



Automation Reference

This appendix provides the background and technical information you need to start building Cisco Info Center automations. It includes sample automations, as well as information about the default automations installed with Cisco Info Center. The procedures in this appendix are best practices to solve some specific problems. There may be various solutions to some situations.

This appendix contains the following sections:

- [Types of Automation Triggers, page B-1](#)
- [Sampling and Timing, page B-4](#)
- [Generic Clears, page B-5](#)
- [Clearing Old Alerts, page B-15](#)
- [Clearing Journals and Details Tables, page B-19](#)
- [Highlighting Unacknowledged Alerts, page B-20](#)
- [Highlighting Frequent Alerts, page B-21](#)
- [Removing Alert Highlighting, page B-22](#)
- [Suppressing Related Alerts, page B-23](#)
- [E-mail Automations, page B-24](#)
- [SNMP Generic Traps, page B-27](#)
- [Link Up and ISDN, page B-30](#)
- [Cisco Info Center Customized Automations, page B-33](#)
- [Case Example One - Batching and Sending Alerts, page B-45](#)
- [Case Example Two - Acknowledging Alerts with Sound, page B-47.](#)

For an overview of automations and information about using the Configuration Manager to create automations, see [Chapter 4, “Automation.”](#)

Types of Automation Triggers

The trigger type is determined by how the action is called. There are four types of triggers:

- Edge
- Level

- Delayed Edge
- Delayed Level.

The following sections describe each type of trigger.

Notes About Trigger Types

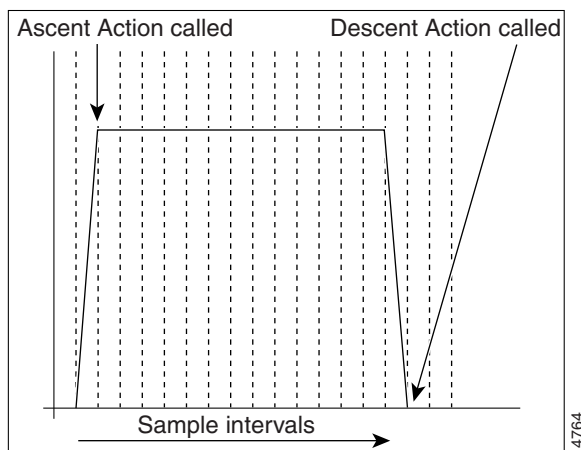
Consider the following for each trigger type:

- A trigger may have either or both ascent and descent actions.
- Triggers are primed when the number of alerts returned by the trigger SQL statement match the Row Count condition. For information about the Row Count condition, see [Entering a Trigger Condition, page 4-8](#).
- There are situations where you may not want an automation to start as soon as a trigger is primed, but may want the trigger to be primed for a number of sample intervals before calling the ascent action. This type of behavior is called a delayed trigger.
- The number of sample intervals before triggering is called the threshold.

Edge Trigger Behavior

An edge trigger calls the ascent action when the trigger goes from un-primed to primed. When the trigger is no longer primed, the descent action is called.

Figure B-1 Edge Trigger

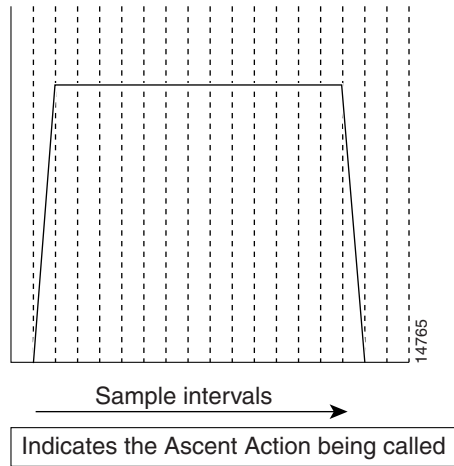


This type of trigger allows a single response to a situation.

Level Trigger Behavior

A level trigger calls its ascent action every time the filter matches any number of rows. A descent action, even if specified, is never called.

Figure B-2 Level Trigger

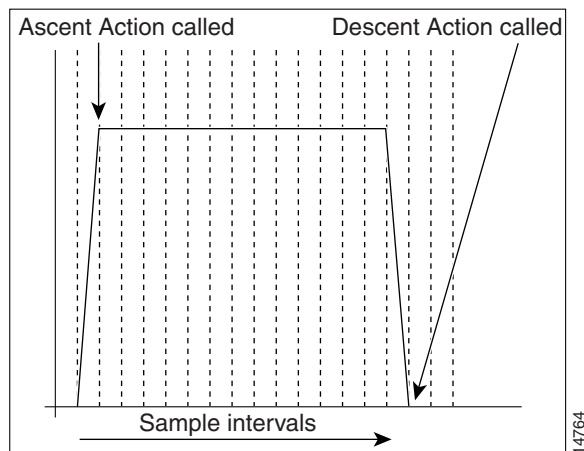


This type of trigger allows for multiple responses to a situation. The responses stop only when the situation has been resolved.

Delayed Edge Trigger Behavior

A delayed edge trigger delays triggering until the threshold has been exceeded. In [Figure B-3](#), the threshold has been set to 5, so the trigger must be primed for five samples in a row before firing the ascent action. It then waits for the trigger to evaluate false, when it calls its descent action.

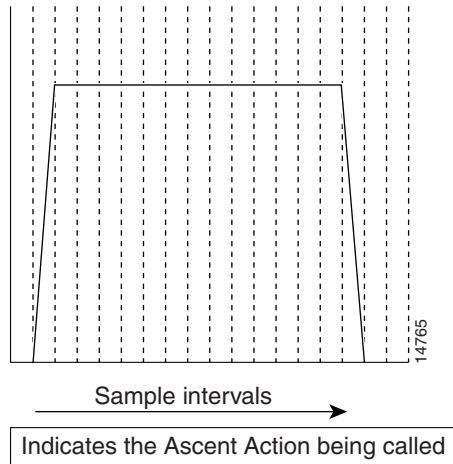
Figure B-3 Delayed Edge Trigger



Delayed Level Trigger Behavior

A delayed level trigger delays triggering until the threshold has been exceeded. Once the threshold has been met, the trigger calls the ascent action for every sampling where the trigger is still primed.

Figure B-4 Delayed Level Trigger



This type of trigger allows for the delaying of an action until it is absolutely necessary.

Sampling and Timing

Automation triggers are checked at regular intervals to see if the trigger should fire. The sample rate controls the check. For example, the sample rate could be set to 60 seconds. This means every 60 seconds, the SQL inside the automation trigger is executed and the results are evaluated to see if the trigger should fire. Therefore, every 60 seconds, there is a query on the database.

Sample Rate of 0

A sample rate of **0** is a special case, and one to use with caution. When set to **0**, the sample rate is ignored and the trigger becomes real-time. It is not constantly executed after every insert or update to the alerts.status table, but the trigger is evaluated in the second after every new alert arrives in the Cisco Info Server.

Note the trigger performs an SQL query on the entire database which could adversely affect performance.

For example, if you expect to have hundreds of alerts arriving constantly, a real-time trigger is a bad solution; if 100 alerts arrive over ten seconds, the SQL is executed 10 times, and the trigger and action may use the majority of the computer's processing power. This means the Cisco Info Server has less time to respond to users and their alert lists, less time to process other automations, and, in the worst case, not enough time to finish the action before the next alerts come in and set it off again.

With a high alert load, it is better to use a standard, non-zero sample rate which is set to a value that gives you predictable performance, and is not dependent on the number of arriving alerts.

There are occasions when real-time triggers are appropriate, but these are rare. Often people may use a real-time trigger to change incoming data from a Cisco Info Mediator; for example, to swap fields around or change severity values. This is an inappropriate use of real-time triggers. The Cisco Info Mediator rules file should be used as it permits great flexibility in what gets sent to the Cisco Info Server and how it is formatted. Changes made to the Cisco Info Mediator rules file do not impact the performance of the Cisco Info Server, and in cases where alerts need to be discarded, can improve Cisco Info Server performance.

