Tivoli Netcool Support's
Guide to
Process Automation Configuration
by
Jim Hutchinson
Document release: 2.2

Table of Contents

1 Introduction

1.1 Scope

This document is limited to the UNIX variant of Process Control. Although the nco_pad binary is available for Windows, there are a number of differences which complicate the description of Processes Control. It is recommended that Windows users first read through this document, then examine the Windows specific details found in the appendix, as well as the on-line manual and the support portals FAQ database.

1.2 Overview

Netcool/OMNIbus Process Control system allows the configuration of local and remote processes. It is designed to simplify the configuration and management of Netcool/OMNIbus components such as object servers, probes and gateways. The term Process Control and Process Automation are inter-changeable, which explains why the default Process Control server is called ‘NCO_PA’ and why all the default names refer to ‘PA’ rather than ‘PC’.

The process control system contains the following elements:

Process Control Agents:

These are programs installed on each host with the responsibility of managing processes.

Command line utilities:

These provide an interface to process management.

The process control agents co-operate automatically. They can start remote processes and are capable of keeping processes running. Processes can be defined to be dependent on the starting of a previous process or have timed threshold dependencies.

```
e.g.
nco_service 'Core'
{
        ServiceType      =       Master
        ServiceStart     =       Auto
        process 'MasterObjectServer' 20
        process 'BackupObjectServer' 'MasterObjectServer'
}
```

The length of the strings used by Process Control need to be limited to 28 characters.

Command line utilities are provided:

```
nco_pa_addentry
nco_pa_convert
nco_pa_crypt
nco_pa_shutdown
nco_pa_start
nco_pa_status
nco_pa_stop
```

2 Process Automation

The process that handles Process Control is called the Process Automation Daemon [nco_pad | nco_pad.exe].

2.1 nco_pad options [UNIX]

```
nco_pad [ options ]

-name         <name>     : Name that server is to use.
-logfile      <filename> : Filename of Log file.
-logsize      <value>    : Maximum log size in Kb (Min:16K, Default:1024Kb).
-debug        <value>    : Server message level ( 1-5 )
-newlog                  : Start a new log file.
-nodaemon                : Do not fork to daemon process.
-apicheck                : Enable Sybase API checking.
-noconfig                : Server does not read any config file.
-configfile   <name>     : The name of the configuration file.
-noautostart             : Do not start auto-start services on start.
-retrytime    <value>    : Failed process start retry time-out.
-stacksize    <value>    : Size of the thread stack.
-redirectfile <filename> : File to redirect stderr/stdout to.
-authenticate <value>    : Authentication system.
                              (UNIX|KERBEROS|PAM|HPTCB|none)
-ticketdir    <dir>      : Directory for kerberos tickets.
-pidmsgpool   <value>    : Size of signal handling message pool.
-msgpoolsize  <value>    : Number of messages available to server.
-roguetimeout <value>    : Allowed time in seconds for process shutdown.
-walkhosttab             : Walk the hosts table to verify all aliases.
-tracenet                : Enable the net library tracing.
-tracemsgq               : Enable the tracing of message queue.
-traceevtq               : Enable the tracing of event queue activity.
-tracemtx                : Enable the tracing of mutex locks.
-help                    : Display this help message text.
-version                 : Display servers version details.
-DNS          <hostname> : Overide hostname for DNS environments
-pidfile      <filename> : Filename for pid storage relative to $OMNIHOME
-killprocessgroup        : Kill processes belonging to the process group
-secure                  : Secure mode.
-user         <name>     : User for logging in to another PAD.
-password     <value>    : Password for logging in to another PAD.
-admingroup   <name>     : Administration Group Name. (Default:ncoadmin)
-keyfile      <filename> : Location of the file containing the key for
                           encrypted passwords.
-cryptalgorithm  <value> : The cryptographic algorithm to use when
                           decrypting passwords.
-connections  <value>     : Maximum number of connections permitted.
                           (Default:30)
```

2.2 nco_pad Process Naming

The process automation daemon's name is set using the '-name' option from the command line. This name is the name defined in the omni.dat file which allows the Sybase client application layer to communicate between process automation daemons and object servers.

omni.dat;

```
[MY_PA]
{
        Primary: <FQDN>  <port>
}
```

Where <FQDN> is the Fully Qualified DomianName or IP Address, and port is a valid available port.

