Using Netcool To Create Availability Reports

James Brunke
Chief Technologist
Crystal Technology Solutions Group Inc
jbrunke@ctsgi.com
October 13, 2004

What will I get out of this presentation?

- Relatively simple method
- Based on well known Netcool concepts
- Extensible/applicable to any/all environments
- Some code to munch on (Yum!)

The Problem

- IT Director has weekly meeting with all business line stake holders to discuss technology issues
- Prior to meeting, IT Director review business impacting outages with IT managers prior to weekly meeting to prepare for discussion
- Getting constantly surprised at meeting by “unknown” outages

The Solution

- Single automated report of all outage events
- Report will be presented at weekly business manager's meeting
- NO SURPRISES!

The Availability Report

Shouldn't it be called the Unavailability Report?

- Report should contain information on
  - Network device outages
  - Servers outages
  - Websites outages
  - Network performance exceptions
- Each row of report would contain info on a single outage:
  - Start Time
  - End Time
  - Description of Outage
  - Description of Corrective Action

Netcool Deployment

- ObjectServer
- Multithreaded Trap Probe
- NMS Probe
- Internet Service Monitors
- Reporter Gateway
- Trouble Ticket Gateway

Monitoring

- Receive events from existing Network Management System which is polling all critical IP nodes
- Receive outage events via traps from ATM switch network devices
- Receive network performance events via traps from Network Performance Management platform
- ISMs setup to monitor critical IP services (DNS, DHCP, HTTP)

Concept Review - Deduplication

- Repeated alerts are identified and stored as a single alert to reduce the amount of data in the ObjectServer.
- @Identifier field controls which events deduplicate
- Unique @Identifier created in rules file
- On deduplication, fields that are updated include:
  - @LastOccurrence
  - @Tally
  - @StateChange
  - Any fields set to UPDATE ON DEDUPLICATION in sql file
  - Any fields where update() function is used in rules file
- Already done for majority of events in supplied rules files & the NCiL/SiL

Concept Review - Event Correlation

- "Clearing an event" means to set @Severity = 0
- @Type = 1 are Problems, @Type = 2 are Resolutions
- Generic Automation clears pair of events when following fields match:
  - @Manager
  - @Node
  - @AlertGroup
  - @AlertKey
- Alarms are only cleared if Resolution event comes after Problem
- Example: LinkUp events clears LinkDown events
- Already done for majority of events in supplied rules files & the NCiL/SiL

So what?

How can we use these built in features to help us create our report?

New Configuration

- New ObjectServer fields
  - LastClearTime (time) – What time was this Problem last cleared?
  - Duration (int) – What is the total accumulated problem duration?
  - ReportFlag (int) – Do we want this event included on the report?
- We use the Generic Automation to populate new fields for Problem events (@Type = 1)

Generic Automation (pre 3.6) Trigger

- **GenericClear**

  select * from alerts.status where Type = 2 and Severity > 0;

Generic Automation (pre 3.6) Old Action

- GenericClear

```
update alerts.status set Severity = 0 where Severity >
    0 and Type = 1 and LastOccurrence <
    @LastOccurrence and AlertGroup = '@AlertGroup'
    and Manager = '@Manager' and Node = '@Node' and
    AlertKey = '@AlertKey';

update alerts.status via '@Identifier' set Severity = 0;
```

Generic Automation (pre 3.6) New Action

- Generic Clear

  update alerts.status set Severity = 0, ***LastClearTime = @LastOccurrence, Duration = (Duration + (@LastOccurrence – LastOccurrence))*** where Severity > 0 and Type = 1 and LastOccurrence < @LastOccurrence and AlertGroup = '@AlertGroup' and Manager = '@Manager' and Node = '@Node' and AlertKey = '@AlertKey';

  update alerts.status via '@Identifier' set Severity = 0;

Modified Generic Automation (3.6)

- A bit more complicated due to three step automation used for New High Performance Generic Clear
- Need to add the new fields to end of alerts.generic_clear_problem_events table in the sql file
  - Duration int,
  - LastClearTime time,

Generic Automation (3.6) Trigger 1

- AA_GenericClear_Populate_Problems_Table

```
delete from alerts.generic_clear_problem_events;

select * from alerts.status where Type = 1 and Severity > 0 and
    Manager in (select Manager from alerts.status where Type = 2
    and Severity > 0) and AlertGroup in (select AlertGroup from
    alerts.status where Type = 2 and Severity > 0) and AlertKey in
    (select AlertKey from alerts.status where Type = 2 and Severity
    > 0);
```