Choosing a Sampling Rate

There is a simple rule you can use to calculate your sampling rate. For example, if something happens at approximately n seconds, sample at $n/2$. If something happens every 30 seconds (or you want the system to handle it as soon as possible), sample at 15 seconds.

Low sample rates have a drawback similar to the problem when using real-time triggers. For example, if you have a one second sample rate you can run out of time for the Cisco Info Server to service users, gates, and other connections. Avoid low sample rates, and use the above rule to calculate how fast you need to sample.

When choosing a sampling rate, you must also consider any other automations in the Cisco Info Server. For example, you may have 10 automations, each of which fires every 60 seconds. In this situation you will have 10 automations attempting to fire simultaneously while the rest of the 60 second time slot is left free. Multiple automations attempting to fire simultaneously can slow the Cisco Info Server drastically.

To avoid this problem, choose a different time interval close to 60 seconds for each automation. For example, the automations could be staggered at two second intervals firing at 52, 54, 56, 58, 60, 62, 64, 66, 68, and 70 seconds.

Generic Clears

Within Cisco Info Center automations, the most frequently used type of automation is one that will clear a negative alert upon receipt of a related positive alert. This type of automation is typically used for clearing a Link Fail alert when a corresponding Link OK alert is received. An example automation is:

```
Trigger Condition      select * from alerts.status where Summary like 'Link Up';

Action Data Effect    update alerts.status set Severity=0 where
                       LastOccurrence <= @LastOccurrence
                       and Node = '@Node'
                       and AlertKey = '@AlertKey'
                       and Summary = '@Summary';
```

This automation detects a Link Up alert and clears any Link Down alert which came from the same node, if it was matched on the **AlertKey** field, and was older. There should be a similar automation detecting Node Up and clearing Node Down. This situation can be applied to many different management platforms and applications, resulting in many different automations to handle them. All of these automations require processing resources from the Cisco Info Server to find alerts which require action, execute automations, and update clients.

An alternative method of resolving this problem would be to create one automation to recognize many different types of association. This is possible as the only component that differs between automations is often the text or fault code being searched for. This requires a generic way of looking at certain faults.

Analyzing Alerts for Generic Clear Automations

By closely examining alert data, it is possible to create generic clear automations for the majority of standard up/down alerts. To identify alerts that negate one another, you can start by asking the following questions:

- Is this a problem or a clear?
- From which network element did the alert originate?
- Which other alerts is this alert associated with?

This information could be obtained and used in a model as follows:

Is This a Problem or a Clear?

The **Type** field in the **alerts.status** table can contain one of the following values for an alert:

1. (0) Ignore—the default value; these alerts are ignored by a generic clear automation.
2. (1) Problem—indicates a problem alert that can be cleared when a corresponding alert with a **Type** value of **2** is received.
3. (2) Clear—clears corresponding alerts with a **Type** value of **1**.



Note

Values **3-13** are also valid for the **Type** field, but are used for special cases. See [@Type Field, page B-7](#).

From Which Network Element Did It Originate?

To locate alerts that originated from the same network element, find matching values for the following **alerts.status** table fields:

- **Manager**—narrows the scope to one management domain
- **Node**—further narrows the scope to a particular node within the domain
- **AlertKey**—narrows the scope to a particular element within the domain of the node. This is usually one of the following:
 - Interface number or name
 - Shelf/card/slot/port combination
 - Application name
 - System resource ID; for example, disk name/printer ID.

Which Other Alerts Is This Alert Associated With?

To locate alerts that correspond to one another, use the **alerts.status** table **AlertGroup** field. Match the values in the **AlertGroup** field to other alerts that will clear or be cleared by this alert.

For example, Link Up, Link Down, and Link Protocol Fail alerts might all have an **AlertGroup** value of Link Warnings.

After identifying whether the alert is a problem or a clear, and then locating alerts with matching identification information, you should be able to create a generic clear automation for most link up/link down alert types.

Fields Used

This section describes the Cisco Info Server fields used within the automations after fault-related information has been gathered.

@Type Field

This field can have one of the values listed in [Table B-1](#).

Table B-1 @Type Field Values

Value	Value Description	Description
0	None	Type not set.
1	Problem	A problem that is a fault for which a resolution is expected.
2	Resolution	A resolution; that is, this fault clears another fault which may already exist in the alerts.status table, and would have a type value of 1 .
3	Visionary problem	A problem received from Visionary.
4	Visionary resolution	A resolution received from Visionary.
7	ISMs new alarm	A problem received from ISMs.
8	ISMs old alarm	A problem received from ISMs.
11	More severe	Indicates the perceived severity in the current alarm is more severe than that reported in any of the related, outstanding alarms.
12	Less severe	Indicates there is at least one related, outstanding alarm with a higher severity than the current alarm.
13	Information	For informational purposes only.

@AlertGroup Field

This field should be populated with the alert identifier. [Table B-2](#) shows an example for the HP OpenView Network Node Manager.

Table B-2 @AlertGroup Field Values for HP OpenView Network Node Manager

Actual Alert Type	Alert Group Value	Type Value
OV_IF_DOWN	OV_IF_WARNING	1
OV_IF_MARGINAL	OV_IF_WARNING	1
OV_IF_WARNING	OV_IF_WARNING	1
OV_IF_CRITICAL	OV_IF_WARNING	1
OV_IF_MAJOR	OV_IF_WARNING	1
OV_IF_UP	OV_IF_WARNING	2

@Manager, @Node, and @AlertKey Fields

These three fields are used to uniquely identify an enterprise element.

@LastOccurrence Field

This standard time stamp field is used to ensure a problem alert is not cleared by a resolution alert which occurred before the problem alert.

Rules File

Using the example from the HP OpenView Network Node Manager Cisco Info Mediator (see [@AlertGroup Field, page B-7](#)), the data that must be defined in the Cisco Info Mediator rules file is shown in [Table B-3](#).

Table B-3 Cisco Info Mediator Rules Fields

Field	Description
Node	This field is usually set in the rules file.
Manager	This field is usually set in the rules file with a unique name for each Cisco Info Mediator.
AlertKey	<p>This field must be added to the rules file. This information is required whether the clearing is performed on an individual alert basis or on a generic basis.</p> <p>For example, depending on the type of Cisco Info Mediator, the information can be obtained in one of the following ways:</p> <ul style="list-style-type: none"> • A field that always contains interface or similar data • An extract() statement to extract the details from a text message • assignment of particular varbinds from an SNMP trap; for example, @AlertKey = \$1 + \$4 + \$5.
Type	<p>This field needs to be set to either 1 or 2 to identify either a problem or a resolution; for example:</p> <p>@Type = 1 (for a problem)</p> <p>@Type = 2 (for a resolution)</p> <p>@Type = 3600 (for an alert which should be deleted if it becomes one hour old)</p>
AlertGroup	This field needs to be set to associate a group of alerts. See the example for OV_IF_* alerts in Table B-2 on page B-7 .

Generic Automations

With the fields set, automations can now correlate problems and resolutions independently of the source of the alerts. The automations can also expire alerts based on an expiry time in the **Type** field. [Table B-4](#) lists the automations used to clear problems and resolutions, which are further described in the following sections.



Note

The following automations are included with a default Cisco Info Center installation.

Table B-4 Generic Clear Automations

Automation	Purpose
GenericClear	Detects and clears (sets Severity to 0) resolution alerts (@Type = 2), and clears any associated problems (@Type = 1) that are older and have matching manager, node, alert key, and alert group entries.
High Performance GenericClear	Identical purpose to the standard generic clear, optimized for systems with a large number of alerts.
Expire	Detects any alerts of @Type > 10 and clears them if they are older than the contents of @Type in seconds.
DeleteClears	Deletes records that have been cleared (Severity = 0) and are over two minutes old.

