ScienceLogic

Run Book Automation

Skylar One (SL1) version 12.5.23

Table of Contents

Introduction **6**

What is Run Book Automation? 7

Custom Settings 7

Automation Policies 8

Action Policies 10

Automation Policies **11**

What is an Automation Policy? 12

Before You Begin 12

Viewing the List of Automation Policies 13

Creating an Automation Policy 14

User-initiated Automations 22

PowerPacks that Contain User-initiated Automations 25

Locating a User-initiated Automation 26

Running a User-initiated Automation 27

Creating a User-initiated Automation 28

Scheduling an Automation Policy 28

Creating an Automation Schedule 29

Creating an Automation Policy Limited by Schedule 31

Creating an Automation Policy to Run on a Schedule 33

Manually Executing a Scheduled Automation Policy 35

"Clear" Policies 35

Aligning an Automation Policy with the System Organization 36

Ordering Actions in an Automation Policy 36

Automation Policies and Event Masks and Event Correlation 37

Events Not Displayed on the Events Page that Might Affect Automation Policies 38

Example 38

Editing an Automation Policy 40

Bulk Actions for Automation Policies 41

Updating Run Book Engine Settings 41

Action Policies **43**

What is an Action Policy? 44

Viewing the List of Action Policies ... 44
Creating an Action Policy ... 45
Creating a Custom Action Type ... 47
Examples: ... 49
Creating an Action Policy that Sends an Email Notification ... 50
Creating an Action Policy that Sends an SNMP Trap ... 53
Using the Default ScienceLogic MIBs to Build an SNMP Trap ... 54
Default Traps from Skylar One ... 54
Varbinds ... 56
Example Trap ... 58
Creating the Action Policy ... 59
Creating an Action Policy that Creates a New Ticket ... 60
Creating an Action Policy that Sends an SNMP Set ... 61
Creating an Action Policy that Executes an SQL Query ... 63
Creating an Action Policy that Updates an Existing Ticket ... 65
Creating an Action Policy that Sends an AWS SNS Message ... 66
Setting up a Topic ARN in AWS ... 66
Creating the Action Policy for AWS SNS ... 67
Using the Action Policy in an Automation Policy for AWS SNS ... 68
Creating an Action Policy that Runs a PowerFlow Application ... 70
Creating an Action Policy that Sends an HTTP Request ... 72
Creating an Action Policy that Uses a Custom Action Type ... 73
Using the Results of a Previous Action ... 74
Using the em7_result_list Variable ... 75
Snippet Actions ... **77**
Creating an Action Policy that Executes a Snippet ... 78
Writing Snippet Code ... 80
Snippet Functions ... 80
Snippet Variables ... 83
Credential Dictionary Structure ... 86
Using the Results of Previous Actions ... 88
EM7_LAST_RESULT_LIST Variable ... 88

EM7_LAST_RESULT Variable ... 90
Examples ... **93**
Action Policy that Sends an Email Message ... 94
Action Policy ... 94
Automation Policy ... 95
Sent Email ... 96
Action Policy that Sends an SNMP Trap to an External Server ... 96
Automation Policy ... 97
Action Policy ... 98
Sent Trap ... 99
Action Policy that Creates a Ticket ... 99
Automation Policy ... 100
Action Policy ... 101
Ticket Template ... 102
Resulting Ticket ... 103
Action Policy that Writes an SNMP Value to an External Server ... 104
Action Policy that Sends an SQL Query to an External Server ... 104
Automation Policy ... 105
Action Policy ... 106
Action Policy that Executes a Snippet and Triggers a New Alert ... 107
Action Policy that Executes a Snippet and Sends the Results to a Second Action Policy ... 110
Automation Policy ... 111
Snippet Action Policy ... 112
Ticket Action Policy ... 113
Ticket Template ... 114
Resulting Ticket ... 115
Run Book Variables ... **116**
Run Book Variables ... 117
Example: Adding a Weather Report to a Ticket ... **1**
Weather Report: Snippet Design ... 2
Weather Report: Snippet Code ... 2
Creating the Credential ... 4

Creating the Automation Action 5
Creating the Update Ticket Action 6
Creating the Automation Policy 7
Configuring a Test Event 9
Testing the Automation Policy 10
Configuring a Device Asset Record 10
Triggering a Test Event 11
Creating a Ticket for the Test Event 13
Possible Next Steps 15

Chapter 1

Introduction

Overview

This chapter describes the automation policies and the action policies that you can use to create an run book automation in Skylar One.

This chapter covers the following topics:

What is Run Book Automation? 7

Custom Settings 7

Automation Policies 8

Action Policies 10

What is Run Book Automation?

Run Book Automation allows you to specify actions you want Skylar One to execute automatically when specific event conditions are met. Automation in Skylar One is divided into two parts:

- An ***automation policy*** defines the event conditions that can trigger an automatic action.
- An ***action policy*** defines an action that can be triggered by an automation policy. An action policy can perform one of the following tasks:
    - Send an email message to a pre-defined list of users and/or external contacts.
    - Send an SNMP trap from Skylar One to an external device.
    - Create a new ticket (using ticket templates defined in the **Ticket Templates** page [Registry > Ticketing > Templates]).
    - Update an existing ticket. An action policy can change the status and/or severity of an existing ticket and/or add a note to an existing ticket. For this action policy to trigger successfully, a ticket must be associated with the event that triggered the action.
    - Write an SNMP value to an existing SNMP object on an external device.
    - Query a database.
    - Run a custom python script, called a snippet.
    - Send an SNS Message to a Topic ARN (Amazon Resource Name). All subscribers to the Topic ARN will receive the message.
    - Run a PowerFlow application.
    - Send an HTTP request.

If an automation policy has a Policy Type of *Scheduled*, then an automation schedule from the **Automation Schedule Manager** page (Registry > Run Book > Schedules) is required instead of an event condition to trigger the automation. You can use an automation schedule to automatically trigger run book automations. The actions are executed according to the schedule, regardless of event status, or you can manually execute a scheduled automation policy at any time. You can also use an automation schedule as a criteria in event-triggered run book automations. For more information, see *Scheduling an Automation Policy*.

In addition, you can enable ***user-initiated automations*** that let a user manually trigger an automation policy and its aligned action policies in the Skylar One user interface. The list of available user-initiated automations for an event are based on the organization, devices, and events that are aligned with the automation policy. For more information, see *User-initiated Automations*.

Custom Settings

The process that executes run book tasks is parallelized. The default settings for parallelization are appropriate for most Skylar One systems. However, the run book feature does include internal settings

that can be changed to support extremely large Skylar One systems. For help customizing run book for your environment, contact ScienceLogic Customer Support.

Automation Policies

An ***automation policy*** defines the event conditions that can trigger an automation action. To view a list of automation policies, create an automation policy, or edit an action policy, go to the **Automation Policy Manager** page (Registry > Run Book > Automation).

NOTE: If an automation policy has a Policy Type of *Scheduled*, then an automation schedule from the **Automation Schedule Manager** page (Registry > Run Book > Schedules) is required instead of an event condition to trigger the automation. For more information, see *Scheduling an Automation Policy*.

NOTE: You can enable ***user-initiated automations*** that let a user manually trigger an automation policy and its aligned action policies in the Skylar One user interface. For more information, see *User-initiated Automations*.

When the event criteria in an automation policy are met, one or more actions are executed. These actions are defined in an action policy. To view a list of action policies, create an action policy, or edit an action policy, go to the **Action Policy Manager** page (Registry > Run Book > Actions).

For example, an automation policy might specify the following:

- If the event "illicit process" occurs on device "mailserver01",
- and the event is not cleared within five minutes,
- then execute the action policy "Email NOC".