The process is started with the '-name' option to define the nco_pad server name;

```
nco_pad –name MY_PA
```

The process automation daemon can be started without any configuration file, when external actions are being executed from another configured process automation daemon using the '-noconfig'.

Normally a process automation daemon configuration file is used, and is set as follows;

```
nco_pad –name MY_PA –configfile /opt/IBM/netcool/omnibus/etc/MY_PA.conf
```

You can also use the property file method, NCO_PA.props, for example:

```
 nco_pad -propsfile /opt/IBM/netcool/omnibus/etc/MY_PA.props
```

Additionally on a production system nco_pad is usually started at boot-time, using the nco script;

e.g.

```
vi /etc/init.d/nco
```

```
NCO_PA="MY_PA"
```

With additional setting being entered manually further down in the file;

```
${OMNIHOME}/bin/ nco_pad –name ${NCO_PA} -configfile /opt/IBM/netcool/omnibus/etc/$
{NCO_PA}.conf &
```

2.3 Administration Group

The process automation administration group is used to control which users are allowed to access the processes, that are run by the process automation daemon. In UNIX, access is controlled using the default naming service; files. For the process automation daemon to access these files, it needs to be run as the administration user ‘root’.

An example configuration for Solaris is;

```
/etc/passwd:
ncoadmin:x:333:333::/opt/IBM/netcool:/usr/bin/bash
/etc/group:
ncoadmin::333:root,netcool,ncoadmin
```

If another group is required then the process automation daemon must be started with the ‘-admingroup’ option and the group added to the files;

e.g.

```
/etc/group:
myadmin::333:root,netcool,ncoadmin

nco_pad –admingroup myadmin
```

The users added to the administration group should then be able to use the nco_pa_* commands against the process automation daemon.

2.4 Process Management

Some processes misbehave when run using the process automation daemon. In these cases it is necessary to set the ‘-retrytime’ and ‘-roguetimeout’ options. These options allow the managed processes to exit before process automation attempts to restart them. An example of this are the probes that use the non-native probe, which means that the probe is run as two processes, one child process and one parent process. In this case, the child process may not exit immediately, and starting the parent process before it has exited successfully would cause problems with the EMS being connected to.

The alternative to setting these timeouts is to add a ‘sleep’ time to the probes script.

This can be done using the probes env file;

```
cd $NCHOME/omnibus/probes/java
vi  nco_p_alcatel_5620_sam_v8.env
sleep 20
#EOF
```

or a copy of the probes script;

```
cd $NCHOME/omnibus/probes
mv nco_p_alcatel_5620_sam_v8 nco_p_alcatel_5620_sam_v8-
cp nco_p_alcatel_5620_sam_v8- nco_p_alcatel_5620_sam_v8
vi nco_p_alcatel_5620_sam_v8
…
exec …
sleep 20
:wq
chmod 755 nco_p_alcatel_5620_sam_v8
```

This method allows probe customisation without affecting other processes.

2.5 External action considerations

When external actions are being used, it may become necessary to allow more connections to the process control daemon. This is done through increasing the number of allowed connections using the ‘-connections’ option.