**Note**

When you create a new Cisco Info Center version 3.6 installation, the high performance generic clear automation is installed by default. In versions 3.4.1 and prior upgraded installations, the standard generic clear automation is installed by default. For versions 3.4.1 and prior, the import file for the high performance generic clear is `$OMNibus/unsupported/new_generic_clear.auto`.

Generic Clear

The generic clear automation detects and clears (sets **Severity** to **0**) resolution alerts (**@Type = 2**) and clears any associated problems (**@Type = 1**) which are older and have matching manager, node, alert key, and alert group entries.

Trigger

This sections describes the entry names and values used in a generic clear trigger.

Trigger Name	GenericClear
Sample Rate	5
Ascent Action	GenericClear
Descent Action	none
Condition	select * from alerts.status where Type = 2 and Severity > 0;
Type	Level
Threshold	0
Execution Mode	For each matched row
Row Count	>0

Action

Table B-5 Generic Clear Action

Action	GenericClear
Data Effect	<pre>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 with '@Identifier' set Severity = 0;</pre>
External Effect	none

High Performance Generic Clear

In the standard generic clear automation, the number of where clause evaluations during the executions of the action is the key sizing factor for performance. The high performance generic clear automations reduce the number of where clause evaluations by using a temporary table called **alerts.generic_clear_problem_events** to store targeted resolved problems.



Note

The **alerts.generic_clear_problem_events** table is emptied before each execution of the generic clear automation; it must not be checkpointed.

The high performance generic clear consists of three triggers, each with corresponding actions. The triggers must be run in sequential order, which is accomplished by naming them in increasing lexicographical order.

The sample rate in each automation must also be equal.

Trigger

This trigger first empties the **alerts.generic_clear_problem_events** table. It then selects all problems for which there are resolutions not yet processed.

The corresponding action inserts the selected entries into the **alerts.generic_clear_problem_events** table for problem/resolution correlation.

Trigger Name	AA_GenericClear_Populate_Problems_Table
Sample Rate	5
Ascent Action	GenericClear_Populate_Problems_Table
Descent Action	none
Condition	select * from alerts.status where Type = 2 and Severity > 0;

Type	Level
Threshold	0
Execution Mode	For each matched row
Row Count	>0

Table B-6 shows the corresponding action.

Action

This action populates the problems table, as shown in Table B-6.

Table B-6 *GenericClear_Populate_Problems_Table Action*

Action	GenericClear_Populate_Problems_Table
Data Effect	<code>insert into alerts.generic_clear_problem_events values ('@Identifier', @LastOccurrence, '@AlertKey', '@AlertGroup', '@Manager', 0);</code>
External Effect	none

Trigger

This trigger selects all the resolutions available in the `alerts.status` table.

The corresponding action processes the `alerts.generic_clear_problem_events` table and sets the **Resolved** field to **1**. It also updates the associated resolution **Severity** to **0** in the `alerts.status` table.

Trigger Name	AB_GenericClear_Correlate_Problems_Table
Sample Rate	5
Ascent Action	Generic_Clear_Correlate_Problems_Table
Descent Action	none
Condition	<code>select * from alerts.status where Type = 2 and Severity > 0;</code>
Type	Level
Threshold	>0
Execution Mode	For each row
Row Count	>0

Table B-7 shows the corresponding action.

Action

This action correlates resolved events in the problems table.

Table B-7 *GenericClear_Populate_Problems_Table Action*

Action	GenericClear_Populate_Problems_Table
Data Effect	<pre>update alerts.generic_clear_problem_events set Resolved = 1 where LastOccurrence < @LastOccurrence and AlertGroup = '@AlertGroup' and AlertKey = '@AlertKey' and Manager = '@Manager' and Node = '@Node'; update alerts.status with '@Identifier' set Severity = 0;</pre>
External Effect	none

Trigger

This trigger scans the **alerts.generic_clear_problem_events** table looking for entries with the **Resolved** field set to **1**. Once selected, the corresponding action sets the **Severity** to **0** for the associated event in the **alerts.status** table.

Trigger Name	AC_GenericClear_Correlate_Status_Table
Sample Rate	5
Ascent Action	Generic_Clear_Correlate_Status_Table
Descent Action	none
Condition	select * from alerts.generic_clear_problem_events where Resolved = 1;
Type	Level
Threshold	>0
Execution Mode	For each row
Row Count	>0

[Table B-8](#) shows the corresponding action.

Action

This action sets event **Severity** to **0** in the **alerts.status** table for all correlated events in the problems table.

Table B-8 *GenericClear_Populate_Problems_Table Action*

Action	AC_GenericClear_Correlate_Status_Table
Data Effect	update alerts.status with '@Identifier' set Severity = 0 where LastOccurrence <= @LastOccurrence;
External Effect	none

Expiring Alerts

This automation is used for alerts that should be indicated to operations staff, but would be out of date after a certain amount of time (for example, capacity counters, which would re-notify after a given period). By assigning a value in seconds to the **ExpireTime** field, alerts are given an age threshold which, when reached, causes the alert to be deleted.

For example, setting **ExpireTime=3600** causes the alert to be deleted if it becomes over one hour old; therefore, if the alert re-occurs every 59 minutes, it is never deleted.

Trigger

Trigger Name	Expire
Sample Rate	60
Ascent Action	Expire
Descent Action	none
Condition	select * from alerts.status where ExpireTime > 0 and Severity > 0;
Type	Level
Threshold	0
Execution Mode	For each matched row
Row Count	>0

Table B-9 shows the corresponding action.

Action

Table B-9 Expiring Alerts Action

Action	Expire
Data Effect	update alerts.status with '@Identifier' set Severity = 0 where LastOccurrence < (getdate - @Type);
External Effect	none

Deleting Alerts

This automation deletes records that have been cleared (**Severity = 0**) and are over two minutes old.

Trigger

Trigger Name	DeleteClears
Sample Rate	60
Ascent Action	none
Descent Action	none
Condition	delete from alerts.status where Severity=0 and LastOccurrence < (getdate - 120)
Type	Level
Threshold	0
Execution Mode	Once only
Row Count	>0

Clearing Old Alerts

When an alert has been marked as clear, it is not automatically deleted. The deletion of alerts is a function of the automation system. This requires you to define a policy for the deletion of alerts and implement it in the automation system. The following sections provide several suggestions and their implementations.

Deleting by Age - Part One

The simplest policy is to expire alerts by how old they are. The first question to ask is: When is an alert considered “old”? There are a number of time fields in an alert that can be used, including **FirstOccurrence**, **LastOccurrence**, and **StateChange**. These times values are stored in seconds (starting at Midnight January 1st 1970).

The requirement is to measure when “now” is and use that as the basis to determine the age of an alert. The Cisco Info Server SQL statement **getdate** returns the time in seconds whenever a Cisco Info Server SQL statement is executed. Using a simple calculation, it is then possible to calculate the time five minutes ago (**getdate - 300**) or the time 24 hours ago (**getdate -86400**). For example, to select all alerts that last occurred five minutes or more ago, enter the following command:

```
select * from alerts.status where LastOccurrence < (getdate - 300);
```

A typical scenario for clearing old alerts would be to delete any cleared alerts which occurred more than five minutes ago. The following tables contain the trigger and action for this scenario.

Trigger

Trigger Name	FindOldAlerts
Sample Rate	60
Ascent Action	DeleteAlert
Descent Action	none
Condition	select * from alerts.status where Severity=0 and LastOccurrence < (getdate - 300);
Type	Level
Threshold	none
Execution Mode	For each
Row Count	>0

[Table B-10](#) shows the corresponding action.

