Tivoli Netcool Supports Guide to the MTTrapd [SNMP] Probe

by

Jim Hutchinson

Document release: 2.4

Table of Contents

1Introduction....3

1.1Overview....3
Event processing....4

2Performance....5

2.1Name Resolution....5
2.2Replacing hostname calls....5
2.3Buffering....6
2.4SNMPv3 configuration....6
2.4.1SnmpConfigChangeDetectionInterval....6
2.5Measuring....7
2.6Monitoring....7
2.7Architectures....8
2.7.1Mixing Traps and INFORMS....8
2.7.2Mixing protocols....8
2.7.3Mixing SNMP versions....8
2.7.4Understanding SNMP data....8
2.7.5Example Architecture....9

3New features....10

3.1Heartbeating....10
3.1.1Heartbeat Property....10
3.1.2ProbeWatchHeartbeatInterval property....10
3.2SNMP engine facilities....11
3.2.1Configuring the command line interface....11
3.2.2New Commands....12
3.3Host Naming facilities....14

4Resilience....15

4.1Overview....15
4.2Failover and Failback....16
4.3Peer to Peer....17
4.3.1BeatThreshold and BeatInterval....17
4.3.2Single probe server....18

5Testing....19

5.1UDP Traps....19
5.2TCP Traps....20
5.3WireShark RAW TCP data....21
5.4WireShark SNMPv3 trap inspection....22
5.5SNMP v3 traps and informs....23
5.5.1Sending an SNMPv3 INFORM trap....23
5.5.2Sending an SNMPv3 trap....23
5.6More on SNMPv3 Traps....24
5.6.1Special characters in a passphrase....24
5.6.2Extended encryption support....24
5.7SNMP Flooding....25

6Troubleshooting....26

6.1Performance....26
6.1.1Name resolution....26
6.1.1.1Rules File Considerations....26
6.1.2SNMP INFORMS....26
6.1.3Object Server....26
6.2Logging dropped traps....27
6.3Missing EngineId's....27

6.4Missing traps....28
6.5SNMPv3 traps....29
6.5.1Additional SNMPv3 security....29
6.5.2Example Secure SNMPv3 configuration....30
6.5.2.1Example trap#1....30
6.5.2.2Example trap#2....30
6.5.2.3Example trap#3....30

7Useful scripts....31

7.1SendSNMPv1HB....31
7.2sendSNMPv2HB....31
7.3sendSNMPv3DES_noauthnopriv....32
7.4sendSNMPv3DES_authpriv....32

8MIB MANAGER....33

8.1Example usage....33

1 Introduction

1.1 Overview

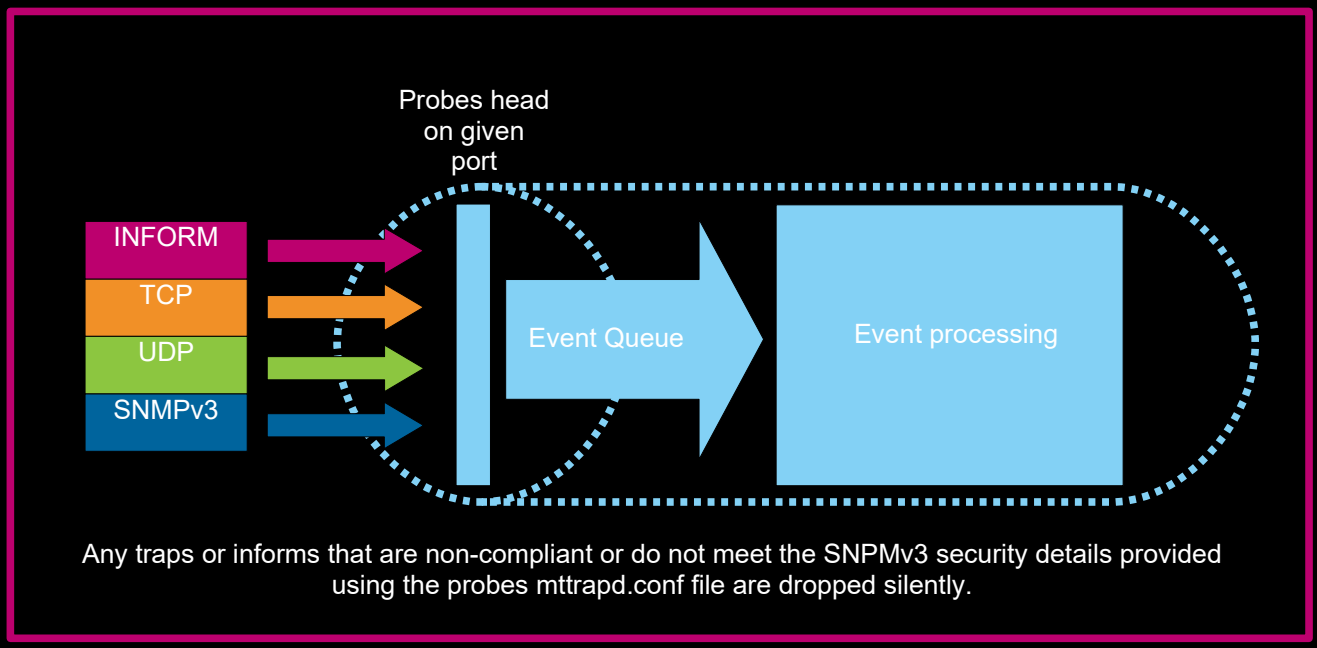
The MTTrapd (SNMP) probe is a Generic probe used to process SNMP traps. It is a multi-threaded probe and supports both UDP and TCP traps.

The SNMP versions supported are;

- SNMPv1 traps
- SNMPv2c traps
- SNMPv3 traps
- SNMPv2c informs
- SNMPv3 informs

Event processing

The Multi-threaded nature of the MTTrap probe receives traps from various source types, parses them using the Net-SNMP API and places the SNMP RFC Compliant traps on the probes queue for processing:



2 Performance

2.1 Name Resolution

Name resolution takes time, therefore turning it off improves performance;
UNIX

```
NoNameResolution : 1
```

Windows:

```
NoNetbiosLookups : 1
```

The latest version of the probe supports a new feature that allows DNS lookups to be managed via the probe property settings. This allows the DNS lookups to be performed.

For example

```
#
# Built-in Hostname cache
#
NoNameResolution : 0
HostnameTableSize : 50000
ActiveHostnameDuration : 30
RefreshHostnameInterval : 1440
```

These settings allow the probe to hold 50,000 lookups in memory, and refresh them every 24 hours.
Please refer to the probes manual to better understand these settings and their impact.