With any connection increase, the shell environment variables may need to be increased. The main setting is performed using the ‘ulimit’ command.

e.g.

```
ulimit –a
…
nofiles(descriptors) 256
…
```

The number of open files can be set within the parent script (/etc/init.d/nco) or the users profile or Netcool/OMNIbus env scripts;

```
#! /bin/sh
ulimit –n 2048
```

2.6 Debugging Process Control

Start the process control process in debug mode and examine its log file:

```
$OMNIHOME/bin/nco_pad -config $OMNIHOME/etc/nco_pa.conf -debug 1 &
```

- Check that all processes specified in the configuration file have started and are 'Running' using the nco_pa_status utility
- Those that are 'Pending' may be waiting for dependent process to start or for the specified wait period

For processes that do not start:-

- Check that the 'command' string can be run from the command line as the user specified in the configuration file
- Check that the correct hostname is specified, and that it is ping-able from the command line
- Check that the user is specified correctly, either as a number or quoted string
- Check that the routing is specified correctly and that only the required Process Control servers are being router
- Check that the security details are correct using nco_p_status and the encrypted passwords are correct

Some of the other useful additional options are;

-newlog Creates a new log file each time the process automation daemon is started.

-logfile Define a different log file name to ensure older logs are not overwritten.

-logsize Increase the maximum log size from 1024Kb to 10240Kb or more.

3 Configuring Process Control

3.1 The Administration group and user

The process control administrator group ‘ncoadmin’ defines which users can connect to the process control daemon.

e.g. For Solaris

If /etc/nsswitch.conf defines a ‘files’ configuration then the file /etc/group needs to contain:

```
ncoadmin::333:ncoadmin,root
```

To add in a non-root user to access process control you would add the following line to the /etc/passwd file and then set their password;

```
ncoadmin::333:333:NCO_Admin:/opt/IBM/netcool:/bin/sh
:wq
% pwconv
% passwd ncoadmin
Password:
Confirm password:
% nco_pa_status -user ncoadmin
Password:
```

3.2 Service and Process Names

Due to the available character widths for service and process names (21 characters), their names should be concise and human readable;

e.g.
Service names : CoreGateways, TrapdProbes, NetcoolServices etc.
Process names : PrimaryBiGateway, MTrapd_01, WebGUI_1 etc.

Equally using the actual omni.dat given name for the main processes can reduce confusion over which process is which;
e.g.
Process names : NCOMS_P, NCOMS_B, BI_GATE, G_ORACLE, etc.

3.3 Process Control Status

The nco_pa_status command allows an administrator user (a user in the Adminstration group) to check the status of processes under the control of the process automation daemon. In order to hide the administrator's password from normal users the nco_pa_crypt command can be used to encrypt the password before using it with the nco_pa_status command.

e.g.

```
nco_pa_crypt password
ECEDBJAGBJFHGD
nco_pa_status -user ncoadmin -password ECEDBJAGBJFHGD
```

3.4 Configuration file commands

There are four main types of commands;

- **nco_process** *'MasterObjectServer'*
- **nco_service** *'Core'*
- **nco_security**
- **nco_routing**

Example of a process definition;

```
nco_process 'MyProbe'
{
        Command '$OMNIHOME/probes/nco_p_mttrapd -propsfile $OMNIHOME
/probes/solaris2/my_mttrapd.props' run as 'root'
        Host            =       'myhost'
        Managed         =       True
        RestartMsg      =       'Restored: ${NAME} on ${HOST}.'
        AlertMsg        =       'Died    : ${NAME} on ${HOST}.'
        RetryCount      =       5
        ProcessType     =       PaNOT_PA_AWARE
}
```

In this example the probe simnet has been defined to run a specific properties file, and is to be run as username 'root'. The process is to have five attempts at starting before giving up. The process is registered as unaware of process control.

```
nco_service 'MyExample'
{
        ServiceType     =       Non-Master
        ServiceStart    =       Non-Auto
        process 'MyProbe' 20
        process 'MyOtherProbe' 'MyProbe'
}
```

The service 'MyExample' would not start the two named processes automatically (**Non-Auto**). When the service 'MyExample' is started, 'MyProbe' would be started after 20 seconds, and 'MyOtherProbe' would only start if 'MyProbe' was successfully started.

There should be only one Master service in the process control configuration file.

'nco_security' is used to define which hosts are allowed to issue commands to the process control server.

'nco_routing' is used to define which process control servers can communicate with each other and the secure users and their 'nco_pa_crypt' encrypted passwords .

