Tivoli Netcool Supports Guide to the Ping Probe

by

Jim Hutchinson

Document release: 2.4

Table of Contents

1 Introduction

1.1 Overview

The probe pings the hosts specified in the ping file in cycles.
The minimum length of each cycle is specified by the **Interval** property.
During each ping cycle, the probe pings the number of hosts specified by the **MaxPingBurst** property.

In each cycle, the probe:

- Finds the next host that has not been deemed unreachable or responsive during the current poll time.
  - Pings *current* host.
  - Checks whether it has pinged the number of hosts specified by the **MaxPingBurst** property during the current cycle. If not, it goes and gets the next host
- Once the probe has pinged the number of hosts specified by the **MaxPingBurst** property, it does the following:
  - Processes all the responses received so far.
  - Processes any timeout's that have occurred.
  - Reloads the ping file if the time specified by the **PingFileCheck** property has elapsed.
- Sleeps for any remaining time specified by the Interval property.

1.2 Latest test fix patch

The latest test fix patch includes the 64-bit binaries and can be requested from IBM Support.

```
PATCH probe-nco-p-ping-7

REVISION 5:
~~~~~~~~~~
APAR IJ06848 - Fixed probe unable to recognise FQDN string as host address but instead falsely
               consider the string as IP address. Target host was also found unreachable to probe
               ping requests. These issues may cause probe to core dump in 64-bit AIX platform.
REVISION 4:
~~~~~~~~~~
APAR IJ03219 / RFE 114760 - Added 64-bit probe binaries for Solaris, Linux, AIX and zLinux platforms.
REVISION 3:
~~~~~~~~~~
APAR IV87938 - Introduced SocketBufferSize property to allow users to increase maximum buffer
               size in socket and avoid ping replies from being dropped.
REVISION 2:
~~~~~~~~~~~
APAR IV60967 - The ping response time on Windows was always set to 0 ms. This has been fixed.
REVISION 1:
~~~~~~~~~~~
APAR IV59592 / RFE 50915 - Probe now generates new elements that available in rules file as:
- $poll_rate
- $trip_time
- $timeout_time
REVISION 0:
~~~~~~~~~~~
GA release of IPv6 support for Linux (as well as other Unix
platforms) and core dump on shut down fixed
```

1.3 The Interval property

The new property **Interval** is used to define the minimum amount of time set aside for each ping:
If the probe pings and processes the responses in a time [given in milliseconds] shorter than the Interval property setting, it sleeps for the remainder of the Interval period before starting the next ping cycle.

Therefore if you consider all the hosts to be available, and with negligible response times, it is possible to determine the maximum number of hosts that can be pinged using the default property settings:

```
Interval                        : 100
MaxPingBurst                    : 1
Poll                            : 600
```

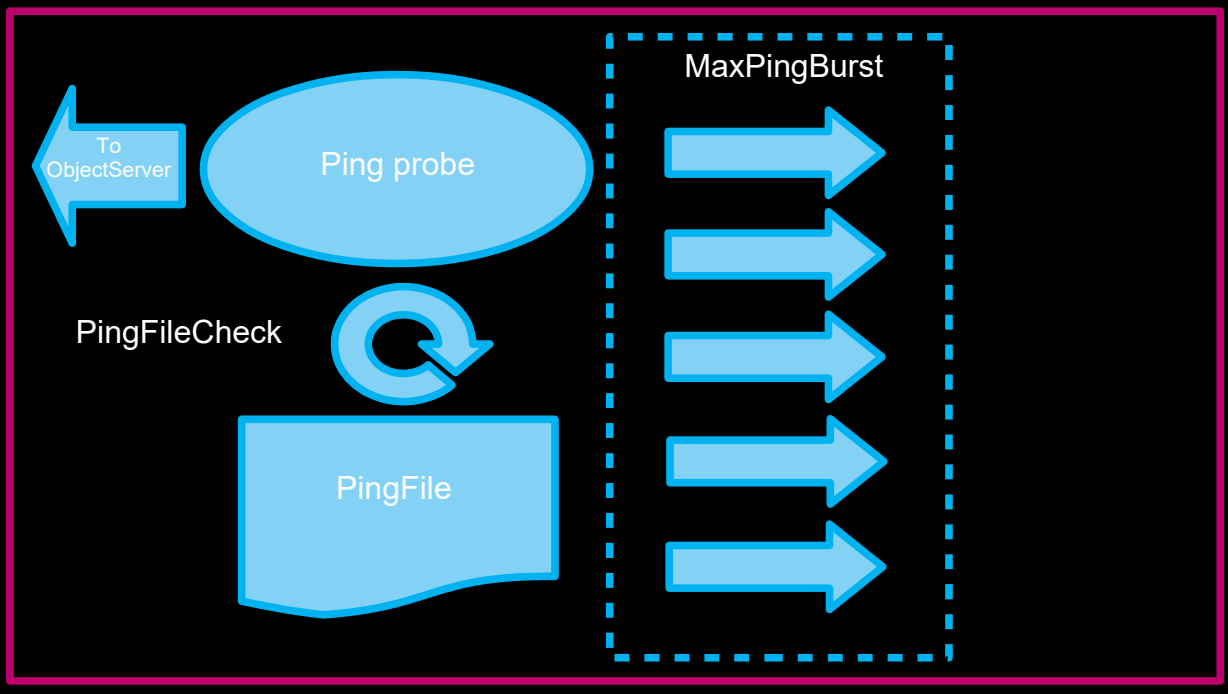
Maximum number of hosts = MaxPingBurst * (Poll / Interval) = 1 * (600 / 0.1) = 6,000 hosts

