Tivoli Netcool Support's Guide to Netcool/OMNIbus Suppression and Escalation

by

Jim Hutchinson

Document release: 2.0

Table of Contents

1Introduction......2

1.1Overview......2
1.2The SuppressEscl flag......2

2Event Management......3

2.1Restricting Events......3
2.2Creating Restriction Filters......3

3Maintenance Event Handling......4

3.1Suppressing events at the probe......4
3.2Manually suppressing events......4

4The SuppressEsclFilter Solution......5

4.1Aggregation Layer......5
4.1.1AGG_GATE configuration......5
4.1.2Failover/Failback procedure updates......6
4.2Testing the solutions behaviour......7
4.2.1Whenis command......8
4.2.2insert_event_maintenance.sh script......8
4.2.3insert_event_suppressed.sh script......8
4.3Rules file implementation......9
4.4Automatic Escalation......10

1 Introduction

1.1 Overview

Netcool/OMNIbus provides a suppression and escalation flag called SuppressEscl, which can be used to manage events suppression and escalation within the network management system. There is also a tool called Suppress/Escalate which allows users to set the SuppressEscl using the Netcool/OMNIbus event list.

Although the feature is available as a standard, no guidance is provided in how to implement solutions within a Network Management environment. This Support's guide was written to provide further guidance and example configurations to extend its usage.

1.2 The SuppressEscl flag

Value	Description
0	Normal
1	Escalated
2	Escalated to Level 2
3	Escalated to Level 3
4	Suppressed
5	Hidden
6	Maintenance

2 Event Management

2.1 Restricting Events

Object Server users can have events restricted using the Restriction Filters, as found in the Administration Tool's Users tab.
To restrict all of the unwanted events; Suppressed, Hidden and Maintenance, create a restriction filer like:

Name	filtered
Condition	SuppressEscl < 4

Assign the Restriction Filter to all users who do not need to see the unwanted events, such as users who are not system administrators.

2.2 Creating Restriction Filters

You can create a full set of restriction filters for use with different user roles.
e.g.

For specific access:

Purpose	Name	Condition
Normal events	normal_events	SuppressEscl = 0
L1 events	l1_events	SuppressEscl = 1
L2 events	l2_events	SuppressEscl = 2
L3 events	l3_events	SuppressEscl = 3
Suppressed events	suppressed_events	SuppressEscl = 4
Hidden events	hidden_events	SuppressEscl = 5
Maintenance events	maintenance_events	SuppressEscl = 6

For user roles:

Purpose	Name	Condition
Normal user	normal_user	SuppressEscl = 0
L1 user	l1_user	SuppressEscl <= 1
L2 user	l2_user	SuppressEscl <= 2
L3 user	l3_user	SuppressEscl <= 3
Administration user	admin_user	SuppressEscl < 6

3 Maintenance Event Handling

3.1 Suppressing events at the probe

Events can be suppressed in the rules file using a number of methods. One method is to add a rules file code block to allow events to be discarded or suppressed based upon some time period from the current time. The probes rules file is updated with the new rules file logic at the end. When the outage is required, the probe can be REHUPPED to enable the maintenance mode event handling. Once the maintenance is complete the maintenance rules file code is removed and the probe REHUPPED.

```
# ************** MAINTENANCE MODE LOGIC
if ( match(%start_now,"") )
{
  %STARTTIME=getdate
  %ENDTIME= int(%STARTTIME) + 3600
  %start_now = 1
  log(WARN,"Setting start_now : " + %STARTTIME )
}

$now = getdate
log(DEBUG,"Now : " + $now + " : " + %STARTTIME  + " : " + %ENDTIME)
if ( int($now) > int(%STARTTIME) and int($now) < int(%ENDTIME) )
{
# Suppress ALL events as in MAINTENANCE
  log(WARN,"Suppressing event : " )
  @SuppressEscl = 6
}
#EOF
```

3.2 Manually suppressing events

Administrators can manually suppress events by creating a filter that shows all the events and then manually setting the correct SuppressEscl value. They can also create temporal triggers or modify the new_row and deduplication triggers to set evevents SuppressEscl field based on some Object Server field criteria.