3.5 Hostname

Process automation uses the local systems interface, and the IP addresses associated with it. It is important to use the Fully Qualified Domain Name or unique IP Address for hostnames used within the process control configuration files.

i.e.

If the identity of a system is ‘myhost.mydomain.com’, then ‘myhost’ would not work for process control, other hosts called ‘myhost’ could exist within other domains (e.g. ‘myhost.yourdomain.com’). The precise specification of hostnames allows the process control daemon to handle machines of the same name in different domains.

If process control daemon is unable to find the local machines primary hostname from the naming service, it automatically adds an entry to indicate itself. All other hostnames are then considered to be remote.

3.6 Setting umask’s for non-root users

```
Command '[UMASK=23]$OMNIHOME/bin/nco_p_heartbeat' run as 'ncoadmin'
```

3.7 PA Aware or NOT PA Aware

Within the definition of the nco_process you can configure the process to be PA aware or not PA aware. All bespoke scripts should be entered as ‘PaNOT_PA_AWARE’. To determine if the specific Netcool/OMNIbus process has the PA functionality in unix you can check using the ‘strings’ function;

e.g.

```
strings nco_p_nonnative  | grep PA
*NOT* running PA Aware
PaInformPA() failed: retcode = %u
Running PA Aware
```

If the process executable has the PA feature, then the process can be entered as ‘PaPA_AWARE‘, although this may not necessarily be the best option.

3.8 Properties

The nco_pa.props file has a number of properties that are useful to improve process control behaviour.

```
RetryTime: 5                   Process retry time-out
RogueTimeout: 30               Time in seconds allowed for a process shutdown
KillProcessGroup: FALSE        Kill processes belonging to the processes process group (UNIX only)
```

Setting higher RetryTime and RogueTimeout settings improves behaviour where probes can be busy and/or require additional time before an attempt to restart the probe should occur.

Enabling KillProcessGroup is useful where non-native probes and gateways are being used.

```
Connections: 30           Maximum number of connections permitted
```

Increasing the Connections setting is useful when external actions are being used.

4 Troubleshooting

Why does process control not work as expected?

- You need to use ‘nco_pad -authenticate PAM’ if you are using Linux and want to use authentication
- The user is not a member of ncoadmin on the process control server machine
- There is a mismatch in the passwords being used

4.1 Authenticate NONE

In order to use the nco_pa_<command> commands, authentication needs to be configured. This is sometimes undesirable due to potential security risks, and when only local access to the nco_pad process is required. In these circumstances the '-authenticate NONE' option or Authenticate: 'NONE' property can be used, with the NCO_PA process interface being configured with 'localhost' to prevent unauthorized remote access.

This configuration is useful where PAM and UNIX authentication is difficult to configure.

File : $NCHOME/etc/omni.dat

```
[NCO_PA]
{
        Primary: localhost 4200
}
```

4.2 Problems restarting processes

In some circumstances processes require additional handling in order to recover from an unexpected exit.

By default the backoff strategy is to restart the process every 2^n seconds to a maximum of 256 seconds:

```
2,4,8,16,32,64,128,256 -> 2,4,8,16,32,64,128,256 ...
```

Although this retry sequence suits most processes, it may be too aggressive for processes that require time to logoff a system or clean up large volumes of files. Problems can also occur when the process becomes busy and the process manager marks the process as dead rather than waiting for a response.

Note that if the process takes longer than the RetryTime, the process will be restarted immediately as it will have been deemed as run successfully, so in general, it is best to set a RetryTime to 30 or 60 seconds. This will allow the back-off strategy to be activated.

By default the nco_pa.props set:

```
RetryTime: 5
RogueTimeout: 30
```

Increasing these timings will provide better handling of more heavily loaded processes.

Additionally you can add 'sleep n' to the processes env file, to affect an individual processes start-up timing.
e.g.

```
vi $NCHOME/omnibus/probes/java/nco_p_generic_3gpp.env

# Added a sleep time to allow probe to exit from system
sleep 10
#EOF
```

4.3 Common log messages

Message : No free messages

Solution : nco_pad -msgpoolsize <VALUE> where value is 10x greater than before.