2.2 Replacing hostname calls

You can reduce the number of calls made to hostname and the probes hosts IP Address using a one time lookup and use static variables instead when the values are required.

```
if(match(%MyHostname,""))
{
        %MyHostname = hostname()
        %MyIPHostname = gethostaddr(%MyHostname)
}
@Manager = %Manager + "@" + %MyIPHostname + ":" + "0"

log(debug,"MYDEBUG: %MyHostname = " + %MyHostname)
log(debug,"MYDEBUG: %MyIPHostname = " + %MyIPHostname )
```

2.3 Buffering

Enabling buffering on probes allows a busy probe to send events in blocks, based on BufferSize, which is much more efficient for an object server that needs to manage more than a few clients. Ensure the FlushBufferInterval is set to a value that is suitable for the environment [Typically from 1-60];

e.g.

```
Buffering                   : 1
BufferSize                  : 200
FlushBufferInterval         : 9
```

2.4 SNMPv3 configuration

It is recommended to create a directory to hold the source SNMPv3 security configuration file per probe instance (ConfPath). This allows the source configuration file to be maintained, rather than overwritten. The ConfPath and PersistentDir should be secured as required using the operating systems file system permissions.

e.g.

```
mkdir -p $OMNIHOME/var/snmpv3/conf
vi $OMNIHOME/var/snmpv3/conf/mttrapd.conf
createUser -e 010203040506 trapuser  MD5 md5password DES despassword
:wq

vi $OMNIHOME/probes/Solaris2/mttrapd.props

ConfPath                : '$OMNIHOME/var/snmpv3/conf'
PersistentDir           : '$OMNIHOME/var/snmpv3'
:wq
```

2.4.1 SnmpConfigChangeDetectionInterval

The property SnmpConfigChangeDetectionInterval determines the periodicity of the checks on the ConfPath mttrapd.conf file. The default is to check every minute and see if there are new entries to parse. At message level informational the probe will log these checks.

- No updates

```
Config File is old and will not be parsed.
```

- ConfPath mttrapd.conf file was read

```
Processing config file completed.
User credentials have been refreshed.
```

With the default property settings you can update an existing engineid by first removing the entry from the ConfPath mttrapd.conf file, and waiting for the 'refreshed' message. After this the entry can be reinstated into the ConfPath mttrapd.conf, after which another 'refreshed' message is seen, and the new details added to the probes PersistentDir mttrapd.conf. This can be confirmed by checking the entry disappearing from the PersistentDir mttrapd.conf when it is commented out in the ConfPath mttrapd.conf file.

2.5 Measuring

The MTTrapd probes performance can be measured using rules file commands.
e.g.

```
if ( match(%load,"") )
{
        %load = "60.60"
}
else
{
        %load = updateload(%load)
        $current_load = getload(%load)
        log(INFO,"Average Events per second = " + $current_load)
}
```

Alternatively they can be calculated manually using maths functions.

```
e.g.
if ( match(%eps_counter,"") )
{
        %eps_counter = 1
        %start_time=getdate
}
else
{
        %end_time=getdate
        $time_elapsed = real(int(%end_time) - int(%start_time))

        if (int($time_elapsed) > 59 )
        {
                $current_load = real(%eps_counter) / real($time_elapsed)
                log(warn,"EPS: " + $current_load + " " + (%eps_counter) + " [ " + $time_elapsed + " ]")
                %eps_counter = 1
                %start_time=getdate
        }
        else
        {
                %eps_counter = int(%eps_counter) + 1
        }
}
```

If more complex load gathering is required, please refer to the Netcool/OMNIbus probes and gateway manual under ‘Enabling self monitoring of probes’.

2.6 Monitoring

The probe does allow specific features to be monitored using the probe property values.
Enable the property LogStatisticsInterval, by setting a time in seconds, to see the queue size and number of traps processed.
e.g.

```
LogStatisticsInterval : 60


Error: E-UNK-000-000: Trap queue size is 0
Error: E-UNK-000-000: Inform queue size is 0
Error: E-UNK-000-000: Number of traps read in the last 59 seconds: 0
Error: E-UNK-000-000: Number of traps processed in the last 59 seconds: 0
```

You can also enable the TrapStat property for further information logging at MessageLevel warning. More details monitoring is possible when the probe is logging at MessageLevel Debug.

2.7 Architectures

The MTTrapd probe is susceptible to trap sender problems, as it listens on an open for and queues events for processing on a single queue. When deploying the MTTrapd probe it is important to consider the volume of traps the probe is going to process, the traps sources performance, network congestion and vulnerability.

2.7.1 Mixing Traps and INFORMS

Using a single MTTrapd probe probe or P2P failover pair for all traps and informs is not recommended.

INFORMs require a response from the MTTrapd probe, which can cause delays in event processing. Therefore it is recommended that a dedicated probe or P2P failover pair is used for INFORMs, especially when the volume of SNMP senders is large.

2.7.2 Mixing protocols

Although the MTTrapd probe is capable of receiving both TCP and UDP at the same time, this may cause problems with event reception, under high loads. TCP is the best protocol to use, where event deliver needs to be guaranteed. TCP is best for SNMPv3 traps as UDP SNMPv3 traps will be discarded silently.

2.7.3 Mixing SNMP versions

SNMPv3 is a secure variant of SNMP with SNMP traps being the most secure. With this in mind, it makes sense to use a dedicated MTTrapd probe or P2P failover pair for SNMPv3, and configure the probes properties to ensure only SNMPv3 traps are received.

- Snmpv3ONLY
- snmpv3MinSecurityLevel

2.7.4 Understanding SNMP data

SNMP traps and INFORMs can be created periodically, during a system restart or shutdown, excessively due to misconfiguration, and contain large volumes of textual data. When designing the collection of SNMP data it is important to understand the impact of missing data, and how best to manage unusual circumstances.
When the MTTrapd probe is placed into debug mode, it logs all of the tokens for each event, as well as event processing details. During the planning phase of the MTTrapd probe deployment it is recommended that several hours of MTTrapd probe logs are collected from the network for analysis; Although it is understood that typically, this type of analysis will occur after deployment, and usually after problems have already occurred.