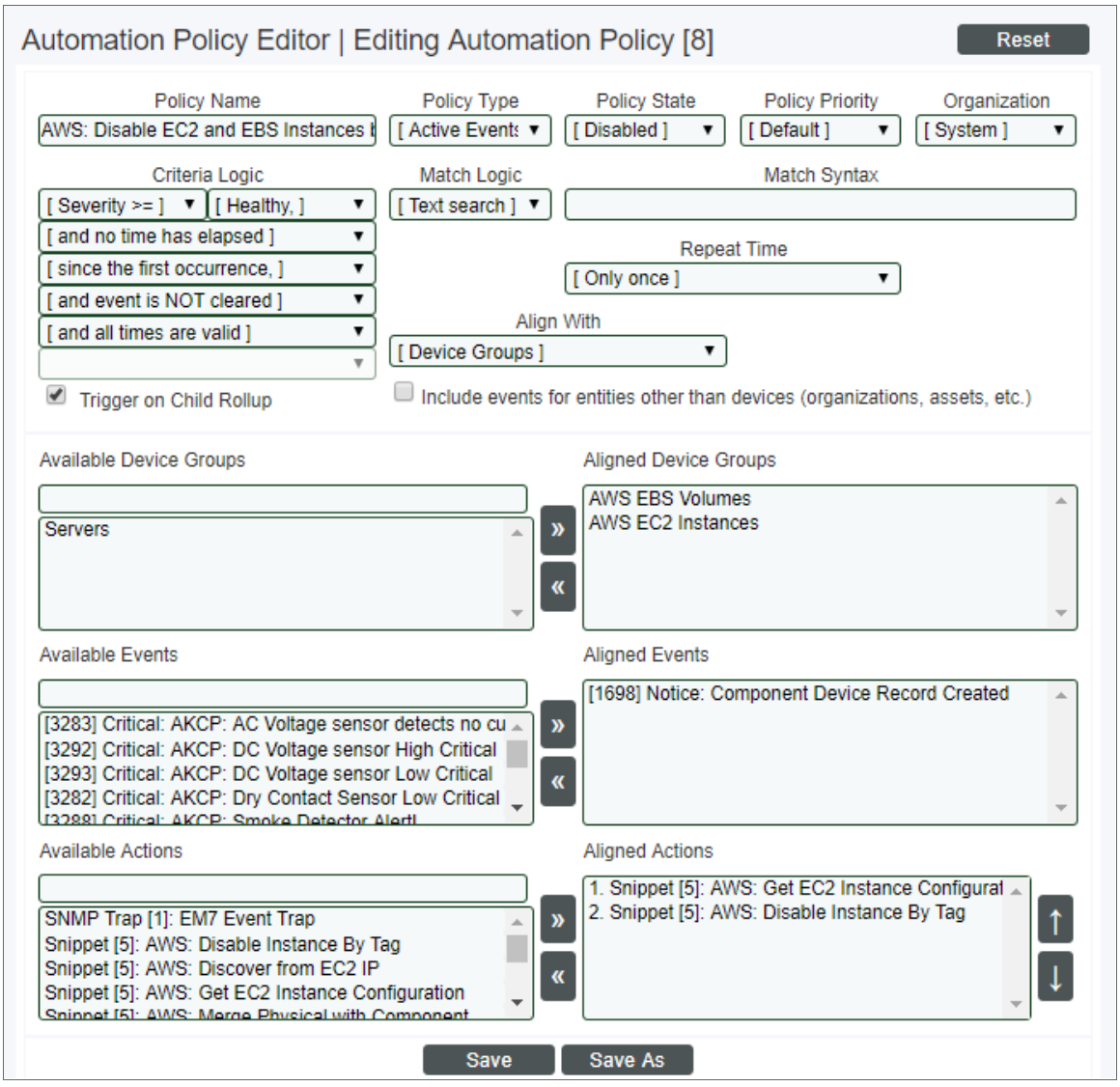
The action policy "Email NOC" could notify all NOC staff about the "illicit process" event.

NOTE: When an automation policy executes actions, the time stamps for the actions will use the time zone defined in the ***System Timezone*** field (System > Settings > Behavior page). However, the timestamp for the Event Action Log window will display the user's local time zone, as defined in the **Account Information** page (Preferences > Account > Information). For more information, see the "My Contact Information" section in the ***Introduction to Skylar One*** manual.

Automation policies can describe the following criteria. One or more of these criteria must be met before an action is executed.

NOTE: Scheduled automation works differently. For more information, see *Scheduling an Automation Policy*.

- At least one of the specified events must have occurred.
- Event(s) must have occurred on at least one of the specified devices.
- Event(s) must have specified severity (critical, major, minor, notice, or healthy).
- Event(s) must have specified status (event is not cleared, event is now acknowledged, ticket is not created for event).
- Specified amount of time must elapse after the event occurs and before the other criteria are evaluated by Skylar One.
- Specified text must appear in the event message.



When the criteria are met, the automation policy triggers the execution of one or more specified action policies. The automation policy specifies one or more actions to execute and the order in which to execute those actions.

To create an automation policy, go to the **Automation Policy Manager** page (Registry > Run Book > Automation). For details, see *Automation Policies*.

Action Policies

An ***action policy*** is an action that can be automatically triggered in Skylar One when certain event criteria are met. To view a list of action policies, create an action policy, or edit an action policy, go to the **Action Policy Manager** page (Registry > Run Book > Actions). For details on creating action policies, see *Action Policies*.

> **NOTE:** The triggers for action policies are defined in an automation policy on the **Automation Policy Manager** page (Registry > Run Book > Automation).

An action policy can perform one of the following tasks:

- Send an email message to a pre-defined list of users and/or external contacts.
- Send an SNMP trap from Skylar One to an external device.
- Create a new ticket (using ticket templates defined in the **Ticket Templates** page [Registry > Ticketing > Templates]).
- Update an existing ticket. An action policy can change the status and/or severity of an existing ticket and/or add a note to an existing ticket. For this action policy to trigger successfully, a ticket must be associated with the event that triggered the action.
- Write an SNMP value to an existing SNMP object on an external device.
- Query a database.
- Run a custom Python script, called a snippet.
- Send a Message to a Topic ARN (Amazon Resouce Name). All subscribers to the Topic ARN will receive the message.
- Run a PowerFlow application.
- Send an HTTP request.

Chapter 2

Automation Policies

Overview

This chapter describes the automation policies found on the **Automation Policy Manager** page (Registry > Run Book > Automation).

This chapter covers the following topics:

What is an Automation Policy? 12

Before You Begin 12

Viewing the List of Automation Policies 13

Creating an Automation Policy 14

User-initiated Automations 22

Scheduling an Automation Policy 28

"Clear" Policies 35

Aligning an Automation Policy with the System Organization 36

Ordering Actions in an Automation Policy 36

Automation Policies and Event Masks and Event Correlation 37

Events Not Displayed on the Events Page that Might Affect Automation Policies 38

Example 38

Editing an Automation Policy 40

Bulk Actions for Automation Policies 41

Updating Run Book Engine Settings *41*

What is an Automation Policy?

An ***automation policy*** defines the combination of event conditions that can trigger an automatic action.

When the criteria in an automation policy is met, one or more actions are executed. Each action is defined in an action policy. Action policies are described in detail in the section on *Action Policies*.

NOTE: When an automation policy executes actions, the time stamps for the actions will use the time zone defined in the System > Settings > Behavior page, in the ***System Timezone*** field. However, "Send an Email Notification" actions will use the time zone associated with each recipient's account, as defined in the **Account Preferences** page for each recipient. For more information on the Account Preferences, see the chapter on *Managing User Accounts* in the manual ***Organizations and Users***.

Automation policies can describe the following criteria. One or more of these criteria must be met before an action is executed.

- At least one of the specified events must have occurred.
- Event(s) must have occurred on at least one of the specified devices.
- Event(s) must have specified severity (critical, major, minor, notice, or healthy).
- Event(s) must have specified status (event is not cleared, event is now acknowledged, ticket is not created for event).
- Specified amount of time must elapse after the event occurs and before the other criteria are evaluated by Skylar One.
- Specified text must appear in the event message.

