffic and host system load vs. an increase in response time of 1.5 seconds. To put this in perspective a system administrator should judge the benefits of the extra 1.5 seconds to users of the service against the increase in traffic and work. In the above example the bidirectional packet difference (per day) is 1382400 vs. 5529600 = 4147200 packets.

\textbf{4.2.5.3 OPTION PROCESS-PRIORITY <NUMBER>}
Sets the process priority NetMon sets itself to on start-up. This is a global option only.

\textbf{4.3 Group Options}
An interface grouping of monitored machine is introduced by the INTERFACE keyword followed by a list of machines to monitor using the MACHINE keyword. The format of these options are:

\textbf{4.3.1 INTERFACE <INTERFACE-NAME>}
Starts a group with the name of the interface \texttt{<INTERFACE-NAME>}, all machine entries following this keyword are bound for monitoring purposes to this interface and monitoring packets will be sent out via this interface. The group is completed by either reaching the end of the config file or upon encountering another INTERFACE keyword, whereby a new interface group is started.

\textbf{4.3.2 MACHINE <ADDRESS>@[ROUTER] [METHOD <TYPE>]}
Following an INTERFACE definition comes a list of machines to monitor so NetMon can determine that packets can be send and received through it, and by inference, that the interface is healthy. The simplest form of MACHINE entry is one where a single locally attached machine is declared for monitoring thus:

\begin{verbatim}
MACHINE 10.0.5.7
\end{verbatim}
Typically this machine will be on the locally connected network (a must for ARP monitoring). If you have a global default method set (see DEFAULT_METHOD) then monitoring of individual hosts can be modified to use a different method using the optional METHOD keyword to specify a different type. Types recognised are ARP, PING, TIME and NTP. Finally where a host exists on a non-local network you can specify a router to use using the @ROUTER syntax. Note that you cannot use a router for ARP as it is a non-routable protocol. For completeness the following example shows a fully configured MACHINE entry:

```
MACHINE 10.0.5.7@10.1.2.1 METHOD PING
```

### 4.4 Value Formats

The timing parameters applied to the options POLL_TIME, PKT_TIMEOUT and ACT_DELAY are in seconds, in increments of 1 microsecond. Practically, any value too small is realistically un-schedulable in even the most modern of operating systems, and times less than one tenth of a second are not recommended.

When setting the timings up, POLL TIME should be bigger than PKT_TIMEOUT, otherwise you may miss poll intervals while a host is timing out. ACT_DELAY should be bigger than one poll interval to avoid unnecessary failover due to temporary network failures.

A truth-value is considered true if it is one of either true, enable or 1, and false if it is one of false, disable or 0.

Finally, NetMon reads its configuration file once on start-up. Should any errors be found (for example a truth-value setting of yes) then an error message identifying the offending line and value is printed out and NetMon exits.

### 4.5 NetMon status reporting using http

The system can report its status via the WebFront system, which is controlled from the configuration file. The default is to enable this on port 40000 and listen for connections (for example, use the URL: http://<hostname>:40000/). Each page returned will refresh itself every few seconds, to provide up-to-date information.

The front index page shows the general status of the system, showing each group or interface being monitored, and the state of this (either UP or DOWN). The interface name links to the status page for that interface, showing more detailed information. An example of the front page is:

```
Overall Status:

Current network status is up

<table>
<thead>
<tr>
<th>Interface</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>lan0</td>
<td>up</td>
</tr>
<tr>
<td>lan1</td>
<td>down</td>
</tr>
</tbody>
</table>
```

Generated at Thu Jul 24 12:08:29 2003 by netmon on WebFront HPUX
Powered by High-Availability.Com's NetMon
Detail Configuration

This front page shows the current network status followed by the status of individual interfaces. Clicking on the hyperlink for each interface shown results in a summary page being displayed, for example:

**Status for lan0:**

<table>
<thead>
<tr>
<th>Machine</th>
<th>Status</th>
<th>Polls Sent</th>
<th>Polls Lost (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>217.40.20.250</td>
<td>up</td>
<td>355</td>
<td>3 (0 %)</td>
</tr>
</tbody>
</table>

Generated at Thu Jul 24 12:18:07 2003 by netmon on WebFront HPUX
Powered by [High-Availability.Com's NetMon](http://www.high-availability.com)

### 4.5.1 WebFront Options

The **OPTION** directive in the configuration file is also used to configure webfront. The available options are as follows:

#### 4.5.1.1 OPTION WEBFRONT-PORT <INTEGER>

Sets the port number for WebFront to listen for incoming http connection requests. The default is 40000.

#### 4.5.1.2 OPTION WEBFRONT-ENABLE <TRUE|FALSE|ENABLE|DISABLE>

Control whether WebFront is started; enable or true will cause WebFront to start. The default is true.

#### 4.5.1.3 OPTION WEBFRONT-SERVER-NAME <STRING>

Sets the server name as reported on the front html page. The default is to display the string none.

#### 4.5.1.4 OPTION WEBFRONT-DEFAULT-REFRESH <INTEGER>

Configure the number of seconds a WebFront client is advised to wait between page refreshes. Default is 5 seconds.

#### 4.5.1.5 OPTION WEBFRONT-AS-USER <STRING>

The WebFront sub-processes should run with the effective uid of <STRING>. This option will only succeed if the user specified exists and NetMon has permission on the host system to change uid’s using the setuid() system call and gid’s using the setgid() system call.

#### 4.5.1.6 OPTION WEBFRONT-AS-UID <INTEGER>

The WebFront sub-processes should run with the uid number of <INTEGER>. This option will only succeed if the uid specified exists and NetMon has permission on the host system to change uid’s using the setuid() system call.

#### 4.5.1.7 OPTION WEBFRONT-AS-GID <INTEGER>

The WebFront sub-processes should run with the gid number of <INTEGER>. This option will only succeed if the gid specified exists and NetMon has permission on the host system to change gid’s using the setgid() system call.