Action

Table B-10 Deleting By Age Action (Part 1)

Action	DeleteAlert
Data Effect	delete from alerts.status via @Identifier;
External Effect	none

Effect

This trigger and action deletes alerts, but possibly sooner than you desire. Consider the scenario where a single alert comes in at midnight and in the morning, the operations team fix the problem and clear the alert. Because the alert last occurred at midnight, as soon as it is cleared, when the automation is executed (every 60 seconds) the alert is deleted even though it has only been cleared for 60 seconds.

Deleting by Age - Part Two

In the previous example, alerts may have been deleted too quickly. This was because the **LastOccurrence** field was used to test the age of the alert. What may be more useful is testing the time the alert was last changed. For this you can use the **StateChange** field, which reflects the time the alert was last modified by an automation or user.

Trigger

Trigger Name	FindOldAlertsTwo
Sample Rate	60
Ascent Action	DeleteAlertTwo
Descent Action	none
Condition	select * from alerts.status where Severity=0 and StateChange < (getdate - 300);
Type	Level
Threshold	none
Execution Mode	For each
Row Count	>0

[Table B-11](#) shows the corresponding action.

Action

Table B-11 Deleting By Age Action (Part 2)

Action	DeleteAlertTwo
Data Effect	delete from alerts.status via @Identifier;
External Effect	none

Effect

Five minutes after an alert is cleared, it will be deleted. This automation still uses a query and multiple deletes and can be improved for performance. This is covered in the next example.

Deleting by Age - Part Three

The previous example has one drawback. If there are many expired alerts then many SQL commands are executed to delete the various alerts one at a time. A more efficient solution would be to use one trigger that detects if any alerts need deletion. If so, then a delete command could be used once (by setting the Execution Mode to Once Only). The delete command should then be set up to delete any old alerts with a “where” clause which is the same as the trigger. The following tables contain the automation information.

Trigger

Trigger Name	FindOldAlertsThree
Sample Rate	60
Ascent Action	DeleteAlertThree
Descent Action	none
Condition	select * from alerts.status where Severity=0 and StateChange < (getdate - 300);
Type	Level
Threshold	none
Execution Mode	Once Only
Row Count	>0

[Table B-12](#) shows the corresponding action.

Action

Table B-12 Deleting By Age Action (Part 3)

Action	DeleteAlertThree
Data Effect	delete from alerts.status where Severity=0 and StateChange < (getdate - 300)
External Effect	none

Deleting by Age - Part Four

There is another way to simplify the entire process. In the previous example, the same query is performed twice, once in the trigger, once in the action. Whenever the trigger fires, the same query is executed again. Using a delete command in the trigger condition makes the operation more efficient. The trigger is unable to fire Ascent and Descent actions. This allows SQL commands to be executed at regular intervals.

Trigger

Trigger Name	FindOldAlertsFast
Sample Rate	60
Ascent Action	none
Descent Action	none
Condition	delete from alerts.status where Severity=0 and StateChange < (getdate - 300);
Type	Level
Threshold	none
Execution Mode	Once Only
Row Count	>0

Effect

With this single trigger, the SQL condition is executed every 60 seconds, deleting aged alerts. As only one command is ever executed, this is a stable and efficient automation.

To complete this scenario there are two additional elements required. See [Clearing Journals and Details Tables, page B-19](#) for more information.

Clearing Journals and Details Tables

When a user deletes an alert in the Cisco Info Server using the Event List, the Event List not only clears the alert, it also deletes related journal and details entries. When a delete is carried out by automation, the tables are not cleaned-up in the same way. Left unmanaged, the tables will grow continuously. This would consume space and would eventually have an impact on Cisco Info Server performance.

The most efficient way to handle these tables is to implement housekeeping automations to perform a clean-up. These automations are incorporated within the standard product and are active upon installation. They are reproduced here to show an example of this type of automation. They also show example usage of an SQL sub-query statement and use of tables other than the **alerts.status** table.

The journals trigger deletes all entries in the journals table having a serial number not in the **alerts.status** table. The details trigger deletes all entries in the details table having an identifier not in the **alerts.status** table.



Note

The **CleanDetailsTable** and **CleanJournalTable** triggers are included with a default Cisco Info Center installation.

Journals Trigger

Trigger Name	CleanJournalTable
Sample Rate	360
Ascent Action	none
Descent Action	none
Condition	delete from alerts.journal where Serial not in ((select Serial from alerts.status));
Type	Level
Threshold	none
Execution Mode	Once Only
Row Count	>0

Details Trigger

Trigger Name	CleanDetailsTable
Sample Rate	3600
Ascent Action	none
Descent Action	none
Condition	delete from alerts.details where Identifier not in ((select Identifier from alerts.status));
Type	Level
Threshold	none
Execution Mode	Once Only
Row Count	>0

Highlighting Unacknowledged Alerts

You can cause critical alerts to flash on the Event List if they are not acknowledged within ten minutes. This trigger also sets the alert's escalation level to **1** (escalated).



Note

The **FlashNotAck** trigger is included with a default Cisco Info Center installation.

Trigger

Trigger Name	FlashNotAck
Sample Rate	31
Ascent Action	none
Descent Action	none
Condition	update alerts.status set Flash = 1, SuppressEscl = 1 where Flash = 0 and Acknowledged = 0 and Severity = 5 and FirstOccurrence <= (getdate -600);
Type	Level
Threshold	none

Execution Mode	For each matched row
Row Count	>0

Highlighting Frequent Alerts

Another time-based filtering mechanism is to make any alert occurring more frequently than a given threshold flash on the Event List to highlight the alert.

This example uses the **Poll** field to hold a value, which is the threshold for the average number of seconds in between occurrences of the alert. If the average gap between them is less than the **Poll** field value, the alert item flashes in the Event List. The trigger resets the flashing mechanism before execution. This ensures that if the occurrences of the alert stop, then the flashing also stops (when the occurrences occur below the threshold).

Before using this automation, a statement should be added to the rules file, so the Cisco Info Mediator sets the poll value. In this example, enter:

```
@Poll = 6
```

In this case, the value is set to **6**, which sets an average of less than six seconds between alerts.

If you do not want to edit the rules file, you should set the poll value for all alerts by changing the first trigger line. The following example shows this trigger. If you set the poll value in the rules file, omit this statement. An advantage of setting the poll value in the rules file is it allows different intervals for different alerts.

Trigger

Trigger Name	FlashHighRate
Sample Rate	60
Ascent Action	none
Descent Action	none
Condition	update alerts.status set Poll = 6, Flash = 0; select * from alerts.status where Tally > 1 and FirstOccurrence < (getdate - 60);
Type	Level
Threshold	none
Execution Mode	For each matched row
Row Count	>0

Action

Table B-13 Highlighting Frequent Alerts Action

Action	FlashHighRate
Data Effect	update alerts.status via @Identifier set Flash = 1 where Poll >= ((getdate-@FirstOccurrence) / @Tally);
External Effect	none

Effect

The first trigger condition statement resets the flash status for all alerts and sets the poll values. The second trigger condition statement selects any alerts which have occurred more than once, and are at least one minute old. The last condition is to allow for a reasonable sample rate.



Note

You will receive a warning when applying the trigger as it affects all rows in the database. This is correct behavior, because the first statement does not have a condition.

Removing Alert Highlighting

You can remove Event List flashing and any escalation level on acknowledged or cleared alerts.



Note

The **EscalateOff** trigger is included with a default Cisco Info Center installation.

Trigger

Trigger Name	EscalateOff
Sample Rate	6
Ascent Action	none
Descent Action	none
Condition	update alerts.status set Flash = 0, SuppressEscl = 0 where ((Flash = 1 or SuppressEscl > 0) and Acknowledged = 1) or (Severity = 0);
Type	Level
Threshold	none

Execution Mode	Only once
Row Count	>0

Suppressing Related Alerts

In certain situations, alerts become irrelevant or redundant if other related alerts are received. For example, the fact an interface is down can be overshadowed by the fact the whole node is down. This automation is an example of how to perform an efficient sweep of alerts and delete alerts pertaining to any interface failure for which a node failure also exists.