1.4 Ping probe behaviour

The ping probe polls host periodically based on a file containing a list of ip addresses or hostnames. The PingFile is read periodically by a time given by PingFileCheck, and when modified all the hosts are re-read, with the new host list being compared against the current. New and removed hosts create events, as do ping status checks, and problems with name resolution.

Whilst peer-to-peer is supported by the Ping Probe, it does not make sense to use it, since the Ping Probe is a polling probe, so therefore loads the network unnecessarily. It is best to have a cold standby ping probe available, in the event that the primary probe becomes unavailable.

The diagram below highlights the main properties related to the PingFile.



2 Properties

2.1 PingFile

The ping file property sets the path to the file that holds all the hosts that need to be pinged. It's header includes the allowed format for specifying host specific settings:

```
#       host [ poll rate ]
#       host [ poll rate ] [trip time ]
#       host [ poll rate ] [trip time ] [timeout time]
```

The ping.file settings override the probes default settings given in the probes property file.
The period in which the ping.file is re-read is given by the PingFileCheck property.
The hostname lookups are only refreshed when the probe is started or when the ping file is updated.
Therefore to update hostname to ip address lookups, the ping file needs to be modified.

2.2 BindAddress

The Ping probe binds to all available interfaces, but can be restricted to a given interface using the BindAddress property.

2.3 Poll

Time in seconds between pings for each host. If the host responds in a timely manner, this will be consistent, and close to the multiples of the Poll setting.

2.4 TimeOut

Time in milliseconds after which a host is deemed unreachable. Timeout should be set to a maximum expected response time, plus a percentage, to allow adequate reporting of host responsiveness.

2.5 Trip

Time in milliseconds after which a host is deemed slow. Trip should be set to expected response time, plus a percentage, to allow the slow response event to be generated, when applicable.

2.6 Retry

The maximum number of times a host is pinged in a Poll interval. The Retry property can be set to 1, or else to a value that allows for any network issues to be recovered from, before reporting the host as unavailable.

2.7 MaxPingBurst

The maximum number of hosts to ping in parallel.

Setting MaxPingBurst to '1' prevents the likelihood of hosts being deemed 'unreachable' and forces the individual host Polls to be spread sequentially, even on start-up.

Whilst the likelihood of a host being deemed 'unreachable' is increased by setting MaxPingBurst to a value greater than '1', since there may be issues with the network to the hosts being polled (i.e. severe packet loss).

The benefit of setting a high MaxPingBurst is that more hosts can be pinged within a specific time period.

3 Configuration

3.1 Poll Considerations

The Poll property dictates the period of the polling of each host, poll can also be specified individually, for each host row in the ping.file.

```
host [ poll rate ] [trip time ] [timeout time]
```

Consider the following examples:
If 'host_a' does not respond in the TimeOut period for Retry times, its next poll will be given by:
Poll+(TimeOut*Retry)

If 'host_b' responds immediately, it will be polled after a time given by:
Poll+response_time

Example of host spread:

MaxPingBurst: 2
Poll: 60
TimeOut: 5000
Trip: 2500
Retry: 12

For 'host_a' : Poll+(TimeOut*Retry) = 60+5*12 = 120s
Polls occur at : 0s, 120s, 240s, 360s, etc.
For 'host_b': Poll+response_time = 60+0.1
Polls occur at : 0s, 60.1s, 120.2s, 180.3s. etc.

Therefore, because 'host_a' is always timing out, its results are not seen regularly, but instead at 120s intervals.

Whilst 'host_b' initially appears to be regular, but due to the response time, slowly deviates from the original timing.