### 4.5.2 Notes on Using WebFront

If WebFront is enabled when NetMon starts, it will attempt to run the request process every second until a successful start is achieved. If another process is already bound
to the configured port simply stopping this process will enable WebFront. Alternatively use the WEBFRONT-PORT option to configure WebFront on a different port.

WebFront requests are only handled every second so they do not interfere with normal polling functions. Note also that a new process is spawned for each request, so there may be several NetMon processes running at any one time.

4.5.3 Security Advice When Using WebFront

Please be aware that since NetMon is running with root privileges, any security compromises with WebFront will gain the same level of access. It is advisable to ensure that it is firewalled from any un-trusted networks.

The WEBFRONT-AS-USER, and WEBFRONT-AS-<UID|GID> options can help by attempting to drop some of the privileges which the sub-processes inherit from their parent. This is not guaranteed to lose all the privileges as the process may still be connected to a console and any sockets that have been left open.
5 Interface Failover with ResMon

NetMon is primarily designed to monitor interfaces and detect any failures on those interfaces. Should a failure occur, NetMon is able to perform at least simple reporting via the interface scripts located in /opt/HAC/NetMon/etc/interface.%d.

The task of actually performing complex logic operations when interface state changes occur falls to the Resource Allocation Manager (ResMon). Using the information supplied by NetMon, ResMon continually monitors the state of all interfaces on a machine and is configured to perform actions should an interface failure occur. Because ResMon is highly configurable and is able to base the decisions it makes by considering the overall state of a machine, it is by inference considerably more flexible and better suited to the job of performing interface failover than any stand-alone scripts could ever be. It is for these reasons that High-Availability.com recommends using ResMon in conjunction with NetMon to perform interface failover.

The following example is included to help demonstrate the interaction between NetMon and ResMon.

5.1 Example of NetMon and ResMon interaction

If we consider the following ResMon configuration file:

```
# The named pipe for NetMon/ResMon communications.
PIPE /opt/HAC/NetMon/etc/pipe

# The interfaces NetMon is monitoring.
RESOURCE le0
RESOURCE qe0
RESOURCE qe1

# Declare a group of interfaces
INTERFACE_GROUP public {
    REQUIRES le0 OR qe0 OR qe1
}

# The IP address logically assigned to one of the interfaces.
IPADDRESS 192.168.200.101 {
    REQUIRES public
    INTERFACE_GROUP public
    ALIASNO 1
}
```

The first line declares the named pipe used to communicate between NetMon and ResMon – this should be the same as the RESMON PIPE entry of the NetMon configuration file.

The next set of entries starting RESOURCE declares the interfaces being monitored by NetMon. These are then grouped together under a single logical resource named public using the INTERFACES_GROUP syntax. This grouping also has some logic built into it; the logical resource public only being considered available when one or more of the member resources (le0, qe0 and qe1) are themselves available. If we then follow this logic to the IPADDRESS grouping declaration, we can surmise that the IP address 192.168.200.101 only becomes available when at least one of the interfaces in the group public is available.
In this configuration the IPADDRESS grouping is where ResMon really starts to pay its way. Once the sequence of events occur that go to make IPADDRESS grouping active, ResMon then aliases the IP address to one of the available underlying resources. In this example let us say le0. Because we have declared the IPADDRESS grouping to have an ALIASNO of 1, ResMon will allocate the IP address 192.168.200.101 to le0:1 (likewise if we were using qe0, then the IP address would be allocated to qe0:1 and so forth).

Now consider the sequence of events that occur should we lose connectivity to the outside world on interface le0. Initially the failure will be spotted by NetMon and reported to ResMon as an interface down (here we are assuming that qe0 and qe1 are still up and functioning). ResMon now considers this as a state change in resource public, and as such any dependencies on public will need to be un-configured and then re-configured. In this manner, the IP address is removed from le0:1 and reassigned to the next available interface (say qe0:1).

One last point to note, as resource public is considered up when one or all of its required resources are available, should a resource that was previously unavailable (as in the case of le0 above) become available again, ResMon will not attempt any action, but simply increase the number of logical interfaces available to the resource public. Again, using the above example, it is only when a failure is detected on qe0 that the IP address will be reassigned (and not necessarily to le0, it could easily be qe1 as the next considered available interface).

Finally, the example configuration file shown above can be found in the ResMon as /opt/HAC/ResMon/etc/example.config.
6 Licensing and Running NetMon

NetMon is a licensed product and as such, before it can be run, needs either a temporary or permanent licence installed. Licences are obtained from High-Availability.com using the procedure described at http://www.high-availability.com/Secure/lic_process.php.

Once a licence has been obtained it is stored in the file /opt/HAC/NetMon/etc/license. You can run the command netmon with the -v switch to check the validity of the licence installed.

Once installed and licensed NetMon is started with the netmonrc script, installed by default in the system initialisation files directory; its location is system depended, refer to the init(1) manual page for details.

6.1 Starting and Monitoring NetMon

The netmonrc script used to control the starting and stopping of NetMon by the system takes a number of arguments. It can also be modified to alter the options NetMon is started with (see the next section for details). The following table describes the arguments taken.

<table>
<thead>
<tr>
<th>Option Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>Manual</td>
</tr>
<tr>
<td>stop</td>
<td></td>
</tr>
<tr>
<td>restart</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td></td>
</tr>
<tr>
<td>fullstatus</td>
<td></td>
</tr>
</tbody>
</table>

6.2 Command Line Options

The following table describes the optional command line arguments NetMon recognises. These options can be set using the shell variable OPTIONS in the netmonrc start-up file.