Users are also able to set the SuppressEscl field as required, provided the events UIG|GID has not been set.

4 The SuppressEsclFilter Solution

An example solution using the SuppressEsclFilter custom field is provided to illustrate how events can be managed using a temporal trigger. The solution requires the SuppressEsclFilter value to be set as required in the probe rules files, and the Aggregation object server's configured accordingly with the required database table and triggers.

4.1 Aggregation Layer

You can apply the solution to the Aggregation layer in a multiter system.
Apply the solution to both object servers and then configure the Aggregation layer:
e.g.

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

4.1.1 AGG_GATE configuration

Update the Aggregation gateways mapping and table replicate files, and then restart the gateway.

AGG_GATE.map

```
#
# NcSuppressesclTimingsMap
#
CREATE MAPPING NcSuppressesclTimingsMap
(
        'Name'                  = '@Name',
        'Condition'             = '@Condition',
        'start_time'            = '@start_time',
        'end_time'              = '@end_time',
        'SuppressEscl'          = '@SuppressEscl',
        'Active'                = '@Active'
);


#EOF
```

AGG_GATE.tblrep.def

```
REPLICATE ALL FROM TABLE 'nc_suppressescl.timings'
       USING MAP 'NcSuppressesclTimingsMap'
       INTO 'nc_suppressescl.timings';

#EOF
```

4.1.2 Failover/Failback procedure updates

The nc_suppressescl trigger group needs to be enabled in the backup object server on failover, and disabled on failback. Update the procedures and then enable the nc_suppressescl trigger group on the primary object server, and make sure the backup's nc_suppressescl trigger group is disabled.

For example:

```
------------------------------------------------------------------------
-- Signal triggers: gateway_triggers
------------------------------------------------------------------------
create or replace procedure automation_disable()
begin

-- Disable the automations that should not be
-- running when it is a backup ObjectServer
for each row tg in catalog.triggers where
tg.GroupName IN ( 'primary_only' , 'nc_suppressescl' )
begin
alter trigger group tg.GroupName set enabled false;
end;
end;
go

create or replace procedure automation_enable()
begin

-- Enable the automations that should be
-- running when it is a primary ObjectServer
for each row tg in catalog.triggers where
tg.GroupName IN ( 'primary_only' , 'nc_suppressescl' )
begin
alter trigger group tg.GroupName set enabled true;
end;
end;
go
```

4.2 Testing the solutions behaviour

The solution comes with two examples:

Name	Condition	start_time	end_time	SuppressEscl	Active
AlwaysInMaintenance	**Maintenance**	0	1571017439	6	1
AlwaysSuppressed	**Suppressed**	0	1571017439	5	1

Therefore you can insert events with the SuppressEsclFilter set to 'Maintenance' or 'Suppressed' and the events will have their SuppresssEscl flags set.

For example insert a few example 'Maintenance' and 'Suppressed' events using the provided scripts or SQL commands:

```
./insert_event_maintenance.sh
./insert_event_suppressed.sh
```

The observe that the SuppresssEscl flag is set correctly, and the users with the SuppresssEscl restriction filters are unable to view the events, as expected.

You can use the following filter and view to check these events:

Filter : Testing
AlertGroup = 'Testing'

View : SuppressEscl
Node
Summary
Tally
LastOccurrence
SuppressEscl
SuppressEsclFilter

4.2.1 Whenis command

You can use the whenis command to check and convert unix time's.

Script : whenis

```
#!/bin/perl

if($ARGV[0]) {
       $ctime = $ARGV[0];
} else {
       $ctime = time();
}
$hexval = sprintf "%lX", $ctime;
print "$ctime ($hexval) = \n";
print "\t",gmtime($ctime)." GMT\n";
print "\t",localtime($ctime)." localtime\n";
```

4.2.2 insert_event_maintenance.sh script

This script is designed to insert a maintenance event into the AGG_P object server using a specific user:

```
#! /bin/sh
SQLCMD="$OMNIHOME/bin/nco_sql -server AGG_P -user eventuser -password netcool"
export SQLCMD DATE
DATE=`date`
echo $DATE
$SQLCMD <<EOF
insert into alerts.status
(Identifier,Severity,Summary,AlertGroup,AlertKey,OwnerUID,OwnerGID,LastOccurrence,FirstOc
currence,SuppressEsclFilter )
values ('test-alert-${DATE}',4,'Unique Maintenance event @ '
+to_char(getdate),'testing','Maintenance event',65534,0,(getdate),(getdate),'Maintenance'
);
go
quit
EOF
#EOF
```

4.2.3 insert_event_suppressed.sh script

This script is designed to insert a suppressed event into the AGG_P object server using a specific user:

```
#! /bin/sh
SQLCMD="$OMNIHOME/bin/nco_sql -server AGG_P -user eventuser -password netcool"
export SQLCMD DATE
DATE=`date`
echo $DATE
$SQLCMD <<EOF
insert into alerts.status
(Identifier,Severity,Summary,AlertGroup,AlertKey,OwnerUID,OwnerGID,LastOccurrence,FirstOc
currence,SuppressEsclFilter )
values ('test-alert-${DATE}',4,'Unique Suppressed event @ '
+to_char(getdate),'testing','Suppressed event',65534,0,(getdate),
(getdate),'Suppressed' );
go
quit
EOF
#EOF
```

4.3 Rules file implementation

The SuppressEsclFilter is used for time based Suppression/Escalation, whilst the SuppressEscl field is used for forced Suppression/Escalation. Therefore care needs to be taken when setting these fields and by ordering their setting and preventing both fields being set at the same time.

For the setting of the value of SuppressEsclFilter, ensure that any lookup settings are ordered as required by the system, for example, Customer settings are overridden by Node settings.

Given the lookup files exist and are defined at the start of the rules file:

```
# ************** SuppressEsclFilter LOGIC
# Usage:
# SuppressEsclFilter is for time based Suppress/Escalation
# SuppressEscl is for forced Suppress/Escalation
#
# Set SuppressEsclFilter if not set for lookups
# Customers
# Nodes
#
if(match (@SuppressEsclFilter,””) )
{
  @SuppressEsclFilter = lookup(@Customer, SuppressEsclFilter_Customer_t)
}
if(match (@SuppressEsclFilter,””) )
{
  @SuppressEsclFilter = lookup(@Node, SuppressEsclFilter_Node_t)
}
# Only set SuppressEscl when overriding SuppressEsclFilter
# and SuppressEscl is not set.
if(match (@SuppressEsclFilter,””) )
{
 if( int(@SuppressEscl) == 0 )
 {
  @SuppressEscl = lookup(@Customer, SuppressEscl_Customer_t)
 }
 if( int(@SuppressEscl) == 0 )
 {
  @SuppressEscl = lookup(@Node, SuppressEscl_Node_t)
 }
# Override the timed Suppress/Escalation
 if( int(@SuppressEscl) != 0 )
 {
      @SuppressEsclFilter = “”
 }
}
# End SuppressEsclFilter
```

4.4 Automatic Escalation

The solution includes a temporal trigger to automatically escalate unacknowledged Critical (Severity=5) events. The nc_ecalate trigger runs every ten minutes, and escalates events after an hour has passed [FirstOccurrence]. Any event that is not Acknowledged and is not flashing is processed, with the SuppressEscl flag being increased until it reaches the maximum value of 3, at which point the event is set to Flash.

These settings can be adjusted as required, with other fields and actions used to manage the escalation process. Because Acknowledged events prevent further escalation, the Acknowledge and take ownership tool can be combined to store the user who acknowledges the event first.