The automation uses a sub-query, similar to the housekeeping automation for clearing up journals and details. The use of a sub-query is much more efficient than referencing each alert separately in an action. The subquery does not allow complex field comparisons, and therefore is only suited to certain tasks.

Trigger

Trigger Name	SuppressIfevents
Sample Rate	60
Ascent Action	none
Descent Action	none
Condition	delete from alerts.status where Summary like 'Link Down' and Node in ((select Node from alerts.status where Summary like 'Node Down'));
Type	Level
Threshold	0
Execution Mode	Only once
Row Count	>0

When executed, the statement within the double parentheses is executed first. This generates a list of node values for which a **Node Down** alert exists. The main statement is then executed, deleting all alerts reporting **Link Down** whose node value appears in the list generated by the subquery.

E-mail Automations

This section contains information about automations you can use to send e-mail.

Sending E-mail

When a particular alert occurs, it may be useful to generate e-mail for particular users. With most e-mail systems, it is possible to have e-mail forwarded to managers or on-call support staff or even to a text pager. This can be achieved using external actions in Cisco Info Center.

Before installing the automation, it is necessary to create a shell script to compose and send the e-mail. The following is an example script:

```
#!/bin/sh
# Generic script for forwarding mail from Netcool OMNIBus
#
# Parameters to be passed from OMNIBus are;
# $1 - @Node - Name of node
# $2 - @Severity - Severity of problem
# $3 - "Message" - Message header text
# $4 - "user" - User to receive mail
# $5 - @Summary - Summary text
#
cat <<EOF >/tmp/tmp.$!

This message refers to node $1 which has the following problem;

$5

The Severity is $2
Send by the Netcool OMNIBus automation system
EOF
mailx -s $3 $4 </tmp/tmp.$!
rm /tmp/tmp.$!
```

This script should be installed and made executable. For this example, it resides in:

\$OMNIHOME/utils/nco_mail

The host where this file is installed must be running the Process Control system. In this example, it resides on a host called **bester**.

Trigger

Trigger Name	MailOnCritical
Sample Rate	60

Ascent Action	SendMailNow
Descent Action	none
Condition	select * from alerts.status where Severity=5 and Grade=0;
Type	Level
Threshold	none
Execution Mode	For each
Row Count	>0

**Note**

The **MailOnCritical** trigger is included with a default Cisco Info Center installation.

Action**Table B-14 Send Mail Now Action**

Action	SendMailNow
Data Effect	update alerts.status via '@Identifier' set Grade=1;
External Effect	Executable: \$OMNIHOME/utils/nco_mail Arguments: @Node @Severity NCO_MAIL_MESSAGE root \'@Summary\' Host: bester Effective UID: 0

Effect

When an alert of a **Severity** value **5** (Critical) with a **Grade** field set to **0** occurs, the trigger fires the action. The action does the following:

- the data effect sets the **Grade** field to **1** to stop the trigger from firing again
- the external effect calls the shell script with the parameters it needs to send mail. In this case, the mail is sent to the user **root**.

Delaying the Mail

In the previous example, mail was sent as soon as an alert appeared. You may wish to configure the system to let you know, after a period, if an alarm is still present. This can be achieved by modifying the trigger to use a delayed edge. In this example, the trigger is configured to perform the action after five minutes.

Delayed triggers depend on the threshold and sample rate. When an alert matches the condition, the threshold comes into play; with a threshold of **5**, the condition must be fulfilled five times on the next evaluations of the trigger. So where the sample rate is **60** and the threshold is **5**, the trigger condition must be fulfilled five times, which equates to five minutes (60 seconds multiplied by 5 equals 300 seconds, or five minutes).

Note this trigger does not distinguish between the critical alerts, it just registers there are critical alerts. If an alert arrives as critical in the first minute, followed by a second alert in the second minute, followed by the first alert being deleted, the clock will continue ticking after being started by the first alert. This automation is an example, and it is recommended you make the **where** clause much more specific.

Note also this trigger uses **Once only** as its execution mode.

Trigger

Trigger Name	MailOnCriticalAfter5
Sample Rate	60
Ascent Action	SendMailAfter5
Descent Action	none
Condition	select * from alerts.status where Severity=5 and Grade=0;
Type	Delayed Level
Threshold	5
Execution Mode	Once only
Row Count	>0

Action

Table B-15 Send Mail Now Action

Action	SendMailNow
Data Effect	update alerts.status via '@Identifier' set Grade=1;
External Effect	Executable: \$OMNIHOME/utils/nco_mailafter5 Arguments: NCO_CRITICAL_MAIL root Host: bester Effective UID: 0

Shell Script

```
#!/bin/sh
#
# Alternative script for mailing from Netcool OMNIBus
#
# The only parameters passed are
# $1 - "Message" - Message header text
# $2 - "user" - User to receive mail
#
cat <<EOF >/tmp/tmp.$!
There are critical alerts in the ObjectServer which have existed for more than 5 minutes.
EOF
mailx -s $1 $2 </tmp/tmp.$!
rm /tmp/tmp.$!
```

Effect

When any critical alerts have existed in the system for more than five minutes, a mail message is sent to the user **root**. The trigger fires the action as **Once only**.

Paging

If you want to page someone rather than send mail, the automation works similarly to the automatic e-mail automation. The only change needed is to the shell script, which needs to be modified to send a message to the paging system.

SNMP Generic Traps

There are many generic ways to handle SNMP traps within Cisco Info Center automation. There are six generic types of SNMP traps, as shown in [Table B-16](#).

Table B-16 Types of SNMP Generic Traps

Value	Description
0	Cold start
1	Warm start
2	Link down
3	Link up
4	Authentication
5	EGP neighbor loss
6	Enterprise-specific ¹

1. This is not a generic trap, but a flag for an enterprise-specific trap.

The following sections provide some examples.

Cold Start

In some cases, it may be important to retain the Cold Start message; for example, where you need to know a machine has been power-cycled if it should not have been. To achieve this, you need to flag the Cold Start message has been handled. In the example here, the **Cold** field is used to show the Cold Start has been handled. The trigger looks for all Cold Start messages having a non-clear **Severity** field and a **Cold** field set to **0**.

Trigger

Trigger Name	FindColdStarts
Sample Rate	60
Ascent Action	ClearColdStart
Descent Action	none
Condition	select * from alerts.status where Summary like 'Cold Start' and Severity > 0 and Cold = 0;
Type	Level
Threshold	none
Execution Mode	For each
Row Count	>0

Action

Table B-17 Cold Start Action

Action	ClearColdStart
Data Effect	update alerts.status set Severity = 0 where LastOccurrence < @LastOccurrence and Serial <> @Serial and Node = '@Node';
External Effect	none

Effect

With this trigger and action, when a Cold Start is received, all related alerts are cleared but the cold start itself is not; it just has the **Cold** field set to **1**, which stops the alert from setting off the trigger again.



Note

In the above example, the **Cold** field was added to the **alerts.status** table as a custom field; it is not included in the default table.

Link Up

When a Link Up occurs, any Link Down messages from that node are implicitly no longer valid. It is appropriate to clear all the Link Down messages and the Link Up message.

The **AlertKey** field is critical for Link Up messages, because it should contain the physical port number of the link. When clearing Link Down messages, the only ones to clear are those from the node and physical port from which the Link Up originated.