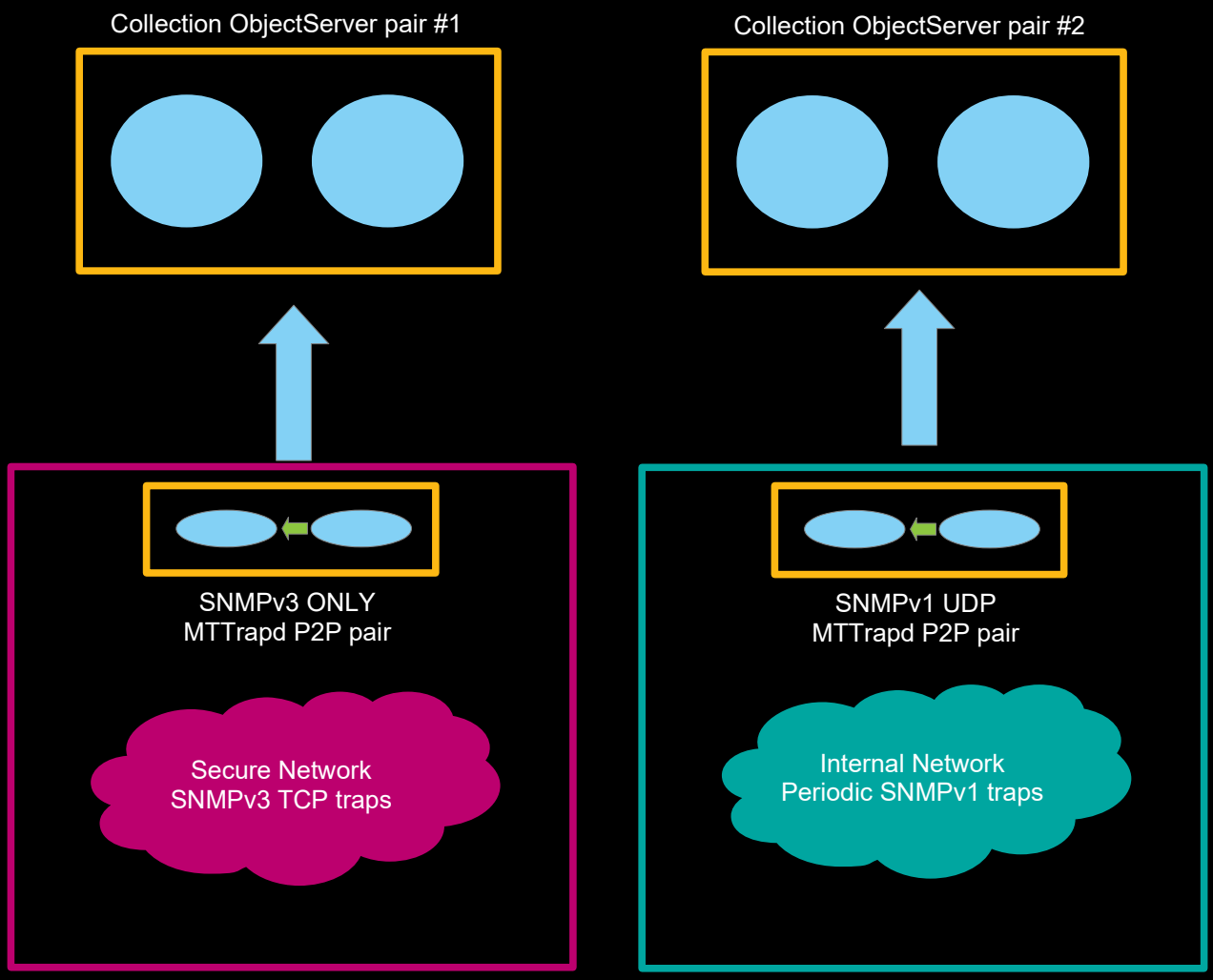
Ways of segregating SNMP data

- Periodic
- Event driven
- Data size
- Number of senders
- Number of traps

Before deciding how to segregate data, it is best to create a test environment and examine the impact of each type of SNMP data on the probe and collection object server.

2.7.5 Example Architecture

In the example given below, the customer identified that they were required to monitor two distinct networks, with each network sending two distinct types of traps. Due to the nature of the secure network, it was decided to implement a dedicated collection layer object server pair to prevent performance problems at the collection layer, and to allow for custom event handling. A firewall was configured on the two MTTrapd probe servers to prevent unauthorized IP addresses from sending data to the MTTrapd probes port. The internal network was found to be susceptible to SNMP trap flooding, and additional logic was added to the dedicated collection object server pair to prevent event floods from affecting the aggregation layer.



3 New features

This chapter outlines some of the new MTTrapd probe features available in the latest probe release.

3.1 Heartbeating

3.1.1 Heartbeat Property

The MTrapd probe could be disconnected from the target system if the connection between them becomes unavailable. The Heartbeat property periodically checks that the connection to the target Object Server, if it detects that the connection is unavailable, it shuts down.

3.1.2 ProbeWatchHeartbeatInterval property

The ProbeWatchHeartbeatInterval property is used by the Return On Investment self monitoring for the probes. It can be set to an interval in seconds:
e.g.

```
ProbeWatchHeartbeatInterval : 60
```

The Return On Investment self monitoring extension is part of the Netcool/OMNIbus extensions, and can be found in the following directory:

```
$NCHOME/omnibus/extensions/roi/

Omnibus_TDW_Reports_ROI.zip
probemanagement.sql
probestats.sql
probewatch.include
```

The ProbeWatchHeartbeatInterval property populates the %OplStats variables, which can be used outside of the ROI extension to log specific performance indicators.
e.g.

```
if ( int(%OplStatsNumberEvents) > 0 )
{
# Log OplStats every 100 events
 if ( ((int(%OplStatsNumberEvents)/100)*100) == int(%OplStatsNumberEvents) )
 {
# Calculate the Average usec timings
  $average_usec = int((real(%OplStatsRulesFileTimeSec) +
(real(%OplStatsRulesFileTimeUsec)/1000000) /  real(%OplStatsNumberEvents)) * 1000000)
# Log required details
  log(info,"%OplStatsRulesFileTime = " + %OplStatsRulesFileTimeSec + "." +
%OplStatsRulesFileTimeUsec)
  log(info,"%OplStatsNumberEvents  = " + %OplStatsNumberEvents)
  log(info,"Average USEC timing    = " + $average_usec + " usec.")
 }
}
```

3.2 SNMP engine facilities

Please refer to the product manuals for complete descriptions of how to enable process control and the new SNMPv3 engine features. The following is provided as an overview of the required actions.

In order to use the new facilities the following needs to be done:

Enable the property setting:

```
EngineInfoProbeWatch     : 1
```

Load the SQL file into the object servers:

```
cat mttrapd_create_SnmpActionTools.sql | nco_sql -server AGG_P -user root -password ''
```

Ensure that external actions are configured in the object servers running the procedures:

File : $NCHOME/omnibus/etc/AGG_P.props

```
PA.Name: 'NCO_PA'
PA.Password: 'CIEGAKHGBFGBBIGEDNBJCABFGEBCBKGE'
PA.Username: 'root'
```

Ensure the object server[s] are run under process control:

File : $NCHOME/omnibus/etc/NCO_PA.conf

```
Command '$OMNIHOME/bin/nco_objserv -name MTTRAPD -pa NCO_PA' run as 'nrv81'
```

Ensure that the PERL commands are installed on the object server[s] running the procedures

```
$OMNIHOME/probes/<platform>/mttrapd_Nhttp_SnmpActions.pl
```

Update the mttrapd.rules rules files to use the include files:

```
include "mttrapd.bidir.rules"
include "mttrapd.snmpwatch.rules"
```

3.2.1 Configuring the command line interface

In order to use the new SNMPV3 engine commands the command line interface needs to configured. The simplest way is to enable the HTTP interface as shown, using the probe properties:

```
# Command port
NHttpd.EnableHTTP           : TRUE
NHttpd.ListeningPort        : 12001
NHttpd.AccessLog            : "$OMNIHOME/log/mttrapd.nhttpd.access.log"
```

3.2.2 New Commands

Get the engine information for a specific engineid

NCO_HTTP command:

```
$OMNIHOME/bin/nco_http -uri http://localhost:12001/probe/common -datatype
application/json -data '{"eventfactory":
[{"snmp_action":"get_engine_info","engine_id":"00000000000000102030405"}]}' -method post
```

Example Debug: logging messages

```
$snmp_action -> get_engine_info
$engine_id -> 00000000000000102030405
>>> Enter mttrapd.bidir.rules. >>>
Received get_engine_info request
Calling get_engine_info
SNMP Message (priority=7): snmp_get_engine_info: Received engineID '00000000000000102030405'.
SNMP Message (priority=7): snmp_get_engine_info: Getting engine info of '00000000000000102030405' completes.
get_engine_info returns 'EngineID:00000000000000102030405  boots:0  time:0  host:'.
EngineID '00000000000000102030405' info: EngineID:00000000000000102030405  boots:0  time:0  host:
<<< Exit mttrapd.bidir.rules. <<<
```