<table>
<thead>
<tr>
<th>Option Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c</td>
<td>NetMon parses the standard configuration file (or the one specified with the -f flag), and then exit with a return code representing whether the configuration file verified OK.</td>
</tr>
<tr>
<td>-d &lt;debug level&gt;</td>
<td>Has the same effect as DEBUG_LEVEL in the configuration file. Setting on the command line overrides any configuration file setting.</td>
</tr>
<tr>
<td>-f &lt;filename&gt;</td>
<td>Specifies the location of the configuration file to read.</td>
</tr>
<tr>
<td>-h</td>
<td>Prints help text, including command line and build information.</td>
</tr>
<tr>
<td>-h &lt;sub help&gt;</td>
<td>Used to get more specific information on options.</td>
</tr>
<tr>
<td>-h options</td>
<td>Provides a list of all the items that the OPTION keyword can take.</td>
</tr>
</tbody>
</table>
Licensing and Running NetMon

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Causes NetMon to report its monitoring status to syslog every hour.</td>
</tr>
<tr>
<td>-l &lt;log level&gt;</td>
<td>Has the same effect as LOG_LEVEL in the configuration file. Setting on the command line overrides any configuration file setting.</td>
</tr>
<tr>
<td>-s</td>
<td>Causes the status and fullstatus reporting option of netmonrc to be sent to syslog as well as stdout.</td>
</tr>
<tr>
<td>-v</td>
<td>Displays version and exits. This will also print whether the product is licensed or not, an exit code of 0 only if licensed.</td>
</tr>
</tbody>
</table>
Important usage Notes

7 Important usage Notes

There are a number of important points that should be considered when configuring and running NetMon. These are:

- Do not monitor a virtual IP address. Also, if the address can run on the local machine, you will not be monitoring a true network connection. In addition, for UDP packets, the source address of the response packet and will not match that of the original destination address NetMon used, causing it to discard valid responses (this is an operating system behaviour where the fixed address is used to identify response packets rather than the virtual address the packet was originally sent to).
- Monitor at least two nodes on each attached network. The more hosts monitored, the more reliable and representative failure detection becomes.
- Ensure that the hosts being monitored are reliable and continually connected. Use other important servers, routers or network nodes.
- Check that the hosts being monitored respond to the protocols that you are using. You can check UDP traffic with utilities such as netcat\(^7\) to probe the relevant port(s).
- When using interfaces (i.e. specifying an interface), devices being monitored must be on the same logical network.
- The problems to be avoided when polling a remote host using ARP have been covered in detail in section 2.1.1 ARP (Address Resolution Protocol). To summarise, when an ARP request is sent out, it is tagged with the IP address of the source (the primary IP address of the interface). Remote machines will learn who the sender is from the packet, which can cause problems when you are failing over using the primary interface. The result of this is that remote machines will have invalid entries in their ARP caches. Avoid this problem by using the ARP source address option.

7.1 Actions on Detecting Change

When NetMon detects an interface or group going up or down, the action delay timer is started (see option \texttt{ACT\_DELAY}) to attempt to stop any minor changes being acted on. Once this delay has expired, the following action is taken:

If you have set \texttt{REPORT\ RESMON} then a message will be sent to ResMon about the new state of the resource. If \texttt{REPORT\ SCRIPTS} is set, then the configured script is run. It is allowable to use both methods if needed.

If the number of groups down exceeds the configured number for network failure, then the network is marked as down. Once the network is marked down, no actions are taken until it comes back. When the network is considered up again, the action timers are re-started if needed.

\(^7\) netcat is installed as part of the HACbase package as /opt/HAC/bin/nc.
# Pre-Installation Worksheet

The following worksheets are provided to help plan an installation. Ideally they should be completed prior to the installation for each machine that will have NetMon running on it. Where NetMon will be installed on multiple machines on the same LAN they may be used to 'test' each other and the list of external machines to test for contact may remain the same for all nodes (except the removal of the current node from the list).

Machine Model ____________________  
Machine Name ____________________  
Hostid ____________________

## Table 3 Machines with NetMon installed

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Network Name</th>
<th>Subnet Mask</th>
<th>NetMon Group</th>
<th>Adapter Name</th>
<th>Backup Adapter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Table 4 Machines being monitored

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Protocol Used(^8)</th>
<th>Network Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^8\) The test protocol may be of the following **ntp/time/ping/arp**. There should be no difference in the operation of NetMon as a result of choosing a different protocol.
Appendix A. NetMon Worked Example

The following section documents the configuration of an example system running NetMon. This is a typical real world example where a system has two interfaces, with one virtual IP address being shared between the two. Netmon is used to monitor some external hosts to ensure the virtual IP is always available. For this example the following network topology is assumed:

The host mustang is a multi-honed Sun host with two unconfigured network interfaces, running `ifconfig -a` on mustang results in the following output:

```
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
    inet 127.0.0.1 netmask ff000000
le0: flags=862<BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 0.0.0.0 netmask ffffff00 broadcast 0.0.0.255
    ether 8:0:20:72:48:e8
le1: flags=863<BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 0.0.0.0 netmask ffffff00 broadcast 0.0.0.255
    ether 8:0:20:72:48:e9
```

Note: `le0` and `le1` have different MAC addresses.

We are interested in configuring one of these network interfaces with the virtual IP address **192.168.1.80**. Clients will then use this IP address to connect to **mustang**. ResMon configures one of these physical interfaces with the virtual IP address based on the results of the connectivity testing NetMon performs against **roady** and **silver**.
Pre-Installation Worksheet

Assuming ResMon and NetMon are installed, we first create a configuration file for NetMon:

```
# The named pipe for NetMon/ResMon communications.
RESMON PIPE /opt/HAC/NetMon/etc/pipe

INTERFACE le0
MACHINE 192.168.1.44 METHOD ARP

INTERFACE le1
MACHINE 192.168.1.17 METHOD ARP
```