NOTE: If an automation policy has a Policy Type of *Scheduled*, then an Automation Schedule from the **Automation Schedule Manager** page (Registry > Run Book > Schedules) is required instead of an event condition to trigger the automation. For more information, see *Scheduling an Automation Policy*.

NOTE: You can enable ***user-initiated automations*** that let a user manually trigger an automation policy and its aligned action policies in the Skylar One user interface. For more information, see *User-initiated Automations*.

Before You Begin

Before you define automation policies, you should consider:

- The types of automatic actions that Skylar One can trigger in response to an automation policy. The choices are:
    - Send an email message to a pre-defined list of users and/or external contacts.
    - Send an SNMP trap from Skylar One to an external device.
    - Create a new ticket (using ticket templates defined in the **Ticket Templates** page [Registry > Ticketing > Templates]).
    - Update an existing ticket. An action policy can change the status and/or severity of an existing ticket and/or add a note to an existing ticket. For this action policy to trigger successfully, a ticket must be associated with the event that triggered the action.
    - Write an SNMP value to an existing SNMP object on an external device.
    - Query a database.
    - Run a custom Python script, called a snippet.
    - Send an SMS Message to a Topic ARN (Amazon Resouce Name). All subscribers to the Topic ARN will receive the message.
- The event conditions that are most critical to your business or organization.
- The event conditions that are best suited to an automatic response (instead of a manual response).

Viewing the List of Automation Policies

The **Automation Policy Manager** page (Registry > Run Book > Automation) displays a list of all existing automation policies.

NOTE: Users of type "user" can view only automation policies that are aligned with the same organization(s) to which the user is aligned. Users of type "administrator" can view all automation policies.

TIP: You can sort the list by a column's values by clicking its column heading. You can also filter the items on the page by typing or selecting filter options in one or more of the filters found above the columns. For more information, see the topic on *Filtering Inventory Pages*.

To view the list of automation policies:

1. Navigate to the **Automation Policy Manager** page (Registry > Run Book > Automation).
2. The **Automation Policy Manager** page displays the following about each automation policy:
    - ***Automation Policy Name***. Name of the automation policy.
    - ***ID***. Unique numeric identifier, automatically assigned by Skylar One to each automation policy.

- ***Policy State***. Specifies whether the policy can be executed (enabled) or cannot be executed (disabled).
- ***Policy Priority***. Specifies whether this policy is high-priority or default priority. These options determine how the policy is queued.
- ***Organization***. Organization associated with the automation policy.
- ***Devices***. Number of devices included in the criteria for the automation policy.
- ***Events***. Number of events included in the criteria for the automation policy.
- ***Actions***. Number of action policies that will be executed by the automation policy.
- ***Edited By***. User who created or last edited the automation policy.
- ***Last Edited***. Date and time the automation policy was created or last edited.

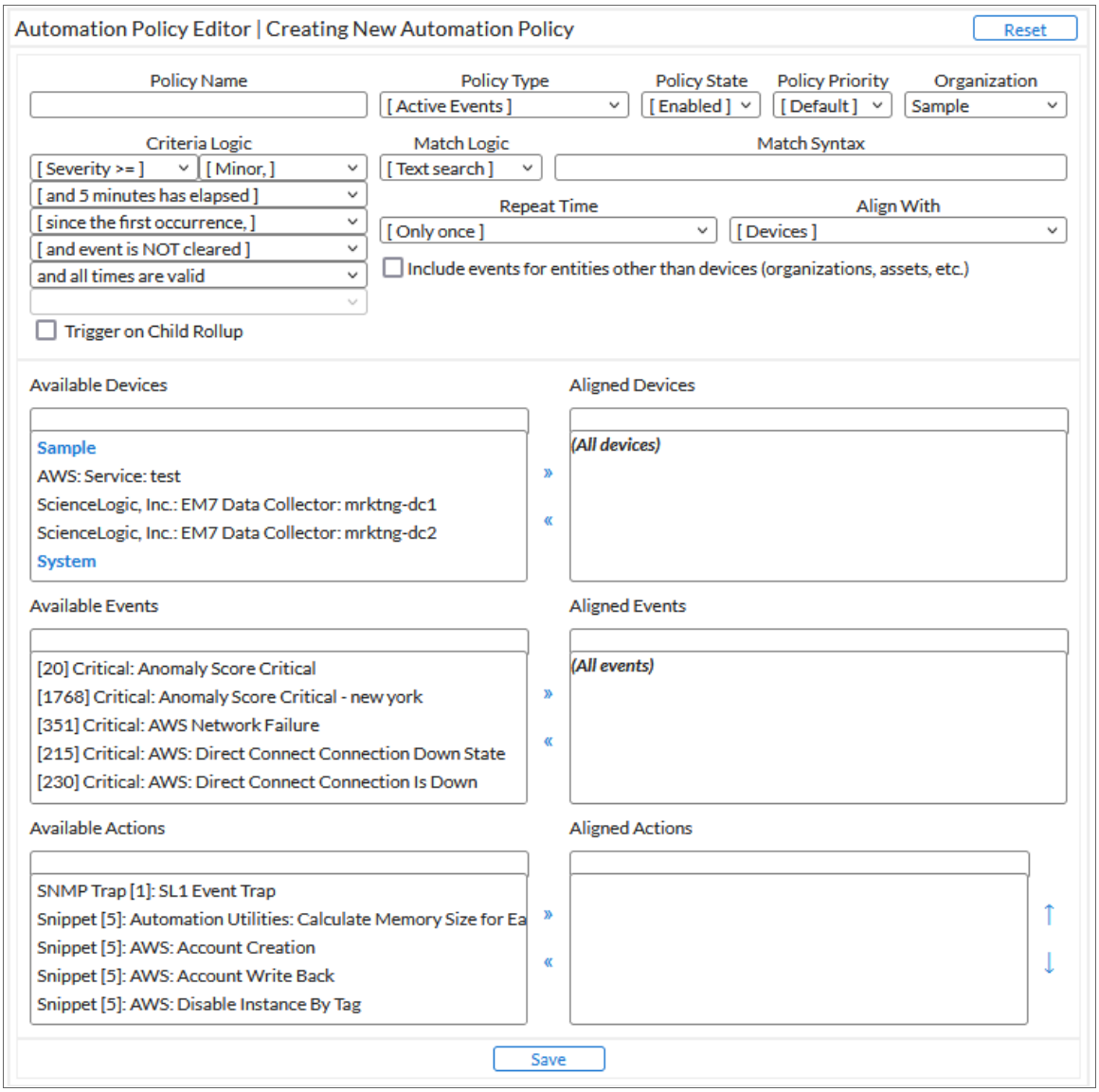
Creating an Automation Policy

You can also create a "user initiated" automation policy that lets a user trigger an automation action (from an action policy) on an as needed, ad hoc basis. A user-initiated automation does not require that the event-based logic in the **Criteria Logic** section matches to *True* in the automation policy, as the user action supersedes that criteria. For more information, see *User-initiated Automations*.

In addition, if an automation policy has a Policy Type of *Scheduled*, then an Automation Schedule from the **Automation Schedule Manager** page (Registry > Run Book > Schedules) is required instead of an event condition to trigger the automation. For more information, see *Scheduling an Automation Policy*.

To create an automation policy:

1. Navigate to the **Automation Policy Manager** page (Registry > Run Book > Automation).
2. Click the **[Create]** button. The **Automation Policy Editor** page appears:



3. In the **Automation Policy Editor** page, supply a value in each of the following fields:
   - ***Policy Name***. Name of the automation policy.
   - ***Policy Type***. Specifies whether the automation policy will search for cleared events or active events, or if this will be a scheduled automation policy. You choices are:
     - *Active Events*. The automation policy will search active events to find events that meet the criteria.
     - *Cleared Events*. The automation policy will search cleared events to find events that meet the criteria.