Trigger

Trigger Name	FindLinkUpOne
Sample Rate	60
Ascent Action	ClearLinkUpOne
Descent Action	none
Condition	select * from alerts.status where Summary like 'Link Up' and Severity > 0;
Type	Level
Threshold	none
Execution Mode	For each
Row Count	>0

Action

Table B-18 Link Up Action

Action	ClearLinkUpOne
Data Effect	<pre> update alerts.status via @Identifier set Severity = 0; update alerts.status set Severity = 0 where LastOccurrence <= @LastOccurrence and AlertKey = '@AlertKey' and Summary like 'Link Down' and Node = '@Node'; </pre>
External Effect	none

Effect

With this Link Up trigger and action, related Link Down alerts are cleared. However, this example may not be powerful enough for all networks (for example, a WAN). The next example describes an alternative method.

Link Up and ISDN

Not all Link Up alerts should clear their corresponding Link Down alerts. For example, if you are monitoring a European ISDN line, you may be more interested in the Link Up messages, as opposed to Link Down messages with traditional, permanent links.

For example, Cisco routers report the traditional **Index** value along with the type of circuit (for example, Ethernet), token ring, and basic rate interface (BRI). This information can be included in the **Summary** field. The BRI information can be used to discriminate between ISDN and non-ISDN link up/down messages. This automation requires two triggers and actions.

Trigger One

Trigger Name	FindNonISDNLinkUp
Sample Rate	60
Ascent Action	ClearNonISDNLinkUp
Descent Action	none

Condition	select * from alerts.status where Severity > 0 and Summary like 'Link up' and Summary not like 'BRI';
Type	Level
Threshold	none
Execution Mode	For each
Row Count	>0

Action One

Table B-19 Clear Non-ISDN Link Up Action

Action	ClearLinkUpOne
Data Effect	update alerts.status via @Identifier set Severity = 0; update alerts.status set Severity = 0 where LastOccurrence <= @LastOccurrence and AlertKey = '@AlertKey' and Summary like 'Link Down' and Node = '@Node';
External Effect	none

Trigger Two

Trigger Name	FindISDNLinkDown
Sample Rate	60
Ascent Action	ClearISDNLinkDown
Descent Action	none
Condition	<pre>select * from alerts.status where Severity > 0 and Summary like 'Link Down' and Summary like 'BRI';</pre>
Type	Level
Threshold	none
Execution Mode	For each
Row Count	>0

Action Two

Table B-20 Clear Non-ISDN Link Up Action

Action	ClearLinkUpOne
Data Effect	<pre>update alerts.status via @Identifier set Severity = 0; update alerts.status set Severity = 0 where LastOccurrence <= @LastOccurrence and Summary like 'Link Up' and Node = '@Node' and AlertKey = '@AlertKey';</pre>
External Effect	none

Effect

With this pair of triggers and actions, ISDN Link Down messages clear Link Up messages and non-ISDN Link Up messages clear Link Down messages.

Cisco Info Center Customized Automations

This section describes the customized automations provided with Cisco Info Center. These automations are customized to work with specific Cisco Info Center components and data sources.



Note

Which customized automations are installed and active depends on how the Admin Desktop was installed on the host where you are running the Administration interface. During Admin Desktop, you are prompted to install a series of **.elc** file—These **.elc** files determine which menus, tools, and automations are installed.

AA_GenericClear_Populate_Problems_Table

The **AA_GenericClear_Populate_Problems_Table** automation correlates two problem/resolution events. For example, it correlates a Device Up event with a Device Down event. It does this by checking the **Type** field to determine whether the event is a problem event (**Type = 1**) or a **Resolution** event (**Type = 2**). It then checks the **LastOccurrence**, **AlertGroup**, **Manager**, **Node**, and **AlertKey** fields to make sure that the problem and resolution pertain to the same device.

Trigger	<pre>select * from alerts.status where Type = 1 and Severity > 0 and AlertGroup in (select AlertGroup from alerts.status where Type = 2) and AlertKey in (select AlertKey from alerts.status where Type = 2) and ((NEName <> “ and NEName in (select NEName from alerts.status where Type = 2)) or (NEAddress <> “ and NEAddress in (select NEAddress from alerts.status where Type = 2)) or (NEName = “ and NEAddress = “ and Node in (select Node from alerts.status where Type = 2));</pre>
Action	None
Status	Active
Sample Rate	5 seconds

AB_GenericClear_Correlate_Problems_Table

The **AB_GenericClear_Correlate_Problems_Table** automation selects all Resolution events from the **alerts.status** table.

Trigger	select * from alerts.status where Type = 2;
Action	Selects all Resolution events.
Status	Active
Sample Rate	5 seconds

AC_GenericClear_Correlate_Status_Table

This automation selects all Resolved events from the **alerts.generic_clear_problem_events** table.

Trigger	select * from alerts.generic_clear_problem_events where Resolved = 1;
Action	Selects all Resolved events from the alerts.generic_clear_problem_events table.
Status	Active
Sample Rate	5 seconds

ConnSecWatchMsgs

Deletes ProbeWatch messages from the Event list display.

Trigger	select * from alerts.status where ((Manager = 'ConnectionWatch' or (Manager = 'SecurityWatch')) and (NEName = ""));
Action	update alerts.status set NEName = '@Node', NEType = 1000, ObjectType = 'InfoCenter.system' where Serial = @Serial;
Status	Active
Sample Rate	33 seconds

Delete 12hrs Clear

Deletes all events that have a Severity of 0 (green, cleared) and which have not been updated for 12 hours.

Trigger	delete from alerts.status where (Severity = 0) and (StateChange < getdate - 43200);
Action	None
Status	Active
Sample Rate	900 seconds

Delete 24hrs Old

This automation unconditionally deletes all events that have not been updated for a period of twenty four hours.

Trigger	delete from alerts.status where (StateChange < getdate - 86400);
Action	None
Status	Inactive
Sample Rate	920 seconds

Delete Clears (30)

This automation deletes all events in the cleared state (Status = 0) and have not been updated for the last 30 minutes.

Trigger	delete from alerts.status where (Severity = 0) and (StateChange < getdate - 1800);
Action	None
Status	Inactive
Sample Rate	241 seconds

Detect Card Removed

This automation detects events describing the removal of cards and calls an Action to clear any related card events that occurred before the removed card event, where the related event is not in a cleared state, and where NEName and Slot are the same as the removed card. It does not delete the removed card event returned in the trigger.

Trigger	select * from alerts.status where ObjectType = 'card' and Summary like '[rR]emoved';
Action	update alerts.status set Severity = 0, ObjectStatus = 4, class = 1070 where ObjectStatus <> 4 and Serial <> @Serial and LastOccurrence <= @LastOccurrence and NEName = '@NEName' and ObjectType = 'card' and Slot = '@Slot';
Status	Inactive
Sample Rate	7 seconds

Detect Clear PE Interface

This automation detects provider edge routers (PEs) that have been cleared.



Note

This automation exists in Cisco Info Center 3.6 only if you installed Cisco Info Center 3.5 and when you upgraded to Cisco Info Center 3.6, retained the Policy Manager component of Cisco Info Center to monitor the Cisco MPLS VPN Solution product.

Trigger	SELECT NEName, PSAP FROM alerts.status where Severity = 0 and (MPLSType = 1 or MPLSType = 2);
Action	update alerts.status set Severity = 0 where Severity > 0 and MPLSType = 3 and NEName = '@NEName' and PSAP = '@PSAP';
Status	Inactive
Sample Rate	67 seconds

Detect Clear PE Node

Detects any PE nodes that have been cleared.



Note

This automation exists in Cisco Info Center 3.6 only if you installed Cisco Info Center 3.5 and when you upgraded to Cisco Info Center 3.6, retained the Policy Manager component of Cisco Info Center to monitor the Cisco MPLS VPN Solution product.