5 APPENDIX

5.1 Example NCO_PA.props property file

```
Name: 'NCO_PA'
ConfigFile: '$OMNIHOME/etc/NCO_PA.conf'
Authenticate: 'NONE'
RetryTime: 60
RogueTimeout: 120
KillProcessGroup: TRUE
Connections: 100
#EOF
```

5.2 Example NCO_PA.conf configuration file

The following is an example NCO_PA.conf file for the backup object server and bi-directional gateway.
In this example the localhost is used, but typically the IP Address or FQDN of the server is used.
Also NCO_PA is the generic name and would normally need to unique name within a typical production system.
In this example processes are run as nrv81.

```
nco_process 'AGG_B'
{
        Command '$OMNIHOME/bin/nco_objserv -name AGG_B -pa NCO_PA' run as 'nrv81'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${NAME} running as ${EUID} has been restored on ${HOST}.'
        AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
        RetryCount      =       0
        ProcessType     =       PaPA_AWARE
}

nco_process 'AGG_GATE'
{
        Command '$OMNIHOME/bin/nco_g_objserv_bi -propsfile $NCHOME/omnibus/gates/AGG_GATE/AGG_GATE.props' run as
'nrv81'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${NAME} running as ${EUID} has been restored on ${HOST}.'
        AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
        RetryCount      =       0
        ProcessType     =       PaPA_AWARE
}

nco_service 'Core'
{
        ServiceType     =       Master
        ServiceStart    =       Auto
        process 'AGG_B' NONE
        process 'AGG_GATE' 'AGG_B'
}

nco_service 'InactiveProcesses'
{
        ServiceType     =       Non-Master
        ServiceStart    =       Non-Auto
}

nco_routing
{
        host 'localhost' 'NCO_PA'
}
#EOF
```

5.3 Running Daemons under Process Control

Process Control cannot run daemons directly (syslogd, etc.), since they are required to run in the foreground of the parent shell. As a workaround a wrapper script can be used to start and monitor the daemons running process;

```
#! /bin/sh
# Script to start a ${DAEMON} and then check that it is running
# Set full path to DAEMON here:-
DAEMON=/path/to/binary/daemon
export DAEMON
PATH=/usr/bin
export PATH
PID=`ps -ef |grep ${DAEMON} |grep -v grep | awk '{print $2}'`
PID=${PID:=not}
I=0
while [ $I -eq 0 ];
do
if [ $PID = "not" ]; then
echo "restarting"
daemon &
exit
else
sleep 5
fi
PID=`ps -ef |grep ${DAEMON} |grep -v grep | awk '{print $2}'`
PID=${PID:="not"}
done
```

5.4 Running external actions under Windows

The Object Server [nco_objserv.exe] requires a valid Windows user to be configured for use with external actions;

Edit the object servers property file NCOMS.props to include a valid administration user [default is ncoadmin];

```
PA.Name    : 'NCO_PA'
PA.Username: 'ncoadmin'
PA.Password: 'DJELBIAEAOFFGAFPDLFOFNGD'
```

Where NCO_PA is the name given for the process automation process in the servers editor, ncoadmin is a Windows user whose account is unlockable [expires/password attempts/etc.], and 'DJELBIAEAOFFGAFPDLFOFNGD' is the nco_g_crypt'ed password for the Windows user ncoadmin.

Update the process automation configuration file nco_pa.conf with the process automation details;

```
Command '$OMNIHOME/bin/nco_objserv -name NCOMS -pa NCO_PA' run as 'ncoadmin'
```

Run the nco_pad process with '-authenticate none'.