- *Scheduled*. The automation policy will execute as specified by a selected Automation Schedule. The Automation Policy will not search events to match criteria. For more information, see *Scheduling an Automation Policy*.
- *User Initiated*. The Automation policy lets the user execute an "ad hoc" automation on an event from the **Events** page, an **Event Investigator** page, or a **Service Investigator** page. For more information, see *User-initiated Automations*. If you select *User Initiated*, the following fields are hidden: ***Policy Priority***, ***Criteria Logic***, ***Match Logic***, ***Match Syntax***, ***Repeat Time***, and ***Trigger on Child Rollup***.
- *Active Events/User Initiated*. The automation policy enables all of the features of the "Active Events" and the "User Initiated" Policy Types. As a result, this automation policy can be triggered by active events that meet the criteria in the policy, or a user can manually trigger the automation.

- ***Policy State***. Specifies whether the policy can be executed (enabled) or cannot be executed (disabled).
- ***Policy Priority***. Specifies whether this policy is high priority or default priority. Options are:
  - *Default*. This policy is placed into a default queue. Skylar One includes multiple worker tasks that constantly check this queue and execute policies in this queue. If there are no policies in the default queue, the worker tasks execute policies in the high-priority queue.
  - *High*. This policy is placed into the high-priority queue. Skylar One includes multiple worker tasks that constantly check this queue and execute the policies. For details on configuring the number of worker tasks for high-priority policies, contact ScienceLogic Customer Support. If there are no policies in the high-priority queue, the worker tasks execute policies in the Default queue.
- ***Organization***. Organization associated with the automation policy. If you select the *System* organization, the behavior of the ***Available Devices*** and ***Available Device Groups*** fields is affected.
- ***Criteria Logic***. These fields specify the conditions that must be met before Skylar One executes the action specified in the automation policy. All conditions must be met for at least one of the selected events on one of the selected devices.
  - *Severity Operator*. Used in conjunction with the *Severity* field. Choices are:
    - Severity >=. Severity is greater than or equal to.
    - Severity =. Severity must be equal to.
  - *Severity*. Event must have the specified severity or have a severity greater than or equal to the specified severity. The choices are:
    - Critical
    - Major
    - Minor
    - Notice
    - Healthy

- *Elapsed time*. The length of time that must elapse after the event occurs but before Skylar One evaluates the other criteria in the automation policy. The choices are intervals of time ranging from "no time has elapsed" to "1 month has elapsed", and you must then specify whether the elapsed time is counted "since the first occurrence" or "since the activation time". You might use this field to allow users to manually perform actions before the automation actions are executed.
- *Since*. Specifies the ScienceLogic event that is applied to *Elapsed time*. The choices are:
  - since the first occurrence
  - since the activation time (when an event became active). For more information, see the section on *event states*.
- *Status*. Event must have the specified status. The choices are:
  - and event is NOT cleared
  - and event is NOT acknowledged
  - and ticket is NOT created
  - and event IS acknowledged
  - and ticket IS created
  - and external ticket IS requested
  - and external ticket IS created

NOTE: The *Status* options "and external ticket IS requested" and "and external ticket IS created" require that you select *Create/View External Ticket* for the global setting ***Event Console Ticket Life Ring Button Behavior*** in the **Behavior Settings** page (System > Settings > Behavior). You can use this *Status* to trigger a custom run book action to create a ticket on the external system or perform actions after a ticket is created on the external system. For more information on system settings, see *global settings*.

NOTE: The *Status* option "and ticket IS created" requires that you select *Create/View EM7 Ticket* for the global setting ***Event Console Ticket Life Ring Button Behavior*** in the **Behavior Settings** page (System > Settings > Behavior). You can use this *Status* to trigger a custom run book action that performs actions after a ticket is created on the Skylar One. For more information on system settings, see *global settings*.

NOTE: The ***Elapsed Time*** and ***Status*** fields do not appear if you selected *Cleared Events* in the ***Policy Type*** field.

- *Time/Schedule*. Specifies the timespan during which the Automation Policy can execute the aligned actions. The choices are:
  - *and all times are valid*. The Automation Policy can execute the aligned actions when all the criteria are met. There is no schedule associated with the criteria.
  - *and the following schedule is active*. The Automation Policy can execute the aligned actions during the timespan specified in the selected schedule.
  - *and the following schedule is NOT active*. The Automation Policy can execute the aligned actions during any time except the timespan specified in the selected schedule. You would use this option in cases where you want to restrict event-driven automation from running during a particular time.
- *Schedule*. If in the *Time/Schedule* field you specified "and the following schedule is active" or "and the following schedule is NOT active", select a schedule in this field. Note that ad hoc automation schedules, which specify a single point in time, will not be shown in this field.

- ***Match Logic***. Specifies whether to process the ***Match Syntax*** field as a regular expression or a simple text match. This field is optional. However, if you enter a value in the ***Match Syntax*** field, you must also select a value in this field.
- ***Match Syntax***. An optional string to further filter events. For Skylar One to execute the actions specified in the policy, the event message must match the text or regular expression defined in this field. For example, if you want to be notified only when an event occurs on a specific sub-entity (like an interface or a file system), you can specify a text match or regular expression that will match that sub-entity in this field. Can be any combination of alpha-numeric characters, up to 48-characters in length. Skylar One's expression matching is case-sensitive.
- ***Repeat Time***. The frequency at which Skylar One should execute the automation policy while the conditions are still met. The choices range from "every 30 seconds until satisfied" to "every 2 hours until satisfied", or "only once". The ***Repeat Time*** field does not appear if you selected *Cleared Events* in the ***Policy Type*** field.
- ***Align With***. Specifies whether to align this automation policy with one or more devices, one or more device groups, or one or more organizations.
  - *Devices*. The ***Available Devices*** field will appear below, where you can select devices to associate with the automation policy.
  - *Device Groups*. The ***Available Device Groups*** field will appear below, where you can select device groups to associate with the automation policy.
  - *Policy Organization*. The ***Available Devices in Organization*** field will appear below, where you can select one or more devices to associate with the automation policy. The list of devices comprises all devices in the organization specified in the ***Organization*** field.
  - *Classic IT Services*. This option refers to the IT Services available only in the "classic" EM7 user interface. These services are now called "Classic IT Services". If you select this option, the ***Available Classic IT Services*** field will appear below, where you can select one or more Classic IT Services to associate with the automation policy.

    - *Services*. This option refers to the business services you can create on the **Business Services** page of Skylar One. If you select this option, the ***Available Services*** field will appear below, where you can select one or more Services to associate with the automation policy.