3.2 Example settings

Based on the Ping probes default property settings:

```
# Interval                      : 100
# MaxPingBurst                  : 1
# Poll                          : 600
# Retry                         : 3
# TimeOut                       : 2500
# Trip                          : 2500
```

Consider the following network conditions:

- 10,000 hosts with average trip times of 250ms [0.25s]
- 1% average failure rate (10,000*0.01 => 100 hosts)

(((number of hosts) * (trip time)) / MaxPingBurst) + (((average failures) * (Retry * TimeOut)) / MaxPingBurst)

e.g.
(10,000*0.25)+(100*3*2.5) = (2500)+(750) = 3250s ~ **60 minutes**

3.3 SocketSize

There was an issue found when large volumes of ICMP ping packets were being read by the Ping Probe and a new property was added to the probe.

IV87938: PING PROBE MISSING PING RESPONSE DUE TO SOCKET BUFFER SIZE LIMITATION

The default socket buffer size for ICMP port needs to be set to a value such as 8 Kbytes. When the ping probe receives ping responses in bulk, the probes socket buffer can overflow, resulting in ICMP ping packets being discarded by Operating System. Depending upon the overall load and probe Retry settings, this may result in probe reporting hosts as down or unresponsive in Object Server.

To minimise risk of missing ping response, it is recommended that the SocketBuffer property is increased to allow more data to be stored in socket buffer.

For AIX the maximum socket size allowed is 64 Kbytes.

3.4 PacketSize

You can set the packet size for the ICMP packet.
This might be useful when using MessageLevel:debug and you require clearer logging.
For example:

```
PacketSize              : 1024
```

Logs these messages for non-ICMP packets:

```
Packet too short. Discarding.
```

Whilst

```
PacketSize               : 32
```

Logs these messages for non-ICMP packets:

```
Packet not ICMP_ECHOREPLY. Discarding.
```

4 Customisation

It is possible to enhance the ping probe rules file to allow greater functionality.
In the example rules file the previous $status token is stored to allow additional generic clearing, and the $response_time token is created to allow it to be stored in a custom Object Server field, if required.

4.1 Elements

Element name	Element description
$alias	This element identifies the alias of the host
$host	This element shows the name of the host
$icmp_stats	This element contains the standard ICMP statistics of the host
$ip_address	This element shows the IP address of the host
$status	This element indicates the status of the host: unreachable: host has failed to respond responded: host has responded alive: host has responded after previously failing to respond slow: host has taken longer than trip time but less than poll-rate to respond noaddress: probe cannot find the IP addressof the host removed: host is removed from the ping file newhost: host is added to the ping file

4.2 Example rules file #1

The following rule file uses the standard method of Generic Clear problem resolution, and multiple events per Node; It allows unreachable events to be cleared by responded events.

```
#Define arrays
array previous_status_array
if( match( @Manager, "ProbeWatch" ) )
{
        switch(@Summary)
        {
        case "Running ...":
                @Severity = 1
                @AlertGroup = "probestat"
                @Type = 2
        case "Going Down ...":
                @Severity = 5
                @AlertGroup = "probestat"
                @Type = 1
        default:
                @Severity = 1
        }
        @AlertKey = @Agent
        @Summary = @Agent + " probe on " + @Node + ": " + @Summary
} else {
        @Identifier = $host + $status
        @Node = $host
        @NodeAlias = $alias
        @Manager  = %Manager
        @AlertGroup = "Ping"
         @AlertKey = $host
        @Class = 100

        $previous_status_array = "none"
        if ( !match("",previous_status_array[$host]) )
        {
                $previous_status_array = previous_status_array[$host]
        }

        switch ($status)
        {
                case "unreachable" :
                        @Severity = 5
                        @Summary = @Node + " is not reachable"
                        @Type = 1
                case "alive" :
                        @Severity = 3
                        @Summary = @Node + " is now alive"
                        @Type = 2
                case "noaddress" :
                        @Severity = 2
                        @Summary = @Node + " has no address"
                case "removed" :
                        @Severity = 5
                        @Summary = @Node + " has been removed"
                case "slow" :
                        @Severity = 2
                        @Summary = @Node + " has not responded within trip time"
                case "newhost" :
                        @Severity = 1
                        @Summary = @Node + " is a new host"
                case "responded" :
                        if( match($previous_status_array,"unreachable") )
                        {
                                @Severity = 1
                                @Type = 2
                        }
                        else
                        {
                                @Severity = 0
                        }
                        @Summary = @Node + " has responded"
                case "default" :
                        @Summary = "Ping Probe Error details : " + $*
                        @Severity = 3
        }
```

```
#
# Save current status for next time
#
        previous_status_array[$host] = $status
#
# icmp_stats:   %PacketSize bytes from <ip-address>: icmp_seq=<incr>. ttl=<n> time=<n> ms
#
        if (exists($icmp_stats))
        {
                $response_time = extract ($icmp_stats, "time=([0-9]+) ms" )
                log(warn,"ResponseTime = " + $response_time)
        }

}
#EOF
```

4.3 Example rules file #2

The following rule file still uses the standard method of Generic Clear problem resolution, and multiple events per Node.