Dump the probes list of engineid's to a file:

```
$OMNIHOME/bin/nco_http -uri http://localhost:12001/probe/common -datatype
application/json -data '{"eventfactory": [{"snmp_action":"print_engines",
"file_path":"/tmp/engineids.txt"}]}' -method post
```

Example Debug: logging messages:

```
$snmp_action -> print_engines
$file_path -> /tmp/engines.txt
Flushing events to object servers
0 buffered alerts
>>> Enter mttrapd.bidir.rules. >>>
Received print_engines request
Calling print_engines
SNMP Message (priority=7): snmp_print_engines: Received filename '/tmp/engines.txt'.
SNMP Message (priority=7): snmp_print_engines completes.
print_engines operation completes.
RULES>> print_engines request successful
<<< Exit mttrapd.bidir.rules. <<<
```

Example contents:
File : /tmp/engines.txt

```
engineid:00000000000000102030405  boots:0  time:0  host:
engineid:80001F8880A862C66948DEFC5D00000000  boots:1  time:0  host:
```

Setting specific parameters for a given engineid:

```
$OMNIHOME/bin/nco_http -uri http://localhost:12001/probe/common -datatype
application/json -data '{"eventfactory":
[{"snmp_action":"update_engine","engine_id":"00000000000000102030405",
"engine_boot":"","engine_time":"", "engine_host":"", "send_probewatch":""}]}' -method
post
```

Example Debug: logging mesages:

```
$snmp_action -> update_engine
$engine_id -> 000000000000000102030405
$engine_boot ->
$engine_time ->
$engine_host ->
$send_probewatch ->
Flushing events to object servers
0 buffered alerts
>>> Enter mttrapd.bidir.rules. >>>
Received update_engine request
Calling update_engine
2019-12-20T07:36:49: Information: I-UNK-000-000: SNMP Message (priority=6):
snmp_update_engine: Received engineID '000000000000000102030405' eboot_update (0)
eboot_time(0) ipaddr ''.
2019-12-20T07:36:49: Information: I-UNK-000-000: SNMP Message (priority=6):
snmp_update_engine: Updating engine '000000000000000102030405' completes.
update_engine operation completes.
RULES>> update_engine request successful
<<< Exit mttrapd.bidir.rules. <<<
```

3.3 Host Naming facilities

The host lookups have always been an issue with the MTTrapd probe. A new feature to manage this performance affecting facility is the internal hostname hash table. These features are enabled when the NoNameResolution property is set to 0. Note that these new features have their limitations, and in general, it is best to disable the naming resolution and use the probes rules files logic instead.

NoNameResolution : 0

When enabled, the probe writes resolved host names and discarded host names to flat files in the $NCHOME/omnibus/var directory.

Related properties:

- ActiveHostnameDuration
- RefreshHostnameInterval
- HostnameTableSize

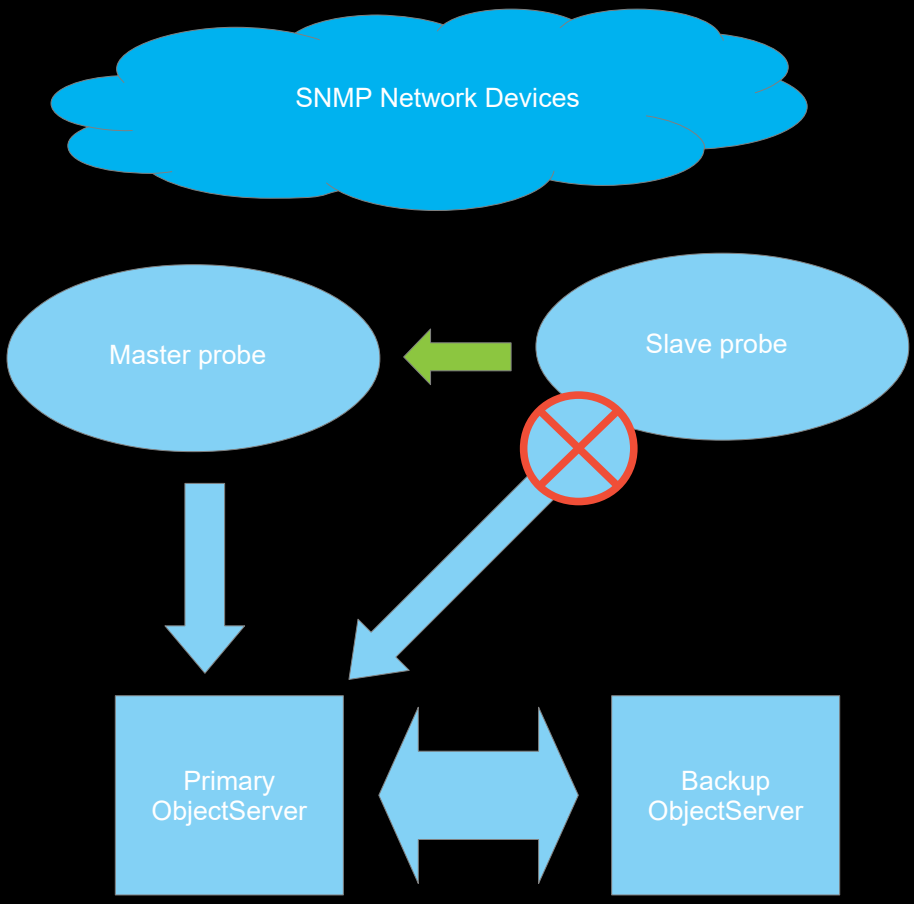
When the host name table has reached its maximum size the probe will stop parsing the active list at the list entry where it detects that the table is full.
The probe will not perform any instantaneous host name resolution for any traps received from new hosts and the IP Address will not be stored in the hash hosts table. The event tokens will use numeric IP Addresses only.

Reading from the IP host flat file
When the probe starts up, it reads from the stored hosts in the file named:
`$OMNIHOME/var/instance_active_iphost.list`
Where *instance* identifies the probe instance uniquely.

4 Resilience

4.1 Overview

With Peer to Peer failover (P2P), two probes are used with the events from the slave probe being discarded. Should the master probe become unavailable, the slave probe begins to forward its SNMP events to the Object Server. The configuration can be a dual site, or an dual resilient object server pair. The two probes can be configured with identical Server and ServerBackup settings to allow the SNMP events to be sent to the same Object Server, if this is required.



4.2 Failover and Failback

These are the common features that allow the MTTrapd probe to failover to a specified Object Server and then failback to the Primary Object Server;

```
Server                          : 'NCOMS_P'
ServerBackup                    : 'NCOMS_B'
NetworkTimeout                  : 30
PollServer                      : 60
```

On the back-up object server the property;

```
BackupObjectServer: TRUE
```

Must be set to allow failback.

The PollServer property must be larger than NetworkTimeout.
i.e.

```
PollServer = 2 * NetworkTimeout
```

4.3 Peer to Peer

When configuring the MTTrapd probe for failover the PeerHost should be different.