- ***Trigger on Child Rollup***. Affects events that are rolled up, either using event correlation or event masks. If selected, all events in a suppression group can trigger the automation policy. If not selected, only a single event in a suppression group can trigger the automation policy.
- ***Include events for entities other than devices (organizations, assets, etc.)***. If you select this checkbox, the automation policy can match events that are not associated with a device. The automation policy will match events that are not associated with a device only if you do not select specific devices or device groups from the ***Available Devices***, ***Available Device Groups***, ***Available Devices in Organization***, ***Available IT Services***, or ***Available Services*** field.
- ***Available Devices***. If you selected *Devices* in the ***Align With*** field, this field displays a list of all devices in Skylar One. You can select one or more devices in this field. The selected event(s) and event criteria must occur on one of the selected devices before the automation policy will be executed. You can use the field at the top of the ***Available Devices*** field to filter the list of devices. If you enter an alpha-numeric string in the field, the ***Available Devices*** field will include only devices that match the string.
    - ***To select a device***, highlight it and click the right-arrow button.
    - ***If you do not select any devices***, the automation policy automatically evaluates all devices associated with the organization you selected in the ***Organization*** field. If you selected *System* in the ***Organization*** field, the automation policy automatically evaluates all devices in Skylar One. Additionally, if the ***include events for entities other than devices (organizations, assets, etc.)*** checkbox is checked, the automation policy will ***evaluate all events associated with all organizations that are not associated with a device***, regardless of the organization selected in the ***Organization*** field.
    - ***If you select specific devices***, the automation policy will evaluate all selected devices. Not selecting specific devices allows an automation policy to evaluate events that are aligned with an entity other than a device.
- ***Aligned Devices***. This pane displays a list of all devices aligned with the automation policy. To de-select a device, highlight it and click the left-arrow button.
- ***Available Device Groups***. If you selected *Device Groups* in the ***Align With*** field, this field displays a list of all device groups in Skylar One. You can select one or more device groups in this field. The selected event(s) and event criteria must occur on at least one device in one of the selected device groups before the automation policy will be executed. You can use the field at the top of the ***Available Device Groups*** field to filter the list of device groups. If you enter an alpha-numeric string in the field, the ***Available Device Groups*** field will include only device groups that match the string.

- ***To select a device group***, highlight it and click the right-arrow button.
- ***If you do not select any device groups***, the automation policy automatically evaluates all device groups to which you have access. Additionally, if the ***Include events for entities other than devices (organizations, assets, etc.)*** checkbox is checked, the automation policy will ***evaluate all events associated with all organizations that are not associated with a device***, regardless of the organization selected in the ***Organization*** field.
- ***If you select specific device groups***, the automation policy will evaluate all selected device groups. Not selecting specific device groups allows an automation policy to evaluate events that are aligned with an entity other than a device.

- ***Aligned Device Groups***. This pane displays a list of all device groups aligned with this automation policy. To de-select a device group, highlight it and click the left-arrow button. Note that selecting a device group aligns all devices that are part of the device group with the automation policy.
- ***Available Devices in Organization***. If you selected *Policy Organization* in the ***Align With*** field, this field displays only devices from the organization selected in the ***Organization*** field. You can select one or more devices in this field. The selected event(s) and event criteria must occur on one selected device before the automation policy will be executed. You can use the field at the top of the ***Available Devices in Organization*** field to filter the list of devices. If you enter an alpha-numeric string in the field, the ***Available Devices in Organization*** field will include only devices that match the string.
  - ***To select a device***, highlight it and click the right-arrow button.
  - ***If you do not select any devices***, the automation policy automatically evaluates all devices associated with the organization you selected in the ***Organization*** field. Additionally, if the ***Include events for entities other than devices (organizations, assets, etc.)*** checkbox is checked, the automation policy will ***evaluate all events associated with the organization specified in the Organization field that are not associated with a device***.
  - ***If you select specific devices***, the automation policy will evaluate all selected devices. Not selecting specific devices allows an automation policy to evaluate events that are aligned with an entity other than a device.
- ***Aligned Devices***. This pane displays a list of all devices aligned with this automation policy. To de-select a device, highlight it and click the left-arrow button.
- ***Available Classic IT Services***. If you selected *Classic IT Services* in the ***Align With*** field, this field displays a list of all Classic IT Services in Skylar One. You can select one or more Classic IT Services in this field. The selected event(s) and event criteria must occur for one of the selected Classic IT Services before the automation policy will be executed. You can use the field at the top of the ***Available Classic IT Services*** field to filter the list of IT service policies. If you enter an alpha-numeric string in the field, the ***Available Classic IT Services*** field will include only IT service policies that match the string.

- ***To select a Classic IT Service***, highlight it and click the right-arrow button.
- ***If you do not select any Classic IT Services***, the automation policy automatically evaluates all Classic IT Services associated with the organization you selected in the ***Organization*** field. If you selected *System* in the ***Organization*** field, the automation policy automatically evaluates all IT Services in Skylar One.
- ***If you select specific Classic IT Services***, the automation policy will evaluate all selected devices. Not selecting specific Classic IT Services allows an automation policy to evaluate events that are aligned with an entity other than an IT Service.

- ***Aligned Classic IT Services***. This pane displays a list of all Classic IT Services aligned with this automation policy. To de-select a Classic IT Service, highlight it and click the left-arrow button.
- ***Available Services***. If you selected *Services* in the ***Align With*** field, this field displays a list of all Services in Skylar One. You can select one or more Services in this field. The selected event(s) and event criteria must occur for one of the selected Services before the automation policy will be executed. You can use the field at the top of the ***Available Services*** field to filter the list of service policies. If you enter an alpha-numeric string in the field, the ***Available Services*** field will include only service policies that match the string.
- ***Aligned Services***. This pane displays a list of all Services aligned with this automation policy.
- ***Available Events***. Displays a list of all defined events in Skylar One. You can select one or more events in this field. One of the selected events and event criteria must occur on one selected device before the automation policy will be executed. ***To select an event***, highlight it and click the right-arrow button. This pane also displays the ID number for each aligned event policy to ensure you select the relevant policy. You can use the field at the top of the ***Available Events*** field to filter the list of events. If you enter an alpha-numeric string in the field, the ***Available Events*** field will include only events that match the string.
- ***Aligned Events***. This pane displays a list of all events aligned with this automation policy, along with the ID number of the aligned event policy. To de-select an event, highlight it and click the left-arrow button.

NOTE: If a triggering event (that is, an event specified in the ***Aligned Events*** field is not aligned with a device (but is instead aligned with an organization), and you have also selected one or more ***Aligned Actions*** that must be executed on a Data Collector, Skylar One will 1) Not execute the action policy; 2) Create a log entry in the audit log for the organization aligned with the triggering event, noting that the criteria in the automation policy were met, but that the action policy was not executed. This does not apply to Action Policies created on an All-In-One Appliance.

- ***Available Actions***. Displays a list of all action policies in Skylar One. (Action policies are defined in Registry > Run Book > Actions.) You can select one or more action policies in this field. If the selected event(s) and event criteria occur on the selected devices or for the selected IT Services, the selected action policies will be executed. To select an action policy, highlight it and click the right arrow-button. You can use the field at the top of the ***Available Actions*** field to filter the list of action policies. If you enter an alpha-numeric string in the field, the ***Available Actions*** field will include only action policies that match the string.

- ***Aligned Actions***. This pane displays a list of all action policies aligned with this automation policy.
    - ***To de-select an action policy***, highlight it and click the left-arrow button.
    - ***To change the order in which one or more action policies are executed***, highlight the action policy and use the up-arrow or down-arrow to move the policy within the list.

> **NOTE:** If you selected multiple action policies in the automation policy, the action policies will be executed in the order specified in the ***Aligned Actions*** field. To change the order of one or more action policies, highlight the action policy and use the up-arrow or down-arrow to move the policy within the list.

- **[Save]**. Saves a new automation policy or saves changes to an existing automation policy.
- **[Save As]**. If you supply a new value in the ***Policy Name*** field, saves the current automation policy, including any edits, as a new policy with a new name.

4. Click the **[Save]** button to save the new automation policy or save changes to an existing automation policy.

User-initiated Automations

For an automation action to be visible on the **Tools** pane, the following two things must be true:

- ***Organization***. The event must occur for the aligned organization specified in the automation policy, or the automation policy is linked to the System organization.
- ***Aligned Event***. The event must match one of the aligned events specified in the automation policy.

> **NOTE:** All user-initiated run book automations are shown on the **Tools** pane, even if they are disabled.

To create a user-initiated automation:

1. On the **Automation Policy Manager** page (Registry > Run Book > Automation), click **[Create]**. The Automation Policy Editor window appears.
2. On the **Automation Policy Editor** page, complete the following fields:
    - ***Policy Name***. Name of the automation policy.
    - ***Policy Type***. Specifies whether the automation policy will search for cleared events or active events, or if this will be a scheduled automation policy. You choices are:

- *User Initiated*. The Automation policy lets the user execute an "ad hoc" automation on an event from the **Events** page or a service from the **Service Investigator** page.