Trigger	SELECT NEName from alerts.status where Severity = 0 and (MPLSType = 1 or MPLSType = 2) and PSAP = “;
Action	update alerts.status set Severity = 0 where Severity > 0 and MPLSType = 3 and NEName = ‘@NEName’ and PSAP = ‘all interfaces’;
Status	Inactive
Sample Rate	69 seconds

Detect ConMon Up

This automation detects events generated internally, notifying the connection of a remote Mediator to the Info Server and clears the disconnection events from that same remote Mediator that occurred before the reconnect. This automation deletes the connection message.

Trigger	delete from alerts.status where (Manager = ‘ConnectionMonitor’) and (Summary like ‘A probe process running .* has connected’);
Action	update alerts.status set Severity = 0 where Manager = ‘@Manager’ and (Summary like ‘A probe process running .* has disconnected’);
Status	Inactive
Sample Rate	15 seconds

Detect Endpoint Deleted

This automation detects events describing the deletion of end points (**endpoint.ATM**, **endpoint.FR**, **endpoint.CE**) and calls an action to clear any related end point events or connection events that occurred before the deleted end point event, where the related event is not in a cleared state and is from the same NEName, and where PSAP, DSAP, and Slot are the same as the deleted end point. This does not delete the deleted end point event returned in the trigger

Trigger	<code>select * from alerts.status where ObjectType like 'endpoint' and Summary like 'Deleted';</code>
Action	<code>update alerts.status set Severity = 0, ObjectStatus = 4, class = 1070 where Serial <> @Serial and ObjectStatus <> 4 and LastOccurrence <= @LastOccurrence and NEName = '@NEName' and PSAP = '@PSAP' and DSAP = '@DSAP' and Slot = '@Slot' and ((ObjectType = '@ObjectType') or (ObjectType = 'connection'));</code>
Status	Inactive
Sample Rate	6 seconds

Detect Line Deactivated

This automation detects events describing the deactivation of line events (**line.access** and **line.trunk**) and calls an Action to clear any related line events that occurred before the deleted line event, where the related event is not in a cleared state, is from the same NEName, and where PSAP and Slot are the same as the deactivated line. This does not delete the deactivated line event returned in the trigger.

Trigger	<code>select * from alerts.status where ((ObjectType = 'line.access') or (ObjectType = 'line.trunk')) and Summary like 'Deactivated';</code>
Action	<code>update alerts.status set Severity = 0, ObjectStatus = 4, class = 1070 where Serial <> @Serial and ObjectStatus <> 4 and LastOccurrence <= @LastOccurrence and NEName = '@NEName' and PSAP = '@PSAP' and Slot = '@Slot' and (ObjectType = '@ObjectType');</code>
Status	Inactive
Sample Rate	6 seconds

Detect Node Removed

This automation detects events describing the removal of nodes and calls an Action to clear any related node events that occurred before the removed node event, where the related event is not in a cleared state, and where the NEName and Slot are the same as the removed node. It does not delete the removed node event returned in the trigger.

Trigger	select * from alerts.status where ObjectType = 'NE' and Summary like 'removed from network';
Action	update alerts.status set Severity = 0, ObjectStatus = 4, class = 1000 where Serial <> @Serial and ObjectStatus <> 4 and Class <> 1000 and LastOccurrence <= @LastOccurrence and NEName = '@NEName' and ObjectType = 'NE';
Status	Inactive
Sample Rate	7 seconds

Detect Port Deactivated

This automation detects events describing the deactivation of ports and calls an Action to clear any related port events that occurred before the deactivated port event, where the related event is not in a cleared state, and where the NEName, Slot, and PSAP are the same as the deactivated port. It does not delete the deactivated port event returned in the trigger.

Trigger	select * from alerts.status where ObjectType like 'port' and (Summary like 'Deactivated') or (Summary like 'deleted')
Action	update alerts.status set Severity = 0, ObjectStatus = 4, class = 1070 where ObjectStatus <> 4 and Serial <> @Serial and LastOccurrence <= @LastOccurrence and NEName = '@NEName' and PSAP = '@PSAP' and Slot = '@Slot' and ObjectType = '@ObjectType';
Status	Inactive
Sample Rate	8 seconds

FlashingAlarm

This automation sets the Event List to flash for uncleared events that have not been acknowledged.

Trigger	update alerts.status set Flash = 1 where (Acknowledged = 0) and ((Severity > 1) and (Flash = 0))
Action	None
Status	Inactive
Sample Rate	4 seconds

DetectUnknownServices

The DetectUnknownServices automation deletes VPN status events when there are no affected Customer Edge (CE) routers in a VPN.



Note

This automation exists in Cisco Info Center 3.6 only if you installed Cisco Info Center 3.5 and when you upgraded to Cisco Info Center 3.6, retained the Policy Manager component of Cisco Info Center to monitor the Cisco MPLS VPN Solution product.

Trigger	UPDATE alerts.status SET Severity = 0 where MPLSType = 4 AND Poll = 0;
Action	Sets the Severity for an MPLS VPN event to 0, thereby suppressing its display.
Status	Active
Sample Rate	61 seconds

EscalateOff

The EscalateOff automation sets the Flash field for an event to 0 (not flashing) and its Grade to 0 (not escalated) when an event that has previously had the Flash field set to 1 or greater is either acknowledged or cleared (Severity = 0).

Trigger	update alerts.status set Flash = 0. Grade = 0 where ((Flash = 1 or Grade > 0) and Acknowledged = 1) or (Severity = 0);
Action	Stops an event from flashing when it has been acknowledged or cleared.
Status	Inactive
Sample Rate	6 seconds

FlashingOff

This automation sets the Event List to stop flashing for uncleared events that have not been acknowledged and were set to flash as a result of the above automation.

Trigger	update alerts.status set Flash = 0 where (Acknowledged = 1) and ((Severity > 1) and (Flash = 1))
Action	None
Status	Inactive
Sample Rate	3 seconds

Generic Clear

The generic clear automation executes in real time. As this automation substantially reduces the overall numbers of automations required, real time processing overhead can be incurred.

Trigger	select * from alerts.status where Type = 2
Action	update alerts.status set Severity = 0, BitMapField = '@BitMapField', ObjectStatus = 4 where Type = 1 and NEName = '@NEName' and AlertGroup = '@AlertGroup' and AlertKey = '@AlertKey' and LastOccurrence <= @LastOccurrence; delete from alerts.status via '@Identifier';
Status	Inactive
Sample Rate	3 seconds

Policy Manager Log Cleanup

This automation removes extra log files that may be created by the Policy Manager component of Cisco Info Center. This automation only needs to be active if you are using the Policy Manager component of Cisco Info Center.



Note

This automation exists in Cisco Info Center 3.6 only if you installed Cisco Info Center 3.5 and when you upgraded to Cisco Info Center 3.6, retained the Policy Manager component of Cisco Info Center to monitor the Cisco MPLS VPN Solution product.

Trigger	SELECT * FROM auto.triggers where Name like 'Policy';
Action	Calls /opt/Omnibus/polmgr/bin/nci_logcleaner.
Status	Active
Sample Rate	86400 seconds

Policy Manager Lookup Table Creation

Creates a lookup table for the Policy Manager application.



Note

This automation exists in Cisco Info Center 3.6 only if you installed Cisco Info Center 3.5 and when you upgraded to Cisco Info Center 3.6, retained the Policy Manager component of Cisco Info Center to monitor the Cisco MPLS VPN Solution product.

Trigger	SELECT *FROM auto.triggers where Name like 'Policy Manager Lookup Table Creation';
Action	Selects events that have Policy Manger contained in them
Status	Active
Sample Rate	1 second

Resolve NEName or NEAddress

Detects the NEName if the NEAddress field is empty or detects the NEAddress if the NEName field is empty.

Trigger	select * from alerts.status where (NEAddress = "" or NEName = "") and not (NEAddress = "" and NEName = "") and Manager not in ('ConnectionWatch', 'SecurityWatch');
Action	Calls /opt/Omnibus/utils/tools/MWFM/address_lookup.pl
Status	Inactive
Sample Rate	5 seconds

Select Type 3

Selects all events with a Type value of 3 and a Grade Value not equal to 5.