Followed by one for ResMon:

```
# The named pipe for NetMon/ResMon communications.
PIPE /opt/HAC/NetMon/etc/pipe

# The interfaces NetMon is monitoring.
RESOURCE le0
RESOURCE le1

# Declare a group of interfaces
INTERFACE_GROUP public {
    REQUIRES le0 OR le1
}

# The IP address logically assigned to one of the interfaces.
IPADDRESS 192.168.1.80 {
    INTERFACE_GROUP public
    ALIASNO 1
}
```

Starting ResMon followed by NetMon will result in ResMon plumbing in the virtual IP address 192.168.1.80 to the first available interface, entering `ifconfig -a` now gives us:

```
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
    inet 127.0.0.1 netmask ff000000
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.80 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:72:48:e8
le1: flags=862<BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 0.0.0.0 netmask ffffff00 broadcast 0.0.0.255
        ether 8:0:20:72:48:e9
```

Clients now access `mustang` using the IP address **192.168.1.80** configured on le0. By disconnecting the cable connecting le0 to the switch, NetMon will signal ResMon that network connectivity has been lost on le0. ResMon will then move the IP address 192.168.1.80 from le0 to le1. Running `ifconfig -a` now produces the following:

```
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
    inet 127.0.0.1 netmask ff000000
le0: flags=863<BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 0.0.0.0 netmask ffffff00 broadcast 0.0.0.255
        ether 8:0:20:72:48:e8
le1: flags=862<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.80 netmask ffffff00 broadcast 192.168.1.255
        ether 8:0:20:72:48:e9
```
Another common application for NetMon is to test the default route on a host where there are two possible choices. Consider the following network topology:

**DEFAULT ROUTE: 192.168.1.1**
**STATIC ROUTE FOR 168.40.0.0 GATEWAY IS 192.168.1.1**
**STATIC ROUTE FOR 172.38.0.0 GATEWAY IS 192.168.1.2**

NetMon tests the ISP's routers directly connected to our default router. Should the default route fail due to either our, or the ISP's router failing, ResMon will reconfigure the default route to 192.168.1.2.
Appendix C. Syslog Messages

The following table documents the messages NetMon will report using the syslog facility upon occurrence of certain events/conditions. The left hand column lists the syslog standard message prefixes and has the format:

HAC[PRODUCT]-[IDENT][LEVEL]-[CODE]:

Where [PRODUCT] is NM (NetMon) or RM for ResMon, [IDENT] identifies the type of message and is either I for LOG_INFO or E for LOG_ERROR. [LEVEL] identifies the level the message will be reported at; this number is relative to the –l argument given to netmon when started up. An entry of [0] means the message will always be written, an entry of [1] corresponds to –l1, an entry of [2] corresponds to –l2. Note that using a log level of X means that messages with priority of X and below will be output, i.e. using a log level of –l2 means that entries corresponding to [0], [1] and [2] will be output. Finally [CODE] identifies the individual message code for the level given.

The right hand column is divided into and upper and lower section. The upper section documents the actual message written to syslog. Each line is preceded by a number in [], In the message part, any variables that are resolved during output by NetMon are listed in chevrons (<>). Finally the lower section documents the circumstances that have caused NetMon to issue the syslog message.

Syslog is opened using the LOG_DAEMON facility with the program name netmon. Some example messages taken from the syslog file is as follows:

```
netmon: HACNM-10-0001: lost packet(s) to router
netmon: HACNM-E0-0005: group interface `lan0’ fcntl failed
netmon: HACNM-W0-0001: warning timer drift (5 secs)
```

### Information messages, level 0

<table>
<thead>
<tr>
<th>HACNM_I0_0001</th>
<th>[arg1] arg2 host arg3 (arg4 up, arg5 down)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In response to a status request, a summary is printed for each interface being monitored (arg1), listing the number of hosts being monitored (arg2), the number marked as up (arg4) and the number marked as down (arg5). This message is only written to syslog if the -s flag is given to NetMon on start-up.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HACNM_I0_0002</th>
<th>group [arg1]: host arg2 (arg3) poll(s): arg4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In response to a fullstatus request, the message for HACNM-I-0001 is output, followed by a more detailed report. This report displays group monitoring for each group (arg1); a line at a time for each host args2 in the group. For each host the state arg3 is given (up or down) followed by a detailed description of each poll. Each poll is listed in arg4 (currently arp, ping and time), the amount of packets sent and the amount of packets lost. The poll(s) entry is repeated for each type of poll sent to the host. As with HACNM-I0-0001, this message is only written to syslog if the -s flag is given to NetMon.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HACNM_I0_0003</th>
<th>arg1 monitoring OK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An hourly log message indicating NetMon is running as expected. Enabled using the -i command line switch; arg1 is the program name as started,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HACNM_I0_0004</th>
<th>arg1 starting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Log message issued when NetMon is first started; arg1 is the program name as started.</td>
</tr>
</tbody>
</table>
### Information messages, level 1

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_I1_0006</td>
<td>Netmon has detected that the hardware address of interface arg1 has changed from arg2 to arg3</td>
</tr>
<tr>
<td>HACNM_I1_0007</td>
<td>Process priority for NetMon is now arg1. Value is taken from the config file PROCESS_PRIORITY entry if it exists.</td>
</tr>
<tr>
<td>HACNM_I1_0008</td>
<td>NetMon has detected that the interface IP address for arg1 has changed during normal operation.</td>
</tr>
<tr>
<td>HACNM_I1_0009</td>
<td>NetMon has detected that packets sent to arg1 have started to be lost.</td>
</tr>
<tr>
<td>HACNM_I1_0010</td>
<td>The WEB front end module of NetMon started after arg1 attempts.</td>
</tr>
</tbody>
</table>

### Information messages, level 2

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_I2_0011</td>
<td>Executing program 'arg1' when an interface group changes state (up/down) and if the use script option is enabled, the command line NetMon is going to run is logged as arg1.</td>
</tr>
</tbody>
</table>