> **NOTE:** If you select *User Initiated*, the following fields are hidden: ***Policy Priority***, ***Criteria Logic***, ***Match Logic***, ***Match Syntax***, ***Repeat Time***, and ***Trigger on Child Rollup***.

- *Active Events/User Initiated*. The automation policy enables all of the features of the "Active Events" and the "User Initiated" Policy Types. As a result, this automation policy can be triggered by active events that meet the criteria in the policy, or a user can manually trigger the automation.

> **NOTE:** If you select *Active Events/User Initiated*, see *Creating an Automation Policy* for descriptions of the additional fields on the **Automation Policy Editor** page.

- ***Policy State***. Specifies whether the policy can be executed (enabled) or cannot be executed (disabled).
- ***Organization***. Organization associated with the automation policy. If you select the *System* organization, the behavior of the ***Available Devices*** and ***Available Device Groups*** fields is affected.
- ***Align With***. Specifies whether to align this automation policy with one or more devices, one or more device groups, or one or more organizations.
  - *Devices*. The ***Available Devices*** field will appear below, where you can select devices to associate with the automation policy.
  - *Device Groups*. The ***Available Device Groups*** field will appear below, where you can select device groups to associate with the automation policy.

> **NOTE:** To run user-initiated automations with a device group, the device group must use a *static* list of devices, not a dynamic list of devices.

  - *Policy Organization*. The ***Available Devices in Organization*** field will appear below, where you can select one or more devices to associate with the automation policy. The list of devices comprises all devices in the organization specified in the ***Organization*** field.
- ***Available Devices***. If you selected *Devices* in the ***Align With*** field, this field displays a list of all devices in Skylar One. You can select one or more devices in this field. The selected event(s) and event criteria must occur on one of the selected devices before the automation policy will be executed.

    - ***To select a device***, select it and click the right-arrow button.
    - ***If you do not select any devices***, the automation policy automatically evaluates all devices associated with the organization you selected in the ***Organization*** field. If you selected *System* in the ***Organization*** field, the automation policy automatically evaluates all devices in Skylar One.
    - ***If you select specific devices***, the automation policy will evaluate all selected devices.

- ***Aligned Devices***. This pane displays a list of all devices aligned with the automation policy. To de-select a device, select it and click the left-arrow button.
- ***Available Device Groups***. If you selected *Device Groups* in the ***Align With*** field, this field displays a list of all device groups in Skylar One. You can select one or more device groups in this field. The selected event(s) and event criteria must occur on at least one device in one of the selected device groups before the automation policy will be executed.
    - ***To select a device group***, select it and click the right-arrow button.
    - ***If you do not select any device groups***, the automation policy automatically evaluates all device groups to which you have access.
    - ***If you select specific device groups***, the automation policy will evaluate all selected device groups.
- ***Aligned Device Groups***. This pane displays a list of all device groups aligned with this automation policy. To de-select a device group, select it and click the left-arrow button.
- ***Available Devices in Organization***. If you selected *Policy Organization* in the ***Align With*** field, this field displays only devices from the organization selected in the ***Organization*** field. You can select one or more devices in this field. The selected event(s) and event criteria must occur on one selected device before the automation policy will be executed.
    - ***To select a device***, select it and click the right-arrow button.
    - ***If you do not select any devices***, the automation policy automatically evaluates all devices associated with the organization you selected in the ***Organization*** field.
    - ***If you select specific devices***, the automation policy will evaluate all selected devices.
- ***Aligned Devices***. This pane displays a list of all devices aligned with this automation policy. To de-select a device, select it and click the left-arrow button.
- ***Aligned IT Services***. This pane displays a list of all IT Services aligned with this automation policy. To de-select an IT Service, select it and click the left-arrow button.
- ***Available Events***. Displays a list of all defined events in Skylar One. You can select one or more events in this field. One of the selected events and event criteria must occur on one selected device before the automation policy will be executed. ***To select an event***, select it and click the right-arrow button. This pane also displays the ID number for each aligned event policy to ensure you select the relevant policy.
- ***Aligned Events***. This pane displays a list of all events aligned with this automation policy, along with the ID number of the aligned event policy. To de-select an event, select it and click the left-arrow button.

NOTE: If a triggering event (that is, an event specified in the ***Aligned Events*** field is not aligned with a device (but is instead aligned with an organization), and you have also selected one or more ***Aligned Actions*** that must be executed on a Data Collector, Skylar One will 1) Not execute the action policy; 2) Create a log entry in the audit log for the organization aligned with the triggering event, noting that the criteria in the automation policy were met, but that the action policy was not executed. This does not apply to Action Policies created on an All-In-One Appliance.

- ***Available Actions***. Displays a list of all action policies in Skylar One. (Action policies are defined in Registry > Run Book > Actions.) You can select one or more action policies in this field. If the selected event(s) and event criteria occur on the selected devices or for the selected IT Services, the selected action policies will be executed. To select an action policy, select it and click the right arrow-button.
- ***Aligned Actions***. This pane displays a list of all action policies aligned with this automation policy.
    - ***To de-select an action policy***, select it and click the left-arrow button.
    - ***To change the order in which one or more action policies are executed***, select the action policy and use the up-arrow or down-arrow to move the policy within the list.

NOTE: If you selected multiple action policies in the automation policy, the action policies will be executed in the order specified in the ***Aligned Actions*** field. To change the order of one or more action policies, select the action policy and use the up-arrow or down-arrow to move the policy within the list.

- **[Save]**. Saves a new automation policy or saves changes to an existing automation policy.
- **[Save As]**. If you supply a new value in the ***Policy Name*** field, saves the current automation policy, including any edits, as a new policy with a new name.

3. Click the **[Save]** button to save the new automation policy or save changes to an existing automation policy.

PowerPacks that Contain User-initiated Automations

The "Datacenter Automation Utilities" PowerPack is required for all Automation PowerPacks.

The "Datacenter Advanced Enrichment Actions" PowerPack is required for Cisco Automation PowerPacks.

NOTE: Follow the installation and configuration instructions in the release notes for the PowerPacks containing the automations you want to use. You must meet all prerequisites for each PowerPack before the automations from that PowerPack will be available in the **Tools** pane in Skylar One.

The following is a partial list of the PowerPacks that contain user-initiated automation policies:

- Cisco IOS Automation Policies PowerPack version 102 or later
- Cisco IOS-XE Automation Policies PowerPack version 102 or later
- Cisco IOS-XR Automation Policies PowerPack version 102 or later
- Cisco Networking Automation Policies PowerPack version 102 or later
- Linux SSH Automation Policies User-initiated Automation PowerPack
- VMware User-initiated Automation PowerPack
- Windows PowerShell User-initiated Automation PowerPack

For a complete list of PowerPacks containing user-initiated automation policies, see the **PowerPacks** page on the ScienceLogic Support Site (Skylar One > PowerPacks).

NOTE: To run user-initiated automations with a device group, the device group must use a *static* list of devices, not a dynamic list of devices.

Locating a User-initiated Automation

You can run a user-initiated automation from the **Run Book Actions** section of the **Tools** pane. The **Tools** pane is available on the **Device Summary** modal for an event associated with a device.

A **Device Summary** modal with a **Tools** pane is available in the following locations in Skylar One:

- The **Events** page, for all events that are aligned with a device. To open the **Device Summary** modal, click the open icon (↗) for that event.

TIP: If needed, click the Select Columns icon (⚙) and select *Automated Actions* to add that column to the **Events** page. This column lists the number of automated actions for each event, and you can click the number to open the **Event Actions Log** window.

- The **Event Investigator** page for an event. Select the event from the **Events** page to view the **Event Investigator** page.