Select Type 4

This automation is used with the Select Type 3 automation.

Transient Condition Clear

Trigger	update alerts.status set Severity = 0, Type = 2 where ObjectStatus = 11 and Type = 1 and Poll + StateChange <getdate;
Action	Sets the Type field to 2 (Resolved)
Status	Active
Sample Rate	60 seconds

VO Re-Open

This automation checks the status of Virtual Operator alarms which have been cleared but not yet deleted.


Note

This automation is installed only if the Virtual Operator component is installed.

Trigger	UPDATE alerts.status SET VOStatus = 0 where VOStatus =3 and Severity > 0;
Action	Sets the Type field to 2, which marks the event as Resolved.
Status	Inactive. Activate this automation if Virtual Operator is installed.
Sample Rate	5 seconds

VO Release

The VO Release automation escalates and dequeues any alarms that have been in the Virtual Operator queue longer than the longest release time out defined for Virtual Operator. This releases events that may have accumulated in the Virtual Operator queue if the Virtual Operator process goes down or the application becomes overloaded and it temporarily cannot process events.



Note

This automation is installed only if the Virtual Operator component is installed.

Trigger	UPDATE alerts.status SET Suppressed - 0, VOStatus = 5 where VOStatus in (1,2) and VOQTime < (getdate - 7201)
Action	Selects and dequeues events that have been in the Virtual Operator processing queue for 1 minute longer than the default timeout (7200 seconds). If the resolution configuration for your Virtual Operator is different, change the VOQTime value to match the timeout value configured for your Virtual Operator configuration,
Status	Inactive. Activate this automation if Virtual Operator is installed.
Sample Rate	10 seconds

Case Example One - Batching and Sending Alerts

The example shows an automation required to collect alerts and forward them to a third-party database as SNMP traps in batches of 100. Additionally they need to be ordered by serial number. It performs the following processing:

1. Alert data is acquired by multiple Strataview Info Mediators. The Info Mediators poll the Informix database underlying Strataview +.
2. The Cisco Info Mediator rules file indicates the fields from the Informix database that should be populated in the Cisco Info Server database fields and indicates how de-duplication within the Cisco Info Server is achieved. The Cisco Info Mediator also creates and sets fields used in the automation.
3. In this case, Cisco Info Server de-duplication is turned off as this is performed within rules file processing.
4. The automation sets the **SendNew** field to 1 for the first 100 alerts, ordered by serial number. It only does this for those alerts that must be sent (those with the **FlagNew** field set to 1). This has the effect of sending alerts in batches of 100 to the SNMP gateway.
5. The SNMP gateway is configured for the required mapping and writer attributes. It performs an **AFTER IDUC DO** command to set the **SendNew** field to 2 and the **FlagNew** field to 0. This prevents alerts from being sent twice to the gateway.

Automation

Trigger Name	Ready to send
Sample Rate	10
Ascent Action	Clear to send
Descent Action	none
Condition	select * from alerts.status where FlagNew = 1 order by Serial asc;
Type	Level
Execution Mode	Once only
Ascent Action SQL	update alerts.status set SendNew = 1 where Serial >= @Serial and Serial <= (@Serial + 99);
Decent Action SQL	none
External Effect	none
Row Count	>0

Gateway Configuration File

The following is a partial gateway configuration file showing the relevant extracts from the file:

```
#
CREATE MAPPING SNMP_MAP
(
0 = '@TYPE',
1 = '@ServerName',
2 = '@Serial',
3 = '@LastOccurrence',
4 = '@Manager',
5 = '@rec_tag',
6 = '@NodeAlias',
7 = '@Node',
8 = '@ntime',
9 = '@edate',
10 = '@Serial',
11 = '@time_zone',
12 = '@svclass',
13 = '@Summary',
14 = '@act_flag',
```

```

15 = '@reserved',
16 = '@Severity'
);

#
# Start up the reader - connect to the ObjectServer NCOMS
#

CREATE FILTER FLAGGED AS 'SendNew=1'

START READER NCOMS READER CONNECT TO NCOMS USING FILTER FLAGGED ORDER BY
Serial asc AFTER IDUC DO 'update alerts.status set SendNew=2, FlagNew=0';
#
# Start up the writer
#
START WRITER SNMP_WRITER
(
TYPE = SNMP,
REVISION = 1,
GATEWAY = 'gate1',
MAP = SNMP_MAP,
FORWARD_UPDATES = TRUE
);
#

```

Case Example Two - Acknowledging Alerts with Sound

The following automation monitors remote host links and plays a sound file on a Desktop screen if a link to a host is lost. The sound file is played every 60 seconds until the alert is acknowledged. Extracts of the rules file, look-up table, the automation, and an external action script are included.

Automation Trigger

Trigger Name	Fire soundfile
Sample Rate	60
Ascent Action	Soundfile
Descent Action	none

Condition	select * from alerts.status where Grade = 0 and Acknowledged = 0 and SoundFile <> " ;
Type	Level
Threshold	none
Execution Mode	For each
Row Count	>0

Automation Action

Table B-21 Sound File Action

Action	Soundfile
Data Effect	update alerts.status via @Identifier set Grade = 1;
External Effect	\$OMNIHOME/utils/audioplay -f @Soundfile >/dev/null 2>&1

Rules File

The complete rules file has not been replicated here. The following are the relevant extracts. The **@Customer** field is already populated with a customer short code; for example, **NY for New York**. The table sounds are stored in:

tablesounds='/nfs/home/sounds/sounds.lookup'

```
<<Main Rules Body>>

if (regmatch@Summary, ".+..+")
{
$soundf=@Customer+" "+ extract @Summary, "([^\]*[^\]*[^\]*).*"
@Location="1"
}
else
{
$soundf=@Customer+" "+@Summary
@Location="2"
}

@SoundFile=lookup($soundf, sounds)

if (match@SoundFile, "")
{
@SoundFile=$soundf
}
}
```

Lookup Table

The content of the **sounds.lookup** table used by the rules file is:

```
NY Link Down      NY_linkdown.au
NY Link Up       NY_linkup.au
NY Node Down     NY_nodedown.au
NY Node Up       NY_nodeup.au
NY Threshold on  NY_bandthresh.au
NY Authentication NY_authfail.au
NY Warm Start    NY_nodewarmstart.au
NY Cold Start    NY_coldstart.au
NY IF down on    NY_IntfaceDown.au
NY IF up on      NY_IntFaceUp.au
NY Rearm Trap    NY_CPU_80.au
```

Script

The following script takes the **@SoundFile** value passed to it by the automation and fires the sound file on the host specified within the script. For this example to work, all hosts must have the **/nfshome** NFS partition mounted.

```
#!/bin/csh -f
#
# audioplay -s hostname -f soundfile
#
# Amend the next line to change the target hosts for sound files
set hosts=(host1 host2 host3 host4)

if (-f/NOAUDIO) then
exit 0
endif
#
if ("x$1"=="x") then
echo "Usage: audioplay -f soundfile"
exit 0
endif

if ("x$1"=="x-f") then
set filename=/nfshome/sounds$2
#shift
#shift
endif

if (! -f${filename}) then
echo "Could not find file ${filename}"
exit 3
endif

foreach i($hosts)
set X='sh -c "/usr/sbin/ping $i 1 2>&1 | grep alive"'
if ("x$x" != "") then
(rsh ${i} /usr/demo/SOUND/play ${filename} >/dev/null&)&
echo `date @+%d/%m/%Y %H:%M:%S`' ${filename} root@${i} >>
/usr/Omnibus/log/audioplay.log &
sleep 1
endif
end
echo ` ` >> /usr/Omnibus/log/audioplay.log
```