### Error messages, level 0

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_E0_0001</td>
<td>Too many hosts in arg1, no reports on host arg2. The number of hosts to monitor for group arg1 has exceeded the maximum allowed, monitoring of the host arg2 will not take place</td>
</tr>
<tr>
<td>HACNM_E0_0002</td>
<td>Host arg1 does not belong in arg2. An internal overflow error has occurred causing arg1 to be referred to in the wrong group (arg2). This is a fatal error and causes NetMon to exit() with status code 255. Should this error occur, it will only be on start-up, during the loading of the config file. It is usually indicative of extraordinary large config files.</td>
</tr>
<tr>
<td>HACNM_E0_0003</td>
<td>Group src interface 'arg1' not found, ignoring. During start-up, NetMon could not locate interface arg1 on the system and therefore has ignored the config file entries related to this interface.</td>
</tr>
<tr>
<td>HACNM_E0_0004</td>
<td>Group interface 'arg1' could not be opened. During start-up, NetMon could not open the interface arg1. NetMon will exit() with the status code 1.</td>
</tr>
<tr>
<td>HACNM_E0_0005</td>
<td>Group interface 'arg1' fcntl failed. During start-up NetMon failed to get parameters for interface arg1, this is a fatal error and causes NetMon to exit() with a status code of 1.</td>
</tr>
<tr>
<td>HACNM_E0_0006</td>
<td>Group interface 'arg1' non-blocking failed. During start-up NetMon failed to set interface arg1 to non-blocking, this is a fatal error and causes NetMon to exit() with a status code of 1.</td>
</tr>
<tr>
<td>HACNM_E0_0007</td>
<td>Group[arg1] cannot be bound, feature is not enabled. Raw access to devices on this architecture is not supported by NetMon.</td>
</tr>
<tr>
<td>HACNM_E0_0008</td>
<td>Failed to allocate memory for 'arg1' in makehost. Indicates that available memory was exhausted during start-up. This causes NetMon to exit() with a status code of 2.</td>
</tr>
</tbody>
</table>
| HACNM_E0_0009 | Cannot find method 'arg1', using default. The config file named the poll method arg1 for the group/host. However, NetMon
### Pre-Installation Worksheet

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_E0_0010</td>
<td>Could not find this method and has reverted back to the default method.</td>
</tr>
<tr>
<td>HACNM_E0_0011</td>
<td>Failed to lock process into memory: arg1</td>
</tr>
<tr>
<td>HACNM_E0_0012</td>
<td>NetMon was unable to unlock itself from memory using memunlock() - reason given in arg1</td>
</tr>
<tr>
<td>HACNM_E0_0013</td>
<td>This build of NetMon does not support locking the process into memory. This is usually because the underlying operating system does not support the call.</td>
</tr>
<tr>
<td>HACNM_E0_0014</td>
<td>The group arg1 received a packet that has its transmission time in the future - this is usually a by-product of the host operating system having its clock changed. NetMon recovers by resetting its internal clocks.</td>
</tr>
<tr>
<td>HACNM_E0_0015</td>
<td>gettimeofday() failed, arg2</td>
</tr>
<tr>
<td>HACNM_E0_0016</td>
<td>NetMon has been unable to find a valid license file/key for this host and will exit() code 1.</td>
</tr>
<tr>
<td>HACNM_E0_0017</td>
<td>During execution at arg1, the gettimeofday() call failed with error arg2</td>
</tr>
<tr>
<td>HACNM_E0_0018</td>
<td>NetMon was unable to open its PID file (arg1) for writing, system error returned in arg2.</td>
</tr>
<tr>
<td>HACNM_E0_0019</td>
<td>NetMon was unable to write to it's PID file (arg1), system error returned in arg2.</td>
</tr>
<tr>
<td>HACNM_E0_0020</td>
<td>NetMon was unable to initialise the ResMon (system name in arg1) section of code given in arg2, either the API or the PostConfig.</td>
</tr>
<tr>
<td>HACNM_E0_0021</td>
<td>During execution, the code identified in arg1 encountered an error reading file arg2, system error number given in arg3.</td>
</tr>
<tr>
<td>HACNM_E0_0022</td>
<td>During initialisation of the NT start-up service code a failure occurred. NetMon will exit() code 1.</td>
</tr>
<tr>
<td>HACNM_E0_0023</td>
<td>Errors occurred (count in arg1) during the loading of the config file. NetMon will exit() code 1.</td>
</tr>
<tr>
<td>HACNM_E0_0024</td>
<td>No hosts have been configured for monitoring, giving NetMon nothing to do! NetMon will exit() code 1.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HACNM_E0_0025</td>
<td>setpriority() failed during initialisation: arg1</td>
</tr>
<tr>
<td>HACNM_E0_0026</td>
<td>During initialisation the setpriority() call failed with error arg1</td>
</tr>
<tr>
<td>HACNM_E0_0026</td>
<td>ntp module: failed to initialise network info</td>
</tr>
<tr>
<td>HACNM_E0_0027</td>
<td>The NTP module failed to find a valid udp port for the ntp service. NetMon will exit() code 1.</td>
</tr>
<tr>
<td>HACNM_E0_0027</td>
<td>The NTP module detected an error whilst sending a UDP packet. This is usually a temporary network failure error.</td>
</tr>
<tr>
<td>HACNM_E0_0028</td>
<td>ntp module: sendto(arg1) sent arg2, requested arg3</td>
</tr>
<tr>
<td>HACNM_E0_0028</td>
<td>During execution at arg1, the opendir(arg2) call failed with error arg3</td>
</tr>
<tr>
<td>HACNM_E0_0029</td>
<td>config: priority-boost must be an integer</td>
</tr>
<tr>
<td>HACNM_E0_0029</td>
<td>The argument given to the priority for NetMon in the config file was found to be invalid.</td>
</tr>
<tr>
<td>HACNM_E0_0030</td>
<td>config: arg1 script not settable here</td>
</tr>
<tr>
<td>HACNM_E0_0030</td>
<td>During parsing of the config file, the script identified in arg1 (UP/DOWN) is not configurable. Typically means it is not declared in the correct section of the configuration file.</td>
</tr>
<tr>
<td>HACNM_E0_0031</td>
<td>config: configuring item REPORT, and null options pointer</td>
</tr>
<tr>
<td>HACNM_E0_0031</td>
<td>This message indicates the options area is un-initialised within NetMon. This usually indicates a severe memory shortage.</td>
</tr>
<tr>
<td>HACNM_E0_0032</td>
<td>config: could not copy string 'arg1'</td>
</tr>
<tr>
<td>HACNM_E0_0032</td>
<td>A copy could not be made of the string given in arg1. This usually indicates a severe memory shortage.</td>
</tr>
<tr>
<td>HACNM_E0_0033</td>
<td>config: config_start_group() called with a null group</td>
</tr>
<tr>
<td>HACNM_E0_0033</td>
<td>Memory for a new group could not be in initialised. This usually indicates a severe memory shortage.</td>
</tr>
<tr>
<td>HACNM_E0_0034</td>
<td>config: resource specified, but group null?</td>
</tr>
<tr>
<td>HACNM_E0_0034</td>
<td>Indicates an error in the ordering of members of the config file.</td>
</tr>
<tr>
<td>HACNM_E0_0035</td>
<td>config: could not build resource arg1</td>
</tr>
<tr>
<td>HACNM_E0_0035</td>
<td>Memory for a new group member could not be in initialised. This usually indicates a severe memory shortage.</td>
</tr>
<tr>
<td>HACNM_E0_0036</td>
<td>config: no support for RESOURCE configuration option</td>
</tr>
<tr>
<td>HACNM_E0_0036</td>
<td>Attempt to use a configuration file option not supported in this version of NetMon.</td>
</tr>
<tr>
<td>HACNM_E0_0037</td>
<td>config: no support for PIPE configuration option</td>
</tr>
<tr>
<td>HACNM_E0_0037</td>
<td>Attempt to use a configuration file option not supported in this version of NetMon.</td>
</tr>
<tr>
<td>HACNM_E0_0038</td>
<td>config: config_build_machine() called with null group</td>
</tr>
<tr>
<td>HACNM_E0_0038</td>
<td>Indicates an error in the ordering of members of the config file.</td>
</tr>
<tr>
<td>HACNM_E0_0039</td>
<td>failed to make icmp socket, error arg1 (arg2)</td>
</tr>
<tr>
<td>HACNM_E0_0039</td>
<td>A call to socket() failed when creating an ICMP socket for ping requests, reason for failure given in arg1.</td>
</tr>
<tr>
<td>HACNM_E0_0040</td>
<td>bind() call failed for socket arg1, arg2 (arg3)</td>
</tr>
<tr>
<td>HACNM_E0_0040</td>
<td>A call to bind() failed during an attempt to bind to socket arg1, reason for failure given in arg2.</td>
</tr>
<tr>
<td>HACNM_E0_0041</td>
<td>method ping: cannot be used with interface specified</td>
</tr>
</tbody>
</table>
| HACNM_E0_0041     | An attempt to send out a ping packet failed on this interface.  