Example configuration:
MASTER HOST Properties file on master_host:

```
PeerHost          : 'slave_host'
PeerPort          : 6789
Mode              : 'master'
BeatInterval      : 11
BeatThreshold     : 3
Port              : 162
```

SLAVE HOST Properties file on slave_host:

```
PeerHost          : 'master_host'
PeerPort          : 6789
Mode              : 'slave'
BeatInterval      : 11
BeatThreshold     : 3
Port              : 162
```

4.3.1 BeatThreshold and BeatInterval

The BeatThreshold and BeatInterval are used to manage the timings of the heartbeats from the slave probe to the master probe. The BeatInterval must always be larger than the BeatThreshold, with the BeatThreshold adjusted to allow probe time to respond to heartbeat requests. The probe may not be able to respond to heartbeat requests if it is extremely busy, or CPU starved.
e.g.

```
BeatInterval  : 60
BeatThreshold : 5
```

4.3.2 Single probe server

It is possible to run two MTTrapd probes on the same host in failover mode. In order to do this you would need to use different trap ports

e.g.

MASTER HOST Properties file on myhost:

```
PeerHost            : 'myhost'
PeerPort            : 6789
Mode                : 'master'
BeatInterval        : 11
BeatThreshold       : 3
MessageLevel        : 'debug'
MessageLog          : '/opt/Omnibus/log/mttrapd_master.log'
Port                : 162
```

SLAVE HOST Properties file on myhost:

```
PeerHost            : 'myhost'
PeerPort            : 6789
Mode                : 'slave'
BeatInterval        : 3
MessageLevel        : 'debug'
MessageLog          : '/opt/Omnibus/log/mttrapd_slave.log'
Port                : 1620
```

5 Testing

5.1 UDP Traps

Sending a single trap to the localhost on Solaris;

```
/usr/sbin/snmp_trapsend -g 6 -s 12 -a ".1.3.6.1.4.1.42.1.1.2 STRING (Testing the trap utility)"
```

Sending a periodic trap;

```
/bin/sh
while true
do
/usr/sbin/snmp_trapsend -g 6 -s 12 -a ".1.3.6.1.4.1.42.1.1.2 STRING (Testing the trap utility)"
sleep 10
done
```

Sending a trap storm;

```
/bin/sh
while true
do
/usr/sbin/snmp_trapsend -g 6 -s 12 -a ".1.3.6.1.4.1.42.1.1.2 STRING (Testing the trap utility)"
done
```

5.2 TCP Traps

The SNMP gateway supports the sending of TCP SNMP traps and so can be used alongside a test object server being fed by a simnet probe to generate large amounts of TCP SNMP traps.

Example configuration;

```
CREATE MAPPING SNMP_MAP
(
        0 = '@Summary',
        1 = '@Severity',
        2 = '@Location',
        3 = '@Node',
        4 = '@AlertGroup'
);

# Start up the reader - connect to the Object Server NCOMS
START READER NCOMS_READER CONNECT TO NCOMS1;
# Start up the writer
START WRITER SNMP_WRITER
(
        TYPE = SNMP,
        REVISION = 1,
        GATEWAY = 'snmp-server',
        PORT = 1620,
        PROTOCOL = 'TCP',
        MAP = SNMP_MAP
);

ADD ROUTE FROM NCOMS_READER TO SNMP_WRITER;
```

5.3 WireShark RAW TCP data

The WireShark program can be used to export an actual trap for replication of specific issues.

http://www.wireshark.org/

To extract the Trap[s]:

- Load snoop/ethereal into the GUI
- Select SNMP packet to export at [+] SNMP
- File-> Export Selected Packet Bytes-> Filename [trap#.bin]

To replay the trap[s]:

- Start the MTTrapd probe

```
$OMNIHOME/probes/nco_p_mttrapd -all -port port
```

- Cat the binary trap file to the probes port using netcat

```
cat trap#.bin | nc localhost port
```

IMPORTANT : For SNMPv3 it is not recommended to replay traps out of sequence or repeatedly, as this can break the SNMPv3 security. If traps need to be replayed more than once, the MTTrapd probe should be restarted before the traps are replayed.

5.4 WireShark SNMPv3 trap inspection

WireShark allows SNMPv3 traps to be inspected through its Graphical User Interface.

Select the trap you would like to inspect, in the trap you can select an item such as the engineid and copy the value using the right menu, or the keys control-shift-V.
e.g.

```
snmp.msgAuthoritativeEngineID == 00:11:22:33:44:55:66:77:88:99:00:11:22:33:44:55:66:77:88:99:00
```

This can be used to set the engineid in the mttrapd.conf file.
e.g.

```
createUser -e 0011223344556677889900112233445566778899000 myusername SHA mySHApassphrase DES myDESpassphrase
```

You can check the contents of the SNMPv3 trap by selecting the trap and opening the right menu:
Protocol preferences → Open Simple Network Management Protocol Preferences
Pop-Up → User Table [Edit]
Pop-Up → +
Engine ID, Username, MD5|SHA1, passphrase, DES|AES, passphrase
[ok]
[ok]

If successful, the GUI will show the contents of the encrypted trap.
If there is a problem, check the SNMPv3 details again.

5.5 SNMP v3 traps and informs

NET-SNMP is a free software package which allows the sending of traps and informs using snmptrap. It is available from these websites:

http://www.net-snmp.org/

Solaris download : http://sunfreeware.blueyonder.co.uk

- Start the mttrapd probe from the command line in debug mode when testing the probe

```
$OMNIHOME/probe/nco_p_mttrapd -messagelevel debug -messagelog stdout -all
```

- From another host send the traps, to test send an SNMPv1 trap

SNMPv1 trap;

```
snmptrap -v 1 -c public PROBEHOST "" "" 6 99 ""e
```

The **PROBEHOST** can be set to have the protocol and port set.

e.g. TCP and 1620 on host 192.168.20.18 would be;

```
TCP:192.168.20.18:1620
```

5.5.1 Sending an SNMPv3 INFORM trap

Add the following line to the $OMNIHOME/var/mttrapd.conf file and restart the probe;

```
createUser jack MD5 jackjill
```

```
snmptrap -v 3 -C i -a MD5 -x DES  -l noAuthNoPriv -u jack -A "jackjill" -X "jackjill" PROBEHOST "" .
1.3.6.1.4.1.2021.251.1
```

INFORM traps reply to the snmptrap application which allows the user/pass pairs to be tested, once they have been added to the mttrapd.conf file.

5.5.2 Sending an SNMPv3 trap

For normal SNMPv3 traps specify the **ENGINE ID** in the both the snmptrap and mttrapd.conf file;

mttrapd.conf :

```
createUser -e 800000010203040506 jack MD5 jackjill
```

To send the trap use;

```
snmptrap -v 3 -e 0x800000010203040506 -a MD5 -x DES -l noAuthNoPriv -u jack -A "jackjill" -X "jackjill"
PROBEHOST "" .1.3.6.1.4.1.2021.251.1
```

5.6 More on SNMPv3 Traps

Common questions arise when creating the mttrapd.conf file.