> **TIP:** To see if an event was associated with an automation, click the **[Actions]** button (···) from the **Events** page and select *View Automation Actions*. The **Event Actions Log** window appears, with a list of all previously run automations. In the classic user interface, you can open the **Event Actions Log** window from the **Event Console** by clicking the View Notification Log icon ( ) when the value in the ***Notify*** column increases

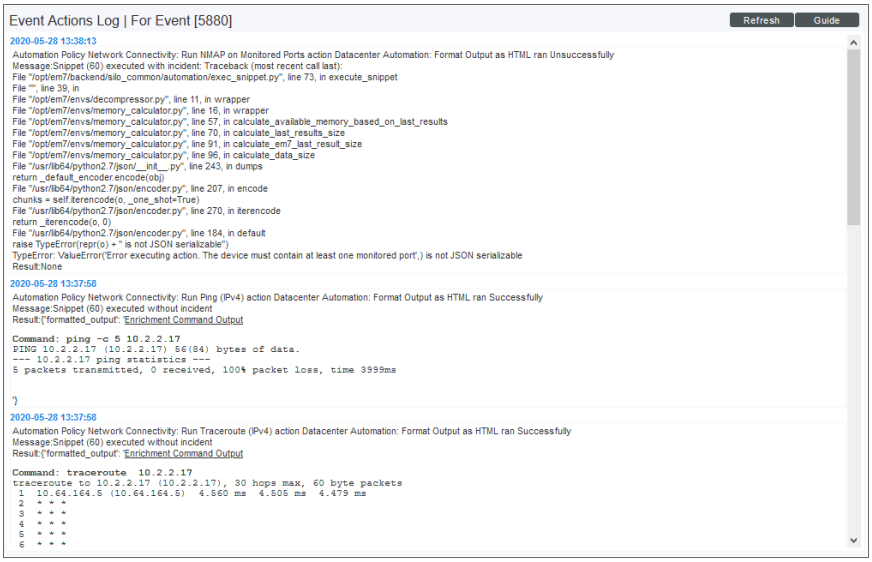
- The **Devices** page. To open the **Device Summary** modal, click the open icon (↗) for that event.
- The **Device Investigator** page. On the **[Events]** tab, click the open icon (↗) for that event.
- The **Service Investigator** page for a service. Select the service from the **Business Services** page to view the **Service Investigator** page, and then scroll down to the **[Events]** tab. To open the **Device Summary** modal, click the open icon (↗) for that event.

Running a User-initiated Automation

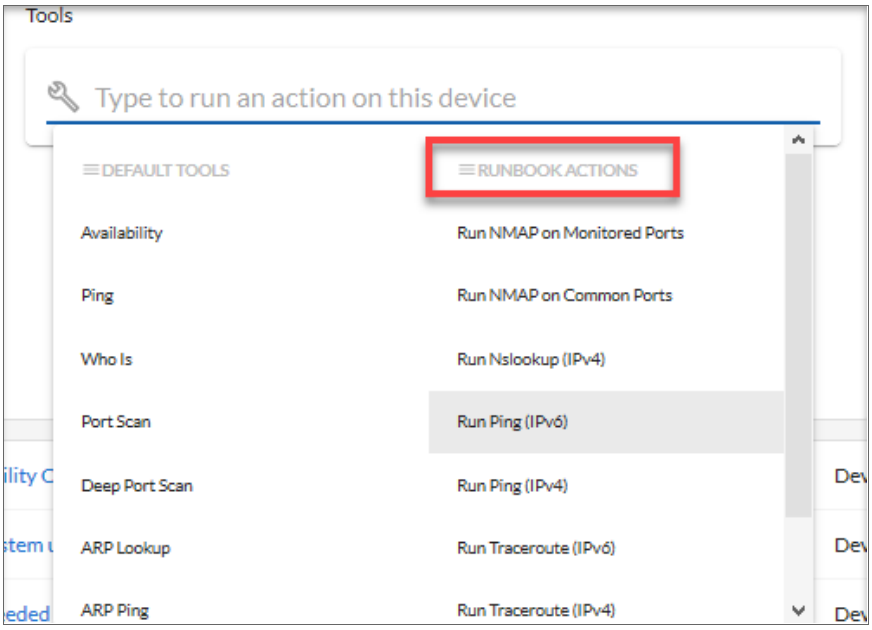
In most situations, you would run a user-initiated automation in response to an event that just occurred. If you have Automation PowerPacks installed on your Skylar One system, the **Event Actions Log** window for that event might contain diagnostic information from other automations that have already run, including information that helps you determine which user-initiated automation you should run next to address the cause of the event.

To run a user-initiated automation:

1. On the **Events** page, locate the event.
2. Click the Actions button (···) for the event and select *View Automation Actions*. The **Event Actions Log** window appears:



3. On the **Event Actions Log** window, review any existing automations that have already run, along with any diagnostic information or instructions that might appear for the event. Close the window when you are done.
4. On the **Events** page, click the open icon (↗) for that event and go to the **Tools** pane. You can also access the Tools pane from the **Devices** and **Service Investigator** pages.
5. From the **Run Book Actions** section on the **Tools** pane, click to run the relevant user-initiated automation:



6. After the action completes, click the **View Logs** link if needed. The **Event Actions Log** window displays updated information about the user-initiated automation.

Creating a User-initiated Automation

You can run or create a "user-initiated" automation policy that lets a user trigger an automation action (from an action policy) on an as-needed, "ad hoc" basis. A user-initiated automation does not require that the event-based logic in the **Criteria Logic** section matches to *True* in the automation policy as the user action supersedes that criteria.

Scheduling an Automation Policy

You can use an Automation Schedule to automatically trigger Run Book Automations based on time and day, or to restrict a Run Book Automation from running during certain time windows. The actions are executed according to the schedule, regardless of event status, or you can manually execute a scheduled automation policy at any time.

For example, you can use an automation schedule to:

- Run a specific Run Book Automation for a given point in time (such as tonight at midnight).
- Run a specific Run Book Automation at an interval (such as every day at 6:00 p.m.).
- Restrict a Run Book Automation from triggering a support event during normal work hours (such as 8:00 to 5:00).

Creating an Automation Schedule

You must create an Automation Schedule on the **Automation Schedule Manager** page before you can use it with an automation policy.

To create a schedule for an automation:

1. Navigate to the **Automation Schedule Manager** page (Registry > Run Book > Schedules).

> **NOTE**: After an automation schedule has been associated with an automation policy, you can click the + icon next to the schedule in the **Automation Schedule Manager** page to expand a list of all automation tasks that are associated with the schedule. A task can be associated with more than one schedule, so if you disable a task, it is disabled for *all* schedules associated with that task. You align a task with a schedule when you *create the scheduled automation policy*.

2. Click the **[Create]** button. A new **Schedule Editor** modal appears.
3. In the **Schedule Editor** modal, complete the following fields:

   Basic Settings

   - ***Schedule Name***. Type a name for the schedule.
   - ***Visibility***. Select the visibility level for the schedule. You can select one of the following:
     - *Private*. The schedule is visible only to the owner selected in the ***Owner*** field.
     - *Organization*. The schedule is visible only to the organization selected in the ***Organization*** field.
     - *World*. The schedule is visible to all users.
   - ***Organization***. If Visibility is set to "Organization", select the name of the organization with which this schedule is associated.
   - ***Owner***. Select the owner of the schedule. The default value is the username of the user who created the schedule.

   Time Settings

   - ***Start Time***. Click in the field and select the date and time you want the schedule to start.
   - ***End Time***. Leave this field blank. If you do set an end time here, this automation schedule will not be available to use with a scheduled Run Book Automation.