Revision 2.2
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_E0_0042</td>
<td>An error occurred sending out a ping packet to host arg1, error given in arg2.</td>
</tr>
<tr>
<td>HACNM_E0_0043</td>
<td>The number of bytes sent out in a ping packet to host arg1 did not match those returned, sent bytes in arg2, returned bytes in arg3.</td>
</tr>
<tr>
<td>HACNM_E0_0044</td>
<td>An attempt to add an ARP entry to the internal NetMon arp cache failed. Usually indicates a memory shortfall.</td>
</tr>
<tr>
<td>HACNM_E0_0045</td>
<td>Data has been received from a host we are not monitoring (listed in arg1) - the data will be ignored.</td>
</tr>
<tr>
<td>HACNM_E0_0046</td>
<td>NetMon has received data on an unconfigured socket - the packet will be ignored.</td>
</tr>
<tr>
<td>HACNM_E0_0047</td>
<td>A problem occurred accessing the NT registry whilst loading configuration values - defaults will be used instead.</td>
</tr>
<tr>
<td>HACNM_E0_0048</td>
<td>An error was returned during initialisation of the ResMon interface structures, typically indicates a memory shortfall.</td>
</tr>
<tr>
<td>HACNM_E0_0049</td>
<td>The resource identified in arg1 failed to register with ResMon, and as such ResMon will not be sent details regarding the state of the resource.</td>
</tr>
<tr>
<td>HACNM_E0_0050</td>
<td>Indicates the server interface to ResMon failed to initialise earlier in the connection process. No data will be sent to ResMon.</td>
</tr>
<tr>
<td>HACNM_E0_0051</td>
<td>Calls to malloc() are failing, usually indicates a memory shortfall.</td>
</tr>
<tr>
<td>HACNM_E0_0052</td>
<td>Calls to malloc() are failing, usually indicates a memory shortfall.</td>
</tr>
<tr>
<td>HACNM_E0_0053</td>
<td>NetMon cannot send raw packet data to the socket as it is not of type AF_INET (attempt to use a bound interface on a non InterNet address family).</td>
</tr>
<tr>
<td>HACNM_E0_0054</td>
<td>Attempt to use a raw socket when this architecture does not support it.</td>
</tr>
<tr>
<td>HACNM_E0_0055</td>
<td>A socket exists that is bound to no group - this is an assertion test and indicates an internal error has occurred.</td>
</tr>
<tr>
<td>HACNM_E0_0056</td>
<td>NT only: if the product RSF-1 is not installed NetMon cannot initialise and therefore will fail to start.</td>
</tr>
<tr>
<td>HACNM_E0_0057</td>
<td>The page type given in arg1 failed to be created - this is a temporary error.</td>
</tr>
<tr>
<td>HACNM_E0_0058</td>
<td>A call to getsockname() on socket arg1 failed, reason given in arg2, error number</td>
</tr>
</tbody>
</table>
### Pre-Installation Worksheet