Add a user to the object server to mirror the process automation user [ncoadmin].

e.g.

```
Username  : ncoadmin
Full Name : NCO PAD User
Userid    : 333
Password  : <same as Windows user account>
```

Update the send_mail procedure to use the process automation user [ncoadmin], on the local server, and run the custom BAT script;

```
Executable : %OMNIHOME\utils\nco_mail.bat
Host       : localhost
User ID    : 333
Group ID   : 0
```

Create a test BAT script [%OMNIHOME\utils\nco_mail.bat];

```
REM ARG#1: node string
REM ARG#2: severity integer
REM ARG#3: subject string
REM ARG#4: email string
REM ARG#5: summary string
REM ECHO ARGUMENTS to log
echo ARG1=%1, ARG2=%2, ARG3=%3, ARG4=%4, ARG5=%5 >> c:\temp\nc_mail.log
```

In the example the default trigger mail_on_critical and procedure are used, to confirm the required behaviour is configured. Create copies of the trigger and procedure to meet the custom solutions requirements.

Create two filters in the event list to monitor the test events progress.

Pre-Trigger SQL filter:-

```
(( Grade < 2 ) AND ( Identifier = 'mail-test-alert' ))
```

Post-Trigger SQL filter:-

```
(( Grade >= 2 ) AND ( Identifier = 'mail-test-alert' ))
```

Start nco_pad.exe from the command line, using the required options, with debugging enabled.

```
nco_pad.exe -authenticate none –debug 1
```

Use the SQL workbench in nco_config [Administrator] to insert the test event with the mail_on_critical trigger disabled;

```
insert into alerts.status (Identifier, Severity, Summary, AlertGroup, AlertKey, Node,
Manager, Type, Grade, LastOccurrence)
values  ('mail-test-alert',5,'mail-test-alert down', 'testing','0','Node0','TEST',1,1,
(getdate -(60*30)));
```

Check the event is visible in the Pre-trigger event list view.

Enable the mail_on_critical trigger and check the test event moves to the
post-trigger event list view.

Check the nc_mail.log file is written to, and later that the mail is sent once mailing is added to the custom BAT script.

To re-test use the following SQL in the SQL Work bench, to reset the test event;

```
update alerts.status set Severity = 5 , Grade = 1, Acknowledged = 0, LastOccurrence =
(getdate -(60*30))
where Identifier = 'mail-test-alert';
```

After testing is completed, remember to install any required options in Windows Services;

e.g.

```
nco_pad.exe /REMOVE

nco_pad.exe /INSTALL /CMDLINE "-authenticate none"
```

5.5 Running external actions in Linux

Create the process control user and administration group;

```
/etc/passwd:
nrv81:x:123:123::/opt/nrv81:/bin/sh
/etc/group:
nrv81admin:x:123:root,nrv81
```

Determine the process control user’s encrypted password string;

```
nco_pa_crypt password
EEDPBIBGCBFHGJFF
```

Edit the object server property file and update the object servers process automation’s settings;

```
vi $NCHOME/omnibus/etc/NCOMS_P.props
PA.Name: 'MY_PA’
PA.Username: 'nrv81’
PA.Password: 'EEDPBIBGCBFHGJFF'
```

Configure PAM using the default PAM files as defined in the Netcool/OMNIbus manual;

```
cd /etc/pam.d
cp passwd netcool
cp system-auth nco_objserv
```

Example MY_PA.conf:-

```
nco_process 'NCOMS_P'
{
        Command '$OMNIHOME/bin/nco_objserv -name NCOMS_P -pa MY_PA' run as 'nrv81'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${NAME} running as ${EUID} has been restored on ${HOST}.'
        AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
        RetryCount      =       0
        ProcessType     =       PaPA_AWARE
}
nco_service 'Core'
{
        ServiceType     =       Master
        ServiceStart    =       Auto
        process 'NCOMS_P' NONE
}
nco_service 'InactiveProcesses'
{
        ServiceType     =       Non-Master
        ServiceStart    =       Non-Auto
}
nco_routing
{
        host 'localhost' 'MY_PA'
}
```

Start the process automation daemon as root;

```
nco_pad -configfile $NCHOME/omnibus/etc/MY_PA.conf -name MY_PA -admingroup nrv81admin

nco_pa_status -server MY_PA -user nrv81 -password netcool
------------------------------------------------------------------------------
Service Name         Process Name    Hostname   User          Status      PID
------------------------------------------------------------------------------
Core                 NCOMS_P         localhost  nrv81         RUNNING     867
------------------------------------------------------------------------------
```