5.6.1 Special characters in a passphrase

The MTTrapd probe

File : mttrapd.conf

```
createUser -e 8000000000000000000000001 someuser SHA PASSW0RD@123 AES PASSW0RD@123
createUser -e 8000000000000000000000002 SomeUSER SHA PASSW0RD@123 AES PASSW0RD@123
createUser -e 8000000000000000000000003 Some_USER SHA PASSW0RD@123 AES PASSW0RD@123
```

Sending traps:

```
snmptrap -M $NETSNMP/share/snmp/mibs -e 8000000000000000000000001 -v3 -u someuser -a
SHA -A PASSW0RD@123  -x AES -X PASSW0RD@123 -l authPriv  TCP:localhost:1620  "" IF-
MIB::linkUp IF-MIB::ifAlias s "alias:123"
$ snmptrap -M $NETSNMP/share/snmp/mibs -e 8000000000000000000000002  -v3 -u SomeUSER
-a SHA -A PASSW0RD@123  -x AES -X PASSW0RD@123 -l authPriv  TCP:localhost:1620  ""
IF-MIB::linkUp IF-MIB::ifAlias s "alias:123"
$ snmptrap -M $NETSNMP/share/snmp/mibs -e 8000000000000000000000003 -v3 -u Some_USER
-a SHA -A PASSW0RD@123  -x AES -X PASSW0RD@123 -l authPriv  TCP:localhost:1620  ""
IF-MIB::linkUp IF-MIB::ifAlias s "alias:123"
```

5.6.2 Extended encryption support

The MTTrapd probe now supports extended encryption types:

```
authtype    MD5, SHA, or SHA256
privtype    DES, AES, AES192 or AES256
```

However, the Net-SNMP snmptrap command only supports privtype DES and AES, so cannot be used to test extended privtype encryption types, even though these can be set and used in the mttrapd.conf file.

```
-a PROTOCOL   set authentication protocol (MD5|SHA|SHA-224|SHA-256|SHA-384|SHA-512)
-x PROTOCOL   set privacy protocol (DES|AES)
```

File : mttrapd.conf

```
createUser -e 8000000000000000000000010 SHA256User1234 SHA256 PASSW0RD123567890 AES PASSW0RD123567890
```

sending the trap:

```
snmptrap -M $NETSNMP/share/snmp/mibs -e 8000000000000000000000010 -v3 -u
SHA256User1234 -a SHA256 -A PASSW0RD123567890 -x AES -X PASSW0RD123567890 -l authPriv
TCP:localhost:1620  "" IF-MIB::linkUp IF-MIB::ifAlias s "alias:123"
```

5.7 SNMP Flooding

The Netcool/SSM agent is excellent at sending vast quantities of traps and can be used to test loading of the MTTrapd probe.

e.g.

```
cd /opt/netcool/ssm/config
vi monitors.cfg
…
#
process attr=averageCpu
process filtertype=name                filter=".*"
process interval=1                     sampletype=delta
process oper=ge                        thresh=0
process actioneventstatus=alwaysReady  actionevent=$psevent
process create
…
:wq

cd /opt/netcool/ssm/bin
./init.ssmagent stop
./init.ssmagent start
./ssmcons
trapdest add <IP ADDRESS>:<PORT> public
trapdest add <IP ADDRESS>:<PORT> public
trapdest add <IP ADDRESS>:<PORT> public
trapdest add <IP ADDRESS>:<PORT> public
```

6 Troubleshooting

6.1 Performance

The performance of the SNMP probe is most affected by name resolution.

6.1.1 Name resolution

The following methods for obtaining hostnames/ip addresses are available to the operating system:

Files [/etc/hosts]

DNS

NIS

If the property 'NoNameResolution' is set to '1' and there still appears to be a performance issue at the probe, check the probe rules files for the commands:

- gethostname
- gethostaddr

As these impede the probes performance with respect to name resolution.

Adding hosts to the /etc/hosts file can improve performance, where DNS is slow.

Use DNS caching when DNS is required and if the /etc/hosts file begins to exceed few hundred lines, as DNS/NIS is a memory resident lookup, which is significantly faster than using files.

6.1.1.1 Rules File Considerations

With the NoNameResolution set to 1, the probe will not use naming resolution, which improves the probes overall performance. However, the rules files may include repetitive host naming lookups, such as looking up the probe servers hostname and IP Address. The way to reduce this load is to use static variables as shown:

```
if(match(%MyHostname,""))
{
        %MyHostname = hostname()
        %MyIPHostname = gethostaddr(%MyHostname)
}
@Manager = %Manager + "@" + %MyIPHostname
```

6.1.2 SNMP INFORMS

SNMP INFORMS require two way communication between the probe and INFORM sender. The MTTrapd probes uses a separate head for TCP/UDP INFORMS, however, the probe still needs to process each INFORM, and it should be expected that INFORMS are at least 50% slower than the equivalent TRAPS.

6.1.3 Object Server

The object server's performance can affect how the probe performs if the object server is continually unresponsive, causing the probe to switch to Store And Forward or event failover to the backup object server. Therefore it is important to check that the object server is able to manage the load from the MTTrapd probe under peak loading, by checking its profiler and trigger statistics.

6.2 Logging dropped traps

The MTTrapd probe includes a method to log what traps dropped, when the trap queue is full, so as to allow further troubleshooting and diagnosis.

DSALog

Use this property to specify whether the probe logs traps that are lost because the trap queue has become full [1]

DSAPeriod

Use this property to specify the time, in seconds, that traps are logged when the DSALog property has a value of 1 [default is 30 seconds].

6.3 Missing EngineId's

Sometime the EngineId's of the devices sending the traps is unknown due to legacy issues. SNMPv3 traps need a engineId to be given in the mttrapd.conf file for them to be read by the probe. If the username and password is known, then TSHARK [part of the Wireshark package] can be used to collect the engineId's arriving at the probes port.

For example:

```
tshark -V -i eth0 -d tcp.port==1620,snmp | grep -i msgAuthoritativeEngineID:
```

Will show lines with msgAuthoritativeEngineID: string for all the TCP SNMP traps on port 1620.

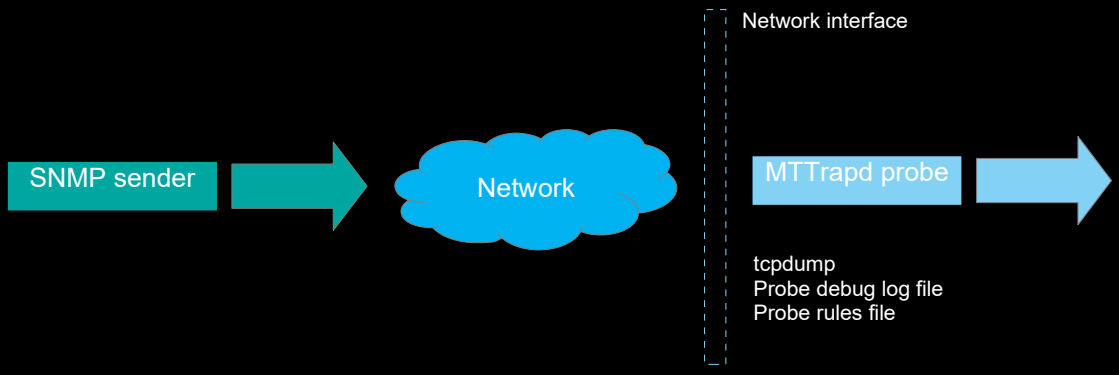
```
msgAuthoritativeEngineID: 010203040506
```

These can then be used in the mttrapd.conf file to create valid createUser entries.

```
 createUser -e 010203040506 trapuser  MD5 md5password DES despassword
```

6.4 Missing traps

Traps can be lost anywhere within the system, from the trap source to the object server. Therefore it is important to identify exactly where the traps are lost.