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_E0_0059</td>
<td><code>socket()</code> failed, arg1 (arg2) A call to <code>socket()</code> failed, reason given in arg1, error number in arg2.</td>
</tr>
<tr>
<td>HACNM_E0_0060</td>
<td><code>getservbyname(arg1, arg2)</code> failed, using default: arg3 A call to <code>getservbyname()</code> on service arg1, protocol arg2 failed, the default port number given in arg3 is used.</td>
</tr>
<tr>
<td>HACNM_E0_0061</td>
<td><code>malloc()</code> failed for request of length arg2, arg3 During execution, the code identified in arg1 failed to <code>malloc()</code> memory of size arg2, error given in arg3.</td>
</tr>
<tr>
<td>HACNM_E0_0062</td>
<td><code>do_swap_ifs: cannot create interface 'arg1'</code> A call to <code>malloc()</code> failed during interface structure initialisation. This message is only issued under test conditions and not part of normal NetMon running.</td>
</tr>
<tr>
<td>HACNM_E0_0063</td>
<td><code>swap_ifs: failed to list interfaces</code> An attempt to get a list of the configured system interfaces failed. This message is only issued under test conditions and not part of normal NetMon running.</td>
</tr>
<tr>
<td>HACNM_E0_0064</td>
<td><code>scandir(arg1)</code> failed, arg2 (arg3) A call to <code>scandir(arg1)</code> failed, reason given in arg2, error number in arg3.</td>
</tr>
<tr>
<td>HACNM_E0_0065</td>
<td><code>stat(arg1)</code> failed, arg2 (arg3) A call to <code>stat(arg1)</code> failed, reason given in arg2, error number in arg3.</td>
</tr>
<tr>
<td>HACNM_E0_0066</td>
<td><code>unlink(arg1)</code> failed, arg2 (arg3) A call to <code>unlink(arg1)</code> failed, reason given in arg2, error number in arg3.</td>
</tr>
<tr>
<td>HACNM_E0_0067</td>
<td><code>fopen(arg1)</code> failed for arg2, arg3 (arg4) A call to <code>fopen(arg1)</code> failed for reading or writing (arg2), reason given in arg3, error number in arg4.</td>
</tr>
<tr>
<td>HACNM_E0_0068</td>
<td><code>rename(arg1, arg2)</code> failed, arg3 (arg4) A call to <code>rename(arg1, arg2)</code> failed, reason given in arg3, error number in arg4.</td>
</tr>
<tr>
<td>HACNM_E0_0069</td>
<td><code>cannot get list of service directories</code> No RSF-1 service directories were found. This message is only issued under test conditions and not part of normal NetMon running.</td>
</tr>
<tr>
<td>HACNM_E0_0070</td>
<td>poll method time; udp time port number unknown. During initialisation, NetMon could not locate the correct port number for the UDP time protocol. This is a fatal error and NetMon will <code>exit()</code> with status code 1.</td>
</tr>
<tr>
<td>HACNM_E0_0071</td>
<td>poll method time: sendto(arg1) returned arg2 (arg3) A network write in the polling method time failed sending to host arg1, reason given in arg2, error number in arg3.</td>
</tr>
<tr>
<td>HACNM_E0_0072</td>
<td>netmon spawn: &quot;arg1&quot; exited with code arg2 An attempt by NetMon to spawn the command given in arg1 failed with exit code arg2.</td>
</tr>
</tbody>
</table>
| HACNM_E0_0073       | netmon spawn: fork() failed during command execution, arg1 (arg2) A call to `fork()` failed, reason given in arg1, error number in arg2.
### Error messages, level 1

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_E0_0075</td>
<td>getprotobyname(&quot;arg1&quot;) failed, using default port: arg2. A call to getprotobyname( arg1 ) failed, the default port given in arg2 will be used instead.</td>
</tr>
<tr>
<td>HACNM_E1_0001</td>
<td>arg1 failed to send poll arg2. NetMon was unable to send a single poll of type arg11 to the host specified in arg2.</td>
</tr>
<tr>
<td>HACNM_E1_0002</td>
<td>send-arp: arg1 cannot add arp entry. NetMon was unable to add to its internal ARP cache table - usually indicates a short term memory shortfall.</td>
</tr>
<tr>
<td>HACNM_E1_0003</td>
<td>arg1 method 'arg2' reports data error. During execution, the code identified in arg1 detected an error on an incoming packet in method arg2. The packet is ignored.</td>
</tr>
<tr>
<td>HACNM_E1_0004</td>
<td>NetMon encountered an error during the recvfrom() system call. The error is given in arg1.</td>
</tr>
<tr>
<td>HACNM_E1_0005</td>
<td>setarg1() failed: arg2 (arg3). A call to setsid(), setgid() or setuid() failed. The error is given in arg2.</td>
</tr>
<tr>
<td>HACNM_E1_0006</td>
<td>webfront: could not start server. The WEBFRONT server could not be started.</td>
</tr>
<tr>
<td>HACNM_E1_0007</td>
<td>webfront: failed to start after arg1 attempts. After arg1 attempts the webfront server could not be started - NetMon will continue to retry.</td>
</tr>
<tr>
<td>HACNM_E1_0008</td>
<td>arp_ioctl() returned error, arg1 (arg2). NetMon encountered an error during the arp_ioctl() system call (attempt to read the system ARP cache). NetMon continues using its own ARP cache.</td>
</tr>
<tr>
<td>HACNM_E1_0009</td>
<td>group[arg1]: couldn't refresh interface info. For the group interface given in arg1, NetMon could not refresh the MAC address information - the previously stored information will be maintained instead of overwritten.</td>
</tr>
<tr>
<td>HACNM_E1_0010</td>
<td>read error from socket arg1, arg2 (arg3). A call to socket() returned a temporary read error, given in arg2.</td>
</tr>
</tbody>
</table>