Generic Automation (3.6) Old Action 1

- **GenericClear_Populate_Problem_Table**

```
insert into alerts.generic_clear_problem_events values
    ('@Identifier', @LastOccurrence, '@AlertKey', '@AlertGroup',
    '@Manager', 0);
```

Generic Automation (3.6) New Action 1

- **GenericClear_Populate_Problem_Table**

```
insert into alerts.generic_clear_problem_events values
    ('@Identifier', @LastOccurrence, '@AlertKey', '@AlertGroup',
    '@Manager', 0, @Duration, @LastClearTime);
```

Generic Automation (3.6) Trigger 2

- **AB_GenericClear_Correlate_Problems_Table**

  select * from alerts.status where Type = 2 and Severity > 0;

Generic Automation (3.6) Old Action 2

- **Generic_Clear_Correlate_Problems_Table**

```
update alerts.generic_clear_problem_events set Resolved = 1
    where LastOccurrence < @LastOccurrence and AlertGroup =
    '@AlertGroup' and AlertKey = '@AlertKey' and Manager =
    '@Manager' and Node = '@Node';
```

Generic Automation (3.6) New Action 2

- **Generic_Clear_Correlate_Problems_Table**

  update alerts.generic_clear_problem_events set Resolved = 1, *LastClearTime = @LastOccurrence, Duration = (Duration + (@LastOccurrence – LastOccurrence))* where LastOccurrence < @LastOccurrence and AlertGroup = '@AlertGroup' and AlertKey = '@AlertKey' and Manager = '@Manager' and Node = '@Node';

Generic Automation (3.6) Trigger 3

- **AC_GenericClear_Correlate_Status_Table**

```
select * from alerts.generic_clear_problem_events where
    Resolved = 1;
```

Generic Automation (3.6) Old Action 3

- **Generic_Clear_Correlate_Status_Table**

```
update alerts.status via '@Identifier' set Severity = 0 where
    LastOccurrence <= @LastOccurrence;
```

Generic Automation (3.6) New Action 3

- Generic_Clear_Correlate_Status_Table

```
update alerts.status via '@Identifier' set Severity = 0,
    LastClearTime = @LastClearTime, Duration = @Duration
    where LastOccurrence <= @LastOccurrence;
```

Results

- @LastClearTime in Problem event will always contain the time the event was last cleared
- @Duration will contain total number of seconds between Problem and Resolution

A Question…

- Why do we need @Duration?
- Couldn't I just calculate @Duration using @FirstOccurrence and @LastClearTime of the Problem event???
- Not Exactly…

Example – farkle link goes down

- **Node farkle LinkDown event at 12:00:00 (Type = Problem)**
- **Node farkle LinkUp event at 12:03:00 (Type = Resolution)**
- **Generic Automation sets @LastClearTime = 12:03:00 & @Duration = 180 seconds in LinkDown event. Generic Automation clears both events**

Example – whoops, it happened again…

- Another LinkDown event at 12:08:30 – This will deduplicate into first event, update @Tally to 2, @Severity = 4, @LastOccurrence = 12:08:30
- Another LinkUp event at 12:10:01 – This will deduplicate into first LinkUp event, update @Tally to 2, @Severity = 1, @LastOccurrence = 12:10:01
- Generic Automation sets @LastClearTime = 12:10:01 & @Duration = 271 seconds (180 + 91) in LinkDown event. Generic Automation clears both events

Read it this way: "Between 12:00:00 and 12:10:01 farkle experienced 2 link outages lasting for a total of 4 minutes 31 seconds."

The Cool Part

Anything and everything that uses the Generic Automation now has @LastClearTime and @Duration set/calculated!

Completing Report

- With these changes implemented, the Problem record actually contains all of the base information we want for our report
- Set the @ReportFlag field to 1 for Problem events you want to show up on the report
- Setup Reporter Gateway with filter (@ReportFlag = 1) to send these events to your database of your choice
- Use favorite reporting tool/script/et all to pull records from database to create report
- Manager = Happy! A good thing.

Enhancements

- Modify @Summary or create new field to make a better description of problem on Report
- Modify DeleteClears timings to determine length of “window”
- Saving the @LastClearTime each time the Problem event is cleared
- If using Trouble Ticket Gateway, your report could include ticket # and text from ticket system (by forwarding appropriate alert.journal entries)
- Could create real-time Report screen in Webtop

Special Thanks

- Martha Rhoads
- Deanna Hall

Questions

???