However, it includes array logic to control unreachable and responded events;
Set NumberOfUnreachables to the number of Unreachable pings that constitute a ping failure, and needs to be reported.
Set NumberOfResponded to the number of Responded pings that constitute the node is alive, following the Unreachable pings [resets the counter for Unreachable pings] .

```
##############################################
#Define arrays at the top
##############################################
array previous_status_array
array unreachable_array
array responded_array
##############################################
# Initialise variables
##############################################
$NumberOfUnreachables = 1
$NumberOfResponded    = 1

$previous_status_array = "none"

if ( match("",unreachable_array[$host]) )
{
   unreachable_array[$host] = 0
   responded_array[$host] = 0
}

##############################################
# ProbeWatch
##############################################
if( match( @Manager, "ProbeWatch" ) )
{
        switch(@Summary)
        {
        case "Running ...":
                @Severity = 1
                @AlertGroup = "probestat"
                @Type = 2
        case "Going Down ...":
                @Severity = 5
                @AlertGroup = "probestat"
                @Type = 1
        default:
                @Severity = 1
        }
        @AlertKey = @Agent
        @Summary = @Agent + " probe on " + @Node + ": " + @Summary
        @Identifier = @Node + ":" + @Manager + ":" + @AlertGroup + ":" + @AlertKey + ":" + @Type
}
else
{
##############################################
# Identifier
##############################################
@Identifier = $host + ":" + $status
##############################################
# Generic clear fields
##############################################
@Node        = $host
@NodeAlias   = $alias
@Manager     = %Manager
@AlertGroup  = "Ping"
@AlertKey    = $host
@Class       = 100
##############################################
# Start of Status Processing
##############################################
if ( !match("",previous_status_array[$host]) )
{
   $previous_status_array = previous_status_array[$host]
}
```

```
switch ($status)
{
####################
# Unreachable - PROBLEM
####################
  case "unreachable" :
    @Summary = @Node + " is not reachable"
    @Severity = 5
    @Type = 1
# Count unreachable and reset responded_array
    unreachable_array[$host] = int(unreachable_array[$host]) + 1
    responded_array[$host]   = 0
####################
# Alive - RESOLUTION
####################
  case "alive" :
    @Summary = @Node + " is now alive"
    @Severity = 3
    @Type = 2
# Reset unreachable counter
    unreachable_array[$host] = 0
####################
# NoAddress - INFO
####################
  case "noaddress" :
    @Severity = 2
    @Summary = @Node + " has no address"
####################
# Removed - CRITICAL
####################
  case "removed" :
    @Severity = 5
    @Summary = @Node + " has been removed"
# Reset unreachable counter
    unreachable_array[$host] = 0
####################
#  Slow - INFO
####################
  case "slow" :
    @Severity = 2
    @Summary = @Node + " has not responded within trip time"
# Reset counters
    unreachable_array[$host] = 0
    responded_array[$host]   = 0
####################
# NewHost - INFO
####################
  case "newhost" :
    @Severity = 2
    @Summary = @Node + " is a new host"
# Reset counters
    unreachable_array[$host] = 0
    responded_array[$host]   = 0
####################
# Responded - RESOLUTION
####################
  case "responded" :
    if( match($previous_status_array,"unreachable") ||
        int(responded_array[$host]) < int($NumberOfResponded) )
    {
        @Severity = 2
        @Type = 2
    }
    else
    {
        $discarded = "true"
        log(debug,"DEBUG: DISCARD: RESPONDED - " + @Identifier  )
        discard
    }
    @Summary = @Node + " has responded"
# Count responded
    responded_array[$host] = int(responded_array[$host]) + 1
# Reset unreachable counter
    unreachable_array[$host] = 0
# DEFAULT
    default :
    @Summary = "Ping Probe Error details : " + $*
    @Severity = 3
}
```

```
################################################
# Only interested in unreachables after successive failures
################################################
if ( int(unreachable_array[$host]) < int($NumberOfUnreachables)
     && match($status,"unreachable") )
{
   $discarded="true"
   discard
   log(info,"DEBUG: DISCARD: " + @Identifier  + " ( " + unreachable_array[$host] + " )" )
}
################################################
# Save current status for next time
################################################
   previous_status_array[$host] = $status


   if (!match($discarded,"true"))
   {
     log(info,"DEBUG: SEND: " + @Identifier  )
   }
}
#EOF
```