### Error messages, level 2

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_E2_0001</td>
<td>waitpid() returned error, arg1 (arg2). A call to waitpid() returned a temporary error whilst waiting for a child process, the error is given in arg2.</td>
</tr>
<tr>
<td>HACNM_E2_0002</td>
<td>arp: error getting ARP for arg1, arg2 (arg3). NetMon was unable to update its ARP cache for host arg1. The host or network may be down.</td>
</tr>
<tr>
<td>HACNM_E2_0003</td>
<td>arp: write failed for socket arg1. An attempt to write an ARP packet on interface arg1 failed.</td>
</tr>
<tr>
<td>HACNM_E2_0004</td>
<td>arp: ifinfo null for interface arg1 (does interface exist?) NetMon failed to read interface information for interface arg1 and as such cannot perform any monitoring on this interface.</td>
</tr>
</tbody>
</table>
### Pre-Installation Worksheet

**HACNM_E2_0005**  
webfront: fork() failed, arg1 (arg2)  
A call to `fork()` failed in the webfront code, reason given in arg1, error number in arg2.

**HACNM_E2_0006**  
failed to send state information to ResMon  
A `write()` to the ResMon named pipe failed. Usually indicates ResMon is not running and the system pipe buffers are exhausted.

### Alert messages, level 0

**HACNM_A0_0001**  
arg1 is now UP  
Host arg1 is now UP.

**HACNM_A0_0002**  
arg1 is now DOWN  
Host arg1 is now DOWN.

**HACNM_A0_0003**  
network is UP  
The network is now in stat UP.

**HACNM_A0_0004**  
network is DOWN  
The network is now in state DOWN, indicating a total failure of all monitored hosts through all monitored network interfaces.

### Alert messages, level 1

**HACNM_A1_0001**  
host arg1 (group 'arg2') is now DOWN, lost arg3 packets  
During interface recovery stage typically host packets tend to be responded to intermittently, this warning is issued to flag the process of recovery and indicates packets are being dropped.

**HACNM_A1_0002**  
host arg1 (group 'arg2') is UP  
During interface recovery stage typically host packets tend to be responded to intermittently, this warning is issued to flag the process of recovery, and indicates the remote host is now responding.

### Warning messages, level 0

**HACNM_W0_0001**  
check_rings: group 'arg1' has arg2 hostarg3 it should not  
When the maximum number of allowable hosts to be monitored has been exceeded (on a per interface basis), NetMon issues this warning and reduces the monitored host list to the maximum allowed.

**HACNM_W0_0002**  
arg1 restarted notify timer  
This message is typically issued when the system time has been reset and NetMon has to adjust its internal timers to compensate.

**HACNM_W0_0003**  
arg1 not licensed, exiting.  
No valid license was found for NetMon and as such it will `exit()` with status code 2.

**HACNM_W0_0004**  
time shift backwards (arg1 seconds)  
Issued when NetMon notices the system time has shifted backwards by arg1 seconds.

**HACNM_W0_0005**  
config: group arg1 defined with no machines  
A group has been defined (i.e. an interface), but no machines have been configured for monitoring. This group will be ignored.

**HACNM_W0_0006**  
web-dbg: could not create arg1  
An attempt to write a web page failed - typically indicates a temporary resource error.
<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>HACNM_W1_0001</td>
<td>warning: timing drift (arg1 secs)</td>
</tr>
<tr>
<td>HACNM_W1_0002</td>
<td>internal time value is drifting arg1, is arg2, tv=arg3</td>
</tr>
<tr>
<td>HACNM_W1_0003</td>
<td>gettimeofday() failed - can't check current timers</td>
</tr>
<tr>
<td>HACNM_W1_0004</td>
<td>method ntp: sendto(arg1) returned arg2</td>
</tr>
<tr>
<td>HACNM_W1_0005</td>
<td>method time: sendto(arg1) arg2 bytes sent, arg3 requested</td>
</tr>
<tr>
<td>HACNM_W1_0006</td>
<td>method: time: sendto(arg1) arg2 bytes sent, arg3 requested</td>
</tr>
<tr>
<td>HACNM_W1_0007</td>
<td>method: time: sendto(arg1) arg2 bytes sent, arg3 requested</td>
</tr>
</tbody>
</table>
Appendix D. Installation

NetMon is supplied in binary format using a standard naming convention for all installation types. This convention is as follows:

HACnetmon-OS-arch.major.minor.rel.yyyy-mm-dd.type

Where OS-arch defines the target operating system and architecture, for example hpux-parisc or solaris-sparc. The release of NetMon is given in major.minor.rel, i.e. 2.2.14. The build date of the package in yyyy-mm-dd, i.e. 2004-01-10 and finally the type of package in type, currently this is rpm, depot or pkg. Downloads of the most recent version are available from the High-Availability WEB site here.

NetMon is supplied in a package format specific for the destination operating system, the current available packages being:

- Solaris pkgadd format
- HPUX depot format
- Linux RPM format

For specific patch and OS version number support please see the OS specific README file in the distribution root directory.

A minimum of 2 Mb free disk space is required for the basic installation, however, depending upon the logging level configured, coupled with the number of interfaces being monitored more free space will probably be required to accommodate log files.

**Solaris**

Supplied in pkgadd distribution format, use the pkgadd command as follows:

```
```

When prompted press return to select all available packages.

**HPUX**

Supplied in depot distribution format, use the swinstall command as follows:

```
```

This will install the complete package including the streams packet filter module required for correct NetMon operation under HPUX (viewed as package HACnetmon.pfmod using the command ‘swlist HACnetmon’).

**Linux**

Supplied in RPM distribution format, use the RPM command as follows:

```
rpm –i HACnetmon-2.2.13-1.i386.rpm
```