Create a script to test external actions;

```
e.g.
vi $OMNIHOME/utils/myAction.sh
#! /bin/sh
echo $* >> $OMNIHOME/log/myAction.log
:wq
```

Login to the object server as root and create a test trigger group, an external action and procedure;

```
Trigger group : Test
```

Copy send_mail procedure to test_external_action;

```
External action : $OMNIHOME/utils/myAction.sh
Hostname : localhost
User ID : 123
Group ID : 123
```

Copy mail_on_critical to test_external_action;

```
Trigger group : Test
Enabled : Yes
Action :
execute test_external_action (critical.Node, critical.Severity, 'Netcool Email',
'root@localhost', critical.Summary, 'localhost');
```

Use the SQL workbench in nco_config [Administrator] to insert the test event with the mail_on_critical trigger disabled;

```
insert into alerts.status (Identifier, Severity, Summary, AlertGroup, AlertKey, Node,
Manager, Type, Grade, LastOccurrence)
values  ('mail-test-alert',5,'mail-test-alert down', 'testing','0','Node0','TEST',1,1,
(getdate -(60*30)));
```

Check the event is visible in the event list.
Check the nc_mail.log file is written to.

To re-test use the following SQL in the SQL Work bench, to reset the test event;

```
update alerts.status set Severity = 5 , Grade = 1, Acknowledged = 0, LastOccurrence =
(getdate -(60*30))
where Identifier = 'mail-test-alert';
```

5.6 Running nco_pad as a non-root user

It is possible to allow a non-root user to access the required system files [/etc/shadow] using the operating systems file access control commands.

For Solaris the command is setfacl.

e.g. as the root user

```
setfacl -s user:nrv81:r--,user::r--,group::---,mask:r--,other:--- /etc/shadow
```

If authentication is not required then the -authenticate NONE switch can be used to disable the need to provide a password when using the nco_p_* commands.

Root access is required if processes need to be run as users other than the nco_pad process user.

5.7 Toggle probe processes

You can use process control to toggle between probe or gateway processes.

Toggling works through checking the status of the processes and starting the primary process if it is not running, and stopping the secondary process. When the primary process is running the primary process is stopped and the secondary process is started.

This script can be run form the command line to toggle the processes, or else added to a trigger and external action call, to allow the object server to manage the processes.

The script could also be called from a Web GUI tool, under the correct configuration at the Web GUI server.