4.4 Custom rules file logic

Rather than storing separate events for each $status type, the probe can use just the latest status value for the event to provide a current service status event.

```
if( match( @Manager, "ProbeWatch" ) )
{
        switch(@Summary)
        {
        case "Running ...":
                @Severity = 1
                @AlertGroup = "probestat"
                @Type = 2
        case "Going Down ...":
                @Severity = 5
                @AlertGroup = "probestat"
                @Type = 1
        default:
                @Severity = 1
        }
        @AlertKey = @Agent
        @Summary = @Agent + " probe on " + @Node + ": " + @Summary
} else {
# Only allow one event per host
        @Identifier = $host
# Default settings
        @Node = $host
        @NodeAlias = $alias
        @Manager  = %Manager
        @AlertGroup = "Ping"
        @Class = 100

        switch ($status)
        {
                case "unreachable" :
                        @Severity = 5
                        @Summary = @Node + " is not reachable"
                case "alive" :
                        @Severity = 2
                        @Summary = @Node + " is now alive"
                case "noaddress" :
                        @Severity = 2
                        @Summary = @Node + " has no address"
                case "removed" :
                        @Severity = 5
                        @Summary = @Node + " has been removed"
                case "slow" :
                        @Severity = 2
                        @Summary = @Node + " has not responded within trip time"
                case "newhost" :
                        @Severity = 2
                        @Summary = @Node + " is a new host"
                case "responded" :
                        @Severity = 0
                        @Summary = @Node + " has responded"
                default :
                        @Summary = "Ping Probe Error details : " + $*
                        @Severity = 4
        }

# icmp_stats:   %PacketSize bytes from <ip-address>: icmp_seq=<incr>. ttl=<n> time=<n> ms
        if (exists($icmp_stats))
        {
          $response_time = extract ($icmp_stats, "time=([0-9]+) ms" )
          log(debug,"ResponseTime = " + $response_time)
        }

        #log(warn,"@Identifier = " + @Identifier )
        log(warn,"@Severity : @Summary = " + @Severity + " : " + @Summary )
}
```

5 Logging

For a responded host you expect to see logging like:

```
Pinging my.server.uk.ibm.com
Response from my.server.uk.ibm.com
[Event Processor] host:      my.server.uk.ibm.com
[Event Processor] alias:     my.server.uk.ibm.com
[Event Processor] ip_address:        <IPADDRESS>
[Event Processor] status:    responded
[Event Processor] icmp_stats:32 bytes from <IPADDRESS>: icmp_seq=3. ttl=255 time=0 ms
[Event Processor] poll_rate: 60
[Event Processor] trip_time: 2000
[Event Processor] timeout_time:      1000
```

Provided all of the other hosts are responding and the probes properties are set correctly for the number of hosts the hosts status: should be returned at the same time every poll.

For example:

```
11:40:58: Debug: D-UNK-000-000: [Event Processor] status:    responded
11:41:58: Debug: D-UNK-000-000: [Event Processor] status:    responded
11:42:58: Debug: D-UNK-000-000: [Event Processor] status:    responded
```

5.1 Unreachable

For an unreachable host you expect to see logging like:

```
Timed out: my.server.uk.ibm.com
Cannot reach my.server.uk.ibm.com
[Event Processor] host:      my.server.uk.ibm.com
[Event Processor] alias:     my.server.uk.ibm.com
[Event Processor] ip_address:        <IP ADDRESS>
[Event Processor] status:    unreachable
```

The start of the ping process is logged as:

```
Pinging my.server.uk.ibm.com
```

Therefore the timing between the start of the ping process and the results of unreachable, should be given by the probe properties:

- Retry
- TimeOut

With the logging timings being:

```
(Retry+1)*Timeout/1000 seconds
```

Therefore for property settings:

```
Retry                   : 10
TimeOut                 : 1000
```

The gap between the start and end of the Ping logging for the host should be 11 seconds.
With the next ping timer being set at the new results timing.
For example:

```
11:20:16: Debug: D-UNK-000-000: [Event Processor] host:      my.server.uk.ibm.com
11:21:28: Debug: D-UNK-000-000: [Event Processor] host:      my.server.uk.ibm.com
11:22:40: Debug: D-UNK-000-000: [Event Processor] host:      my.server.uk.ibm.com
```