Most systems allow tcpdump to be installed and can be used to confirm that the traps are arriving at the probe server:

```
tcpdump -s 0 -w /tmp/tcpdump_ems.pcap host snmp_sender_host
```

Use the resulting data to check and confirm that all the traps arrive and have the right details, such as port, protocol, SNMP settings, etc.
The probes debug log will include all the tokens for the traps that the probe is able to read. Traps that cannot be read will be discarded silently. If the probes debug log does not include a trap seen in the tcpdump data, then there are a number of reasons why this can happen. For SNMPv3 there is a separate section.

For SNMPv1 and SNMPv2c traps:

1. Traps must include uptime
2. The correct protocol must be set for the probe [udp/tcp]

If all the traps are seen in the debug log file, then the problem is with event processing rather than trap reception. The probe supports a raw capture mode that allows events to be saved to a file and replayed to the Standard Input probe in a test environment.

If the debug log file cannot be obtained it is possible to log specific probe tokens as required. For example:

log(warn,"SNMPV3Check:" + $PeerIPaddress + ":" + $IPaddress)

6.5 SNMPv3 traps

SNMPv3 traps have the additional security requirements placed on them. These are detailed within the RFC's regarding SNMPv3. From the MTTrapd probes perspective, SNMPv3 traps must adhere to the RFC requirements as enforced by the NET-SNMP library. Typically, the MTTrapd probe will drop non-compliant traps silently, which can cause administration problems as it makes troubleshooting difficult. The reason why the probe does not perform any analysis of such traps is due to the way the traps are processed, within the NET-SNMP libraries, and the overall requirement to ensure high performance.

Typically traps will be dropped if they are malformed, inconsistent, or time delays past the allowed SNMPv3 tolerance window. For example, if SNMPv3 traps are sent from two or more sources with the same engineID, one or more could be discarded depending upon how these traps present themselves to the probe. It is therefore important to ensure that each element uses a unique engineID to prevent traps from being dropped.

Equally, it is important to administer the network such that only authorised elements are able to send traps to the MTTrapd probe server, to prevent denial of service.

IMPORTANT : The PersistentDir mttrapd.conf file is appended to rather than overwritten. Therefore it should be kept up to date, with unique entries, to prevent security issues. Whilst remembering that the probe will require time to create the PersistentDir mttrapd.conf file from the ConfPath mttrapd.conf file, if the PersistentDir mttrapd.conf file is deleted.

6.5.1 Additional SNMPv3 security

The MTTrapd probe has two SNMPv3 specific properties to ensure that the probe is kept secure:

- snmpv3ONLY
- snmpv3MinSecurityLevel

Setting the probe to use only SNMPv3 traps at a specific security level, allows other traps to be ignored and prevent denial of service attacks.

For example, the highest security setting is:
snmpv3ONLY : 1
snmpv3MinSecurityLevel : 3

With snmpv3ONLY set to '1' the probe logs:

```
Dropping the trap since it is v1/v2c trap
```

With snmpv3MinSecurityLevel set to '2' or '3' the probe logs:

```
Dropping V3 traps/informs because it does not match required SNMPv3 security level
```

6.5.2 Example Secure SNMPv3 configuration

mttrapd.conf:

```
createUser -e 800000020109840311 shauser SHA shapassword DES despassword
```

6.5.2.1 Example trap#1

Trap sent:

```
snmptrap -e 0x800000020109840311 -v3 -u shauser -l NoauthNoPriv UDP:IPADDRESS:1621 0 coldStart.0
```

Working probe properties:
snmpv3ONLY : 1
snmpv3MinSecurityLevel : 1
Log Message: Trap is processed

Blocked probe properties:
snmpv3ONLY : 1
snmpv3MinSecurityLevel : 2

Log message:
Warning: W-UNK-000-000: Dropping V3 traps/informs because it does not match required SNMPv3 security level, configured level = 2, pdu level = 1

6.5.2.2 Example trap#2

Trap sent:

```
snmptrap -e 0x800000020109840311 -v3 -u shauser -a SHA -A shapassword -l authNoPriv UDP:IPADDRESS:1621 0
coldStart.0
```

Working probe properties:
snmpv3ONLY : 1
snmpv3MinSecurityLevel : 2
Log Message: Trap is processed

Blocked probe properties:
snmpv3ONLY : 1
snmpv3MinSecurityLevel : 3

Log message:

```
Warning: W-UNK-000-000: Dropping V3 traps/informs because it does not match required SNMPv3 security level,
configured level = 3, pdu level = 2
```

6.5.2.3 Example trap#3

Trap sent:

```
snmptrap -e 0x800000020109840311 -v3 -u shauser -a SHA -A shapassword -x DES -X despassword -l authPriv
UDP:IPADDRESS:1621  0 coldStart.0
```

Working probe properties:
snmpv3ONLY : 1
snmpv3MinSecurityLevel : 3
Log Message: Trap is processed

7 Useful scripts

7.1 SendSNMPv1HB

```
#! /bin/sh

if [ $# -ne 2 ]
then
        echo "Usage : $0 [host] [port]"
        exit
fi

export counter
counter=1

while true
do
date

snmptrap -v 1 -c public TCP:$1:$2 .1.3.6.1.4.45 localhost 2 0 '0'  .1.3.6.1.4.45.32 s
"Test#${count} trap" &

counter=`expr $counter + 1`
sleep 60
done
#EOF
```

7.2 sendSNMPv2HB

```
#! /bin/sh

if [ $# -ne 2 ]
then
        echo "Usage : $0 [host] [port]"
        exit
fi
export counter
counter=1

while true
do
date
snmptrap -v 2c -c public TCP:${1}:${2} "" NET-SNMP-EXAMPLES-
MIB::netSnmpExampleHeartbeatNotification netSnmpExampleHeartbeatRate i $counter

counter=`expr $counter + 1`
sleep 10
done
#EOF
```

7.3 sendSNMPv3DES_noauthnopriv

```
#! /bin/sh
# mttrapd.conf entry:
# createUser -e 010203040506 trapuser  MD5 md5password DES despassword
#
export count HOST PORT NUMBER
count=1

if [ $# -ne 3 ]
then
        echo "Usage : $0 [host] [port] [number of traps]"
        exit
fi

HOST=$1
PORT=$2
NUMBER=$3

while [ $count -le $NUMBER ]
do
snmptrap -e 0x010203040506 -v3 -u trapuser TCP:${HOST}:${PORT} "" coldStart.0
count=`expr $count + 1`
done
#EOF
```

7.4 sendSNMPv3DES_authpriv

```
#! /bin/sh
# mttrapd.conf entry:
# createUser -e 010203040506 trapuser  MD5 md5password DES despassword
#
export count HOST PORT NUMBER
count=1

if [ $# -ne 3 ]
then
        echo "Usage : $0 [host] [port] [number of traps]"
        exit
fi

HOST=$1
PORT=$2
NUMBER=$3

while [ $count -le $NUMBER ]
do
snmptrap -e 0x010203040506 -v3 -u trapuser -a MD5 -A md5password -x DES -X despassword -l
authPriv TCP:${HOST}:${PORT} "" coldStart.0
count=`expr $count + 1`
done
#EOF
```

8 MIB MANAGER

8.1 Example usage

The files exported by MIB MANAGER include a README.txt that explains how to merge and use the exported rules files.