5.7.1 toggle_between_probes.sh

```
#! /bin/ksh
#
# User requires NCHOME and OMNIHOME set
# and bin paths set
#
# Configure settings
# Password crypted using nco_g_crypt netcool
#
PRIMARY_PA=NCO_PA_1
BACKUP_PA=NCO_PA_2
PRIMARY_PROBE=Probe01
BACKUP_PROBE=Probe02
PA_USERNAME=root
PA_PASSWORD=ECEDBJAGBJFHGD
# EXPORTS
export PRIMARY_PA BACKUP_PA PRIMARY_PROBE BACKUP_PROBE PA_USERNAME PA_PASSWORD
export PRIMARY_STATUS BACKUP_STATUS STATUS
# Get the probes current status
PRIMARY_STATUS=`nco_pa_status -server ${PRIMARY_PA} -user ${PA_USERNAME} -password $
{PA_PASSWORD} | grep ${PRIMARY_PROBE} | cut -c64- | awk '{print $1 }'`
BACKUP_STATUS=`nco_pa_status -server ${BACKUP_PA} -user ${PA_USERNAME} -password $
{PA_PASSWORD} | grep ${BACKUP_PROBE}  | cut -c64- | awk '{print $1 }'`
# Debugging messages
echo PRIMARY_STATUS=$PRIMARY_STATUS
echo BACKUP_STATUS=$BACKUP_STATUS
# If not running
if [ ! -z "$PRIMARY_STATUS" ]
then
  if [ ! -z "$BACKUP_PROBE" ]
  then

  STATUS=$PRIMARY_STATUS:$BACKUP_STATUS
# Debugging messages
  echo "STATUS=$STATUS"
        case $STATUS in
# Restart primary porbe
        DEAD:RUNNING|DEAD:DEAD|DEAD:PENDING|PENDING:PENDING)
# Stop backup
nco_pa_stop -server ${BACKUP_PA} -user ${PA_USERNAME} -password ${PA_PASSWORD} -process $
{BACKUP_PROBE} > /dev/null 2>&1
# Restart primary
nco_pa_stop -server ${PRIMARY_PA} -user ${PA_USERNAME} -password ${PA_PASSWORD} -process
${PRIMARY_PROBE} > /dev/null 2>&1
sleep 2
nco_pa_start -server ${PRIMARY_PA} -user ${PA_USERNAME} -password ${PA_PASSWORD} -process
${PRIMARY_PROBE}
        ;;
        RUNNING:DEAD|RUNNING:PENDING|PENDING|DEAD)
# STOP Primary probe
nco_pa_stop -server ${PRIMARY_PA} -user ${PA_USERNAME} -password ${PA_PASSWORD} -process
${PRIMARY_PROBE} > /dev/null 2>&1
# Restart backup
nco_pa_stop -server ${BACKUP_PA} -user ${PA_USERNAME} -password ${PA_PASSWORD} -process $
{BACKUP_PROBE} > /dev/null 2>&1
sleep 2
nco_pa_start -server ${BACKUP_PA} -user ${PA_USERNAME} -password ${PA_PASSWORD} -process
${BACKUP_PROBE}
        ;;
        *)
        echo "*** Current Status is unknown"
        echo "Checking ... "
        esac
fi
fi
sleep 4
# Current status
DATE=`date`
echo " Current status : " $DATE
nco_pa_status -server ${PRIMARY_PA} -user ${PA_USERNAME} -password ${PA_PASSWORD} | grep
${PRIMARY_PROBE}
nco_pa_status -server ${BACKUP_PA} -user ${PA_USERNAME} -password ${PA_PASSWORD} | grep $
{BACKUP_PROBE}
exit 0

#EOF
```

5.7.2 Process control configuration

```
File : NCO_PA_1.props

RetryTime    : 30
RogueTimeout : 60
# End of File

File : NCO_PA_1.conf

nco_process 'Probe01'
{
        Command '$OMNIHOME/probes/nco_p_simnet -propsfile
$OMNIHOME/probes/linux2x86/simnet1.props' run as 'nrv81'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${NAME} running as ${EUID} has been restored on $
{HOST}.'
        AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
        RetryCount      =       0
        ProcessType     =       PaPA_AWARE
}

nco_service 'Core'
{
        ServiceType     =       Master
        ServiceStart    =       Auto
        process 'Probe01' NONE
}

nco_service 'InactiveProcesses'
{
        ServiceType     =       Non-Master
        ServiceStart    =       Non-Auto
}


nco_routing
{
        host 'localhost' 'NCO_PA_1'
}
#EOF
```

```
File : NCO_PA_2.props

RetryTime    : 30
RogueTimeout : 60
# End of File

File : NCO_PA_2.conf

nco_process 'Probe02'
{
        Command '$OMNIHOME/probes/nco_p_simnet -propsfile
$OMNIHOME/probes/linux2x86/simnet2.props' run as 'nrv81'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${NAME} running as ${EUID} has been restored on $
{HOST}.'
        AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
        RetryCount      =       0
        ProcessType     =       PaPA_AWARE
}

nco_service 'Core'
{
        ServiceType     =       Master
        ServiceStart    =       Non-Auto
        process 'Probe02' NONE
}

nco_service 'InactiveProcesses'
{
        ServiceType     =       Non-Master
        ServiceStart    =       Non-Auto
}


nco_routing
{
        host 'localhost' 'NCO_PA_2'
}
#EOF
```