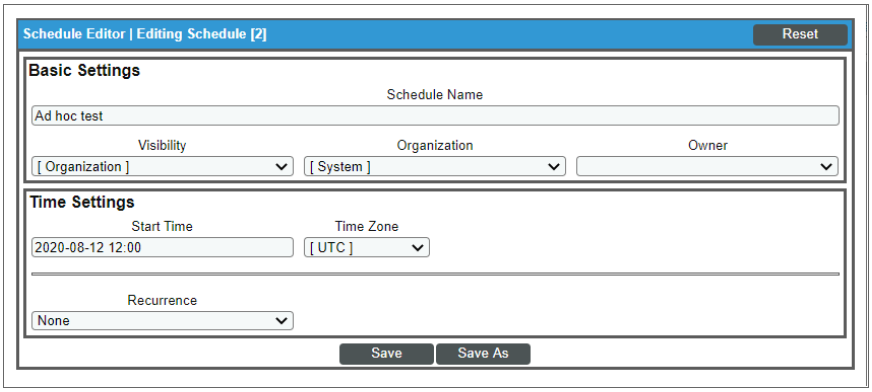
- ***Time Zone***. Select the region or time zone for the scheduled start time. If you want to automatically adjust for daylight savings time (if applicable), then you must select a named region (such as *America/New York*) in the ***Time Zone*** field. If you select a specific time zone (such as *EST*) or a specific time offset (such as *GMT-5*), then Skylar One will not automatically adjust for daylight savings time.
- ***All Day***. If you want this schedule to be valid for a single day, enter a *Start Time* and select this checkbox. The *End Time* selection will be grayed out.
- ***Recurrence***. Select whether you want the schedule to occur once or on a recurring basis. You can select one of the following:
    - *None*. The schedule occurs only once.
    - *By Interval*. The schedule recurs at a specific interval.

      If you select By Interval, specify the interval in frequency number, then select the time interval (*Minutes*, *Hours*, *Days*, *Weeks*, or *Months*). For example:

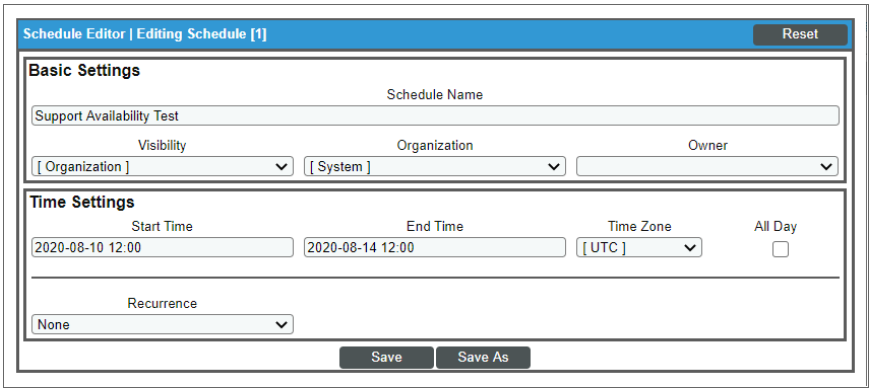
        - If you specify "6 Hours", then the schedule recurs every six hours from the time listed in the ***Start Time*** field.
        - If you specify "10 Days", then the schedule recurs every 10 days from the date listed in the ***Start Time*** field.
        - If you specify "2 Weeks", then the schedule recurs every two weeks, on the same day of the week as the ***Start Time***.
        - If you specify "3 Months" the ticket recurs every three months, on the same day of the month as the ***Start Time***.
    - *<Dynamic>*. The system will offer a dynamic option based on your entries in the fields above. For example, if you have created an automation schedule for date that is a first Monday of the month, your dynamic option would be *Every 1st Monday*. Use this for recurring Run Book Automations that you want to run on a particular day of the month.
- ***Recur Until***. Specifies when the schedule stops recurring. You can select one of the following:
    - *No Limit*. The schedule recurs indefinitely until you disable it.
    - *Specified Date*. The schedule recurs until a specific date and time. If you select *Specified Date*, you must select a date and time in the ***Last Recurrence*** field.
- ***Last Recurrence***. Click in the field and select the date and time you want the schedule to stop recurring.

4. Click **[Save]** to save the new schedule.

This is an example of an automation schedule for a one-time, ad hoc test:



This is an example of an automation schedule for a non-recurrent test within a date range:



Creating an Automation Policy Limited by Schedule

After you create a schedule for the automation on the **Automation Schedule Manager** page (Registry > Run Book > RunBook Schedules), you can create an event-driven automation policy that is limited to run based on an automation schedule.

To create an event-driven automation policy that is limited to run within an automation schedule:

1. Navigate to the **Automation Policy Manager** page (Registry > Run Book > Automation).
2. Click the **[Create]** button. The **Automation Policy Editor** page appears.
3. In the **Automation Policy Editor** page, supply a value in each of the following fields:
    - ***Policy Name***. Name of the automation policy.
    - ***Policy Type***. Select *Active Events*.
    - ***Policy State***. Specifies whether the policy can be executed (Enabled) or cannot be executed (Disabled).

- ***Organization***. Organization associated with the automation policy. If you select the *System* organization, the behavior of the ***Available Devices*** and ***Available Device Groups*** fields is affected.

> **NOTE:** The organization you select must match the organization of the Automation Schedule you specify in the criteria.

- ***Criteria Logic***. Specify criteria for the schedule you want to use to limit the automation policy.
    - *Time/Schedule*. Specifies the timespan during which the automation policy can execute the aligned actions. The choices for scheduled automations are:
        - *and the following schedule is active*. The automation policy can execute the aligned actions during the timespan specified in the selected schedule.
        - *and the following schedule is NOT active*. The automation policy can execute the aligned actions during any time *except* the timespan specified in the selected schedule.
    - *Schedule.* Select an available schedule in this field.
- ***Align With***. Specifies whether to align this automation policy with one or more devices, one or more device groups, or one or more organizations.
    - *Devices*. The ***Available Devices*** field will appear below, where you can select devices to associate with the automation policy.
    - *Device Groups*. The ***Available Device Groups*** field will appear below, where you can select device groups to associate with the automation policy.
    - *Policy Organization*. The ***Available Devices in Organization*** field will appear below, where you can select one or more devices to associate with the automation policy. The list of devices comprises all devices in the organization specified in the ***Organization*** field.
- ***Available Devices***. If you selected *Devices* in the ***Align With*** field, this field displays a list of all devices in Skylar One. You can select one or more devices in this field. The selected event(s) and event criteria must occur on one of the selected devices before the automation policy will be executed.
- ***Aligned Devices***. This pane displays a list of all devices aligned with the automation policy. To de-select a device, highlight it and click the left-arrow button.
- ***Aligned Device Groups***. This pane displays a list of all device groups aligned with this automation policy. To de-select a device group, highlight it and click the left-arrow button. Note that all devices contained in a device group are aligned with the automation policy when you align a device group.
- ***Available Devices in Organization***. If you selected *Policy Organization* in the ***Align With*** field, this field displays only devices from the organization selected in the ***Organization*** field. You can select one or more devices in this field. The selected event(s) and event criteria must occur on one selected device before the automation policy will be executed.

- ***Aligned Devices***. This pane displays a list of all devices from the specified organization that are aligned with this automation policy. To de-select a device, highlight it and click the left-arrow button.
- ***Available Actions***. Displays a list of all action policies in Skylar One. Action policies are defined in Registry > Run Book > Actions. You can select one or more action policies to run on your schedule. To select an action policy, highlight it and click the right arrow-button.
- ***Aligned Actions***. This pane displays a list of all action policies aligned with this automation policy.
    - ***To de-select an action policy***, highlight it and click the left-arrow button.
    - ***To change the order in which one or more action policies are executed***, highlight the action policy and use the up-arrow or down-arrow to move the policy within the list.

4. Click the **[Save]** button to save the new automation policy.

Creating an Automation Policy to Run on a Schedule