This is an example merge of some IBM MIBs for NcKL 4.2:

To create the NcKL rules file for the given MIBs:

- IMPORT MIBS into MIB Manager

Check they are all imported as expected using the pop-up and GUI.

Note: Usually it is easiest to remove all the mibs from MIB MANAGER, then import the base mibs, and then import the custom mibs.

- EXPORT MIBS as NcKL 3.0

This export creates a file in the chosen export directory like:
`<date&time>-nckl_3_0`

- Zip up the directory for copying to the probe server:

`<date&time>-nckl_3_0.zip`

Review the README.txt file:

```
MIB Manager NCKL Format Rulesfiles
------------------------------
Instructions For Use:

The rulesfile was designed for use with the NCKL 3.x format rules.
The current IBM recommended and supported rulesfile format
is NCKL. It is highly recommended all users make use of the NCKL
format if at all possible.

To use this NCKL format rulesfile simply place includes for the two
per vendor master files into your snmptrap.rules file.
Those files are named:
<vendor>/<vendor>.m2r.master.include.lookup
<vendor>/<vendor>.m2r.master.include.rules

If there already exists vendor specific NCKL rules for the vendor
then simply include the above two m2r.master files in

<vendor>/<vendor>.master.include.lookup
<vendor>/<vendor>.master.include.rules

and in this case the <vendor>/<vendor>-preclass.snmptrap.lookup
file should be merged into the existing preclass lookup file.
-------------------------------------------------------------------------
```

To install on the new rules files on probe server:

- Unpack the zip file in a temporary directory

e.g.

```
unzip <date&time>-nckl_3_0.zip
cd <date&time>-nckl_3_0
ls -R1 ibm
ibm:
ibm-IBM-3200-MIB_eventBrowserLogin.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventBrowserLogin.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventBrowserLogout.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventBrowserLogout.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventDoorOpen.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventDoorOpen.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventDriveError.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventDriveError.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventFaultPosted.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventFaultPosted.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventLoaderConfigChange.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventLoaderConfigChange.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventLoaderOK.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventLoaderOK.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventLoaderPasswordChange.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventLoaderPasswordChange.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventLoaderRetriesExcessive.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventLoaderRetriesExcessive.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventMailSlotAccessed.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventMailSlotAccessed.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventRequestDriveClean.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventRequestDriveClean.user.include.snmptrap.rules
ibm-IBM-3200-MIB_eventStatusChange.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_eventStatusChange.user.include.snmptrap.rules
ibm-IBM-3200-MIB_ibm3200Event.adv.include.snmptrap.rules
ibm-IBM-3200-MIB_ibm3200Event.user.include.snmptrap.rules
ibm-IBM-3200-MIB.include.snmptrap.lookup
ibm-IBM-3200-MIB.include.snmptrap.rules
ibm-IBM-3200-MIB.sev.snmptrap.lookup
ibm-IBM-ENETDISPATCHER-MIB.adv.include.snmptrap.rules
ibm-IBM-ENETDISPATCHER-MIB.include.snmptrap.lookup
ibm-IBM-ENETDISPATCHER-MIB.include.snmptrap.rules
ibm-IBM-ENETDISPATCHER-MIB.sev.snmptrap.lookup
ibm-IBM-ENETDISPATCHER-MIB.user.include.snmptrap.rules
ibm-IBM2210-MIB.adv.include.snmptrap.rules
ibm-IBM2210-MIB.include.snmptrap.rules
ibm-IBM2210-MIB.sev.snmptrap.lookup
ibm-IBM2210-MIB.user.include.snmptrap.rules
ibm-IBMIROC-MIB_ibmIROCtrapsels.adv.include.snmptrap.rules
ibm-IBMIROC-MIB_ibmIROCtrapsels.user.include.snmptrap.rules
ibm-IBMIROC-MIB_ibmIROCtrapsfr.adv.include.snmptrap.rules
ibm-IBMIROC-MIB_ibmIROCtrapsfr.user.include.snmptrap.rules
ibm-IBMIROC-MIB_ibmIROCtrapssys.adv.include.snmptrap.rules
ibm-IBMIROC-MIB_ibmIROCtrapssys.user.include.snmptrap.rules
ibm-IBMIROC-MIB.include.snmptrap.rules
ibm-IBMIROC-MIB.sev.snmptrap.lookup
ibm-IBMTCPIPMVS-MIB.adv.include.snmptrap.rules
ibm-IBMTCPIPMVS-MIB.include.snmptrap.rules
ibm-IBMTCPIPMVS-MIB.sev.snmptrap.lookup
ibm-IBMTCPIPMVS-MIB.user.include.snmptrap.rules
ibm-IPSECV1-MIB.adv.include.snmptrap.rules
ibm-IPSECV1-MIB.include.snmptrap.rules
ibm-IPSECV1-MIB.sev.snmptrap.lookup
ibm-IPSECV1-MIB.user.include.snmptrap.rules
ibm-L2TV1-MIB.adv.include.snmptrap.rules
ibm-L2TV1-MIB.include.snmptrap.lookup
ibm-L2TV1-MIB.include.snmptrap.rules
ibm-L2TV1-MIB.sev.snmptrap.lookup
ibm-L2TV1-MIB.user.include.snmptrap.rules
ibm-preclass.include.snmptrap.rules
ibm-preclass.snmptrap.lookup
ibm.m2r.master.include.lookup
ibm.m2r.master.include.rules
```

- Interactively copy the files to the vendor specific directory

```
cp -i * /opt/NcKL_42/rules/include-snmptrap/ibm
```

- Check and compare the pre-class files

Do not copy any files that already exist and check the differences:
check differences

```
diff /opt/NcKL_42/rules/include-snmptrap/ibm/ibm-preclass.include.snmptrap.rules ibm-
preclass.include.snmptrap.rules
diff /opt/NcKL_42/rules/include-snmptrap/ibm/ibm-preclass.snmptrap.lookup ibm-preclass.snmptrap.lookup
```

- Add in the master files

```
cd /opt/NcKL_42/rules/include-snmptrap/ibm
vim ibm.master.include.lookup

# NcKL lookups
include "$NC_RULES_HOME/include-snmptrap/ibm/ibm.m2r.master.include.lookup"
#EOF

vim ibm.master.include.rules
# NcKL Includes
include "$NC_RULES_HOME/include-snmptrap/ibm/ibm.m2r.master.include.rules"
#EOF
```

- Check the rules files syntax using nco_p_syntax

```
$OMNIHOME/probes/nco_p_syntax -rulesfile $NC_RULES_HOME/snmptrap.rules -server NCKL4
-messagelevel warn
```

- Make any required modifications

e.g.

```
vi ibm.m2r.master.include.lookup
#include "$NC_RULES_HOME/include-snmptrap/ibm/ibm-preclass.snmptrap.lookup"
```

The rules files can then be used with the working mttrapd probe.

Notes:
You should always perform rules file work after making a backup of the rules files, and preferably check the edits on a test system, before deployment.
Once installed, the MIB Manager, loaded with the vendor MIBs can be used to send test traps to the MTTrapd probe to confirm the rules files process the traps as required.