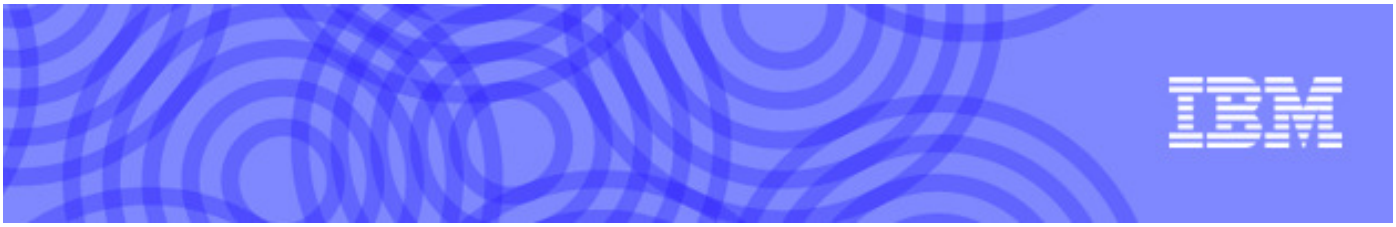


Tivoli Netcool Supports

Guide to

Sub-Second Triggers

by

Jim Hutchinson

Document release: 1.3

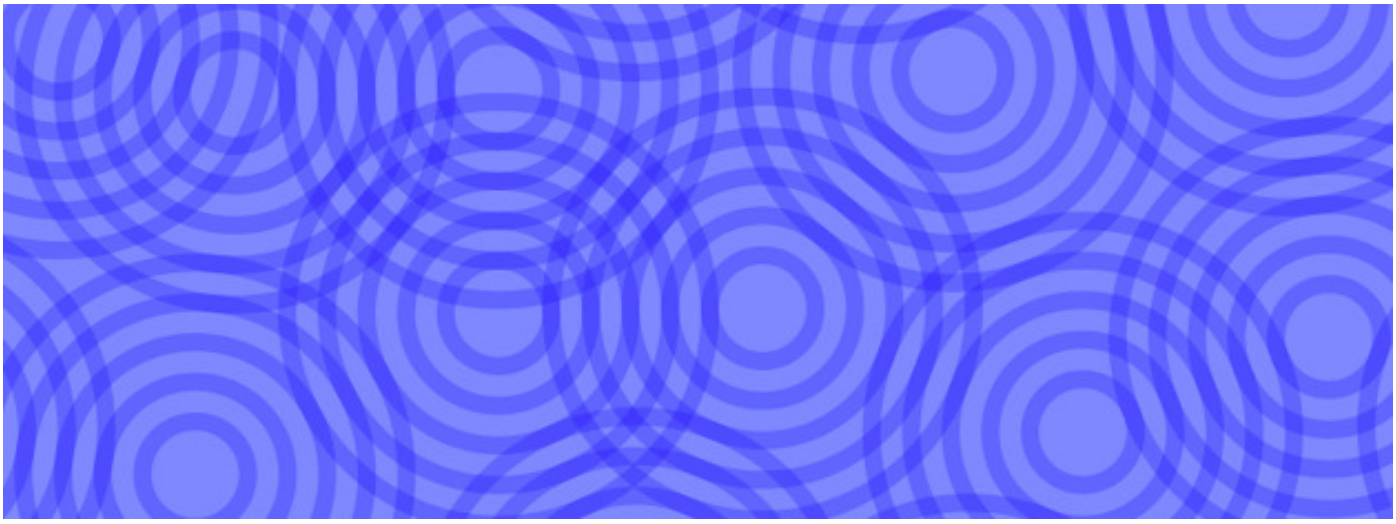


Table of Contents

1 Introduction

1.1 Overview

This document was written to provide a working example of how to configure Netcool/OMNIbus to handle sub-second problem/resolution events.

1.2 Sub-second events

Sub-second problem/resolution events are problem/resolution events that occur in the same second.

Consider the following sequence of events;

Sequence	Timestamp	Type
1	12:01:01	Problem
2	12:01:02	Resolution
3	12:01:05	Problem
4	12:01:05	Resolution
5	12:01:05	Problem

The default object servers triggers would resolve the event as CLEARED (Severity=0). This is incorrect as the last event is a Problem (Type=1) event.

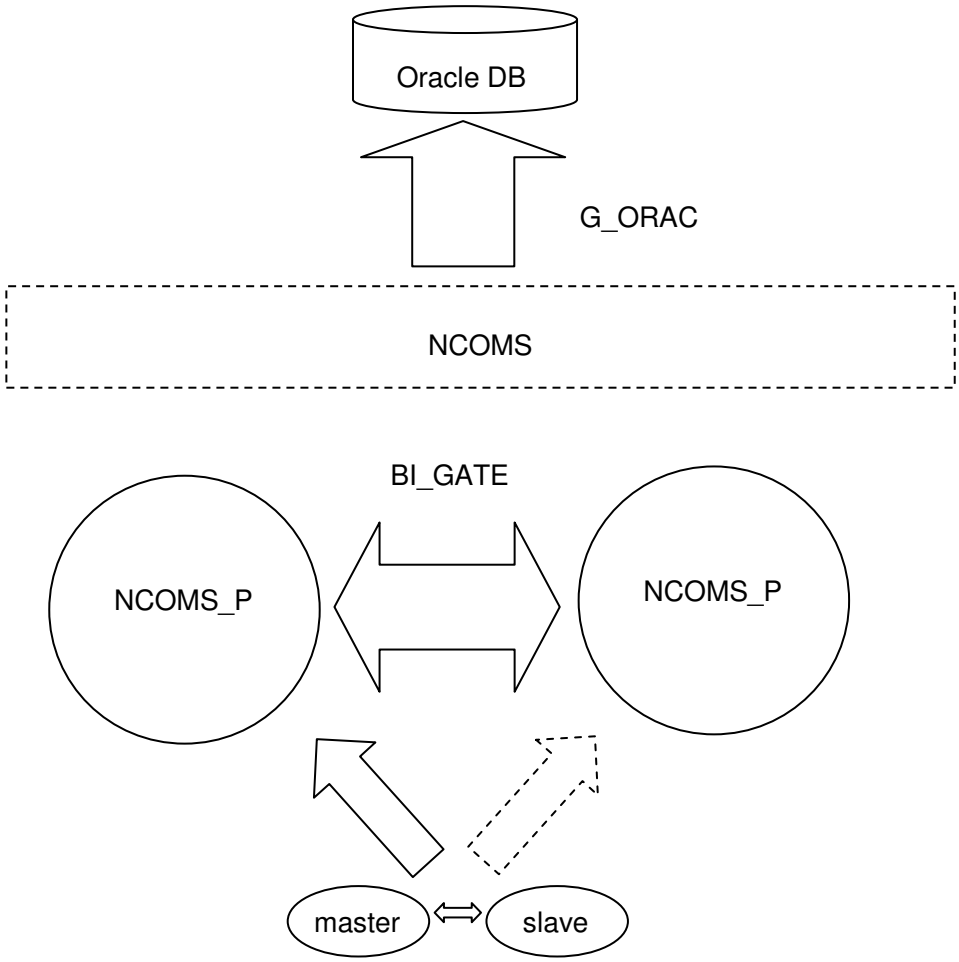
As event rates from devices increases, the probability of same second problem/resolution events being cleared incorrectly has increased resulting in issues with network management.

The solution proposed in this document resolves the issue of same second problem/resolution events using triggers, rather than the solution given using the ProbeSubSecondId field by the main documentation.

1.3 Example Object Server Design

The files provided within the solution are for an example system;

- Dual-resilient object server pair
- MTTrapd probe configured for peer-to-peer (master/slave)
- Oracle gateway reporting to a configured REPORTER Oracle database



2 The Sub-second Solution

The sub-second solution uses a custom table to store the sub-second counter, this counter is updated whenever the alerts.status table is updated. During event resolution the value of the counter is used to decide if the problem event is resolved or not. The solution includes a sub-second journals solution that will allow the nc_jinserts procedure to be used where sub-second journals are most probable.

2.1 Key Objects

New alerts.status Columns:-

NcEventSecondCounter : Integer counter to store current seconds event count
NcReporting : Flag used to control historical event flow

New trigger groups:-

- nc_sub_second
- nc_historical_reporting

New custom table:-

subsecond.counter

TargetTable	varchar(64) primary key,
Counter	int,
Timestamp	time

New triggers:-

- nc_status_reinserts
- nc_status_inserts
- nc_status_updates
- nc_journal_inserts
- nc_new_row
- nc_deduplication
- nc_generic_clear
- nc_delete_clears

New procedure:-

- nc_jinsert

2.2 Installation

The installation of the triggers and procedure into the object server can be performed using nco_sql or manually using the SQL workbench. There is a README.txt file in the sub-second installation directory, and other sub-directories outlining how to install and test the solution, prior to release to a production system.

2.2.1 Object Server Configuration

To install the new triggers and procedure in a new object server;

```
[update omni.dat]
vi $NCHOME/etc/omni.dat
$NCHOME/bin/nco_igen
[create the base object server]
nco_dbinit -server NCOMS_P
[update the object servers property file]
[start the object server]
$NCHOME/omnibus/bin/nco_objserv -name NCOMS_P &
[install the customisation]
cat subsecond.sql | nco_sql -server NCOMS_P -password ''
```

To-turn off trigger groups;

```
cat disable.sql | nco_sql -server NCOMS_P -password ''
```

To turn-on trigger groups;

```
cat enable.sql | nco_sql -server NCOMS_P -password ''
```

Once installed the nc_jinsert procedure must be integrated into all triggers and tools that perform journal entry where journals are at risk of being added in the same second. This is usually only for triggers rather than tools.

Note: Use of $selected_rows within tools is under investigation

2.2.2 Installation Steps Outline

- Update omni.dat
- Copy the object server property files to $OMNIHOME/etc
- Create NCOMS_P and enable the sub-second triggers
- Create NCOMS_B and enable the sub-second triggers
- Disable the nc_generic_clear in NCOMS_B
- Copy BI_GATE to the gates directory
- Start the bi-directional gateway
- Copy the mttrapd probe property files to $OMNIHOME/probes/<platform>

2.3 Testing

The sub-second installation directory includes the configuration and test scripts to check the configuration, and to allow further development, as required. Each sub-directory includes a README.txt describing how to configure and perform the individual tests. Once the dual-resilient object server and Oracle gateway are configured the system may be tested.

Included tests:-

2.3.1 Sub-second problem/resolution

Use the rulesfile and raw capture file in rules sub-directory to test sub-second problem/resolution;

To create a same second problem event [the fix]:-

```
cat problem.cap | $OMNIHOME/probes/nco_p_stdin -rulesfile samesec.rules -server
NCOMS_P
```

Expected : PROBLEM event remains

To create a same second resolution event [to confirm normal operation]:-

```
cat resolution.cap | $OMNIHOME/probes/nco_p_stdin -rulesfile samesec.rules -
server NCOMS_P
```

Expected : Event RESOLVED

2.3.2 Sub-second and normal traps

Use the scripts in dual-resilient/utils sub-directory to send sub-second and normal traps to the object server via the MTTrapd probe configured for peer-to-peer failover/failback. The scripts requires the net-snmp program snmptrap to be installed and in the test users path [see www.net-snmp.org].

sendsubsecondtrap2master : Sends the same ten TCP SNMPv1 traps to the localhost master mttrapd probes port [TCP:localhost:1621]

Expected : One event with tally/count ten

sendtrap2master : Send ten **UNIQUE** TCP SNMPv1 traps to the localhost master mttrapd probes port [TCP:localhost:1621]

Expected : Ten events with tally/count one

sendtrap2slave : Sends ten **UNIQUE** TCP SNMPv1 traps to the localhost slave mttrapd probes port [TCP:localhost:1622]

Expected : NO events seen in object server [events discarded while master probe is running]

2.4 Zip File Contents

README.txt
<dual-resilient>
<oracle_gateway>
<rules>
enable.sql
disable.sql
subsecond.sql
tools.sql

dual-resilient:
<BI_GATE>
<utils>
mttrapd_master.props
mttrapd_slave.props
NCOMS_B.props
NCOMS_P.props
omni.dat
README.txt

dual-resilient/BI_GATE:
objserv_bi.map
objserv_bi.objectservera.tblrep.def
objserv_bi.objectserverb.tblrep.def
objserv_bi.props
objserv_bi.startup.cmd

dual-resilient/utils:
sendsubsecondtrap2master
sendtrap2master
sendtrap2slave

oracle_gateway:
README.txt
CSHRC-10g
nco_g_icmd.props
nco_g_oracle.map
nco_g_oracle.props
nco_g_oracle.startup.cmd
nco_g_oracle.thosts

<rules>:
problem.cap
README.txt
resolution.cap
samesec.rules

3 The ProbeSubSecondId Solution

The ProbeSubSecondId field is the default field used to hold the sub-second counter. By default LastOccurrence can be used, however, it is recommended that a specific internal variable is used to store the events arrival at the probe, e.g. NcEventTimeStamp.

The ProbeSubSecondId solution requires;

- Events to arrive at the probe in sequence
- NcEventTimeStamp to be set to getdate [or else use LastOccurrence]
- ProbeSubSecondId to be set to a unique sequential number in each second

In this way the sequence of problem/resolution events can be managed.

3.1 Considerations

3.1.1 Default Behaviour

In Netcool/OMNIbus v7.3.1, the default behaviour is to have the fields updated at the probe. There are no rulesfile code required as the probe populates the values automatically;

@LastOccurrence is set to an internal timestamp

@ProbeSubSecondId is set to a sequential integer for each second

The deduplication trigger then uses the ProbeSubSecondId number to determine if an event is new or old, and discards it if it is older than the one stored already in the object server in the same second.

It is important to ensure that either the default behaviour is used, or the custom solution presented earlier within this document. Mixing the two methods may result in event loss.

3.1.2 Normal Behaviour

Normally events have a local event timestamp, which is used to determine event sequencing, which is stored in the LastOccurrence field. Check the probes rules file for which token is used to set LastOccurrence. If LastOccurrence is unset within the probes rules file, then it will get set to the ‘getdate’ value automatically in Netcool/OMNIbus v7.3.x or above, and to zero in earlier versions, and so will need to be set manually.

Within the Generic Clear trigger the event source [@Node] is used to differentiate problem/resolutions. Therefore in the traditional case same second events at the event source are less probable.

It is therefore the decision of the network administrator to determine the likelihood of same second events and the impact on the ability of the object server to monitor the system accurately.

3.2 Rulesfile code [pre-Netcool/OMNIbus v7.3.x]

```
# Set the time field NcEventTimeStamp to be used in the Generic
Clear trigger to getdate
@NcEventTimeStamp = getdate


# For pre-Netcool/OMNIbus v7.3.x need to set ProbeSubSecondId
if (match(%EventTimeStamp,@NcEventTimeStamp))
{
    %SubSecondCounter = int(%SubSecondCounter) + 1
}
else
{
    %EventTimeStamp = @NcEventTimeStamp
    %SubSecondCounter = 1
}

@ProbeSubSecondId = %SubSecondCounter
```

3.3 System updates

The object server's need a new field to hold the probe event time and sub-second counter:

For alerts.status [events]

```
alter table alerts.status add NcEventTimeStamp time;
go
```

For alerts.problem_events [generic clearing]

```
alter table alerts.problem_events add NcEventTimeStamp time;
go
alter table alerts.problem_events add ProbeSubSecondId int;
go
```

The object servers deduplication triggers should set the fields explicitly;

```
set old.ProbeSubSecondId = new.ProbeSubSecondId
set old.NcEventTimeStamp = new.NcEventTimeStamp
```

The generic_clear trigger then needs to include logic to take account of the ProbeSubSecondId and the NcEventTimeStamp instead of using the LastOccurrence, which is usually used to hold the element managers timestamp.

Review the subsecond.sql file to see how the generic clear trigger is modified, otherwise refer to example given.

3.3.1 Example generic_clear logic

```
for each row problem in alerts.status where
        problem.Type = 1 and problem.Severity > 0 and
(problem.Node + problem.AlertKey + problem.AlertGroup + problem.Manager) in
        ( select Node + AlertKey + AlertGroup + Manager from alerts.status
where Severity > 0 and Type = 2 )
begin
        insert into alerts.problem_events
(Identifier,NcEventTimeStamp,AlertKey,AlertGroup,Node,Manager,Resolved,
ProbeSubSecondId)
                values ( problem.Identifier, problem.NcEventTimeStamp,
problem.AlertKey, problem.AlertGroup, problem.Node, problem.Manager, false,
problem. ProbeSubSecondId);
end;

-- For each resolution event, mark the corresponding problem_events entry as
resolved
-- and clear the resolution
for each row resolution in alerts.status where resolution.Type = 2 and
resolution.Severity > 0
begin

set resolution.Severity = 0;

update alerts.problem_events set Resolved = true where
(NcEventTimeStamp < resolution.NcEventTimeStamp and
Manager = resolution.Manager and Node = resolution.Node and
AlertKey = resolution.AlertKey and AlertGroup = resolution.AlertGroup) or
(NcEventTimeStamp = resolution.NcEventTimeStamp and
ProbeSubSecondId < resolution. ProbeSubSecondId and
Manager = resolution.Manager and Node = resolution.Node and
AlertKey = resolution.AlertKey and AlertGroup = resolution.AlertGroup);

end;
        -- Clear the Resolved problem events
        for each row problem in alerts.problem_events where problem.Resolved =
true
        begin
                update alerts.status via problem.Identifier set Severity = 0;
        end;
        -- Remove all entries from the problems table
        delete from alerts.problem_events;
end;
go
```

3.3.2 Multitier considerations

If the object servers are part of a multitier configuration, the new fields need to be added to the collection and aggregation layer object servers. Then the object server gateways mapping files updated to include the new fields.

```
'NcEventTimeStamp'   =     '@NcEventTimeStamp',
'ProbeSubSecondId'   =     '@ProbeSubSecondId',
```

Once completed the gateways need to be restarted and the probes rules file updated, as given earlier. Once the data is seen to flow to the aggregation object server the new generic_clear trigger can be enable, and the default trigger disabled.

The new generic_clear trigger will begin to work once the fields are populated correctly; Test behaviour by sending a same second problem and resolution events from the collection layer.

4 Example nco_pa.conf

```
nco_process 'NCOMS_P'
{
        Command '$OMNIHOME/bin/nco_objserv -name NCOMS_P -pa NCO_PA' run as 'nrv731'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${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 'NCOMS_B'
{
        Command '$OMNIHOME/bin/nco_objserv -name NCOMS_B -pa NCO_PA' run as 'nrv731'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${NAME} running as ${EUID} has been restored on ${HOST}.'
        AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
        RetryCount      =       0
        ProcessType     =       PaPA_AWARE
}
nco_process 'BI_GATE'
{
        Command '$OMNIHOME/bin/nco_g_objserv_bi -propsfile
$OMNIHOME/gates/BI_GATE/objserv_bi.props' run as 'nrv731'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${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 'MTTRAPD_MASTER'
{
        Command '$OMNIHOME/probes/nco_p_mttrapd -propsfile
$OMNIHOME/probes/solaris2/mttrapd_master.props' run as 'nrv731'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${NAME} running as ${EUID} has been restored on ${HOST}.'
        AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
        RetryCount      =       0
        ProcessType     =       PaPA_AWARE
}
nco_process 'MTTRAPD_SLAVE'
{
        Command '$OMNIHOME/probes/nco_p_mttrapd -propsfile
$OMNIHOME/probes/solaris2/mttrapd_slave.props' run as 'nrv731'
        Host            =       'localhost'
        Managed         =       True
        RestartMsg      =       '${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
        process 'NCOMS_B' NONE
        process 'BI_GATE' NONE
        process 'MTTRAPD_MASTER' NONE
        process 'MTTRAPD_SLAVE' NONE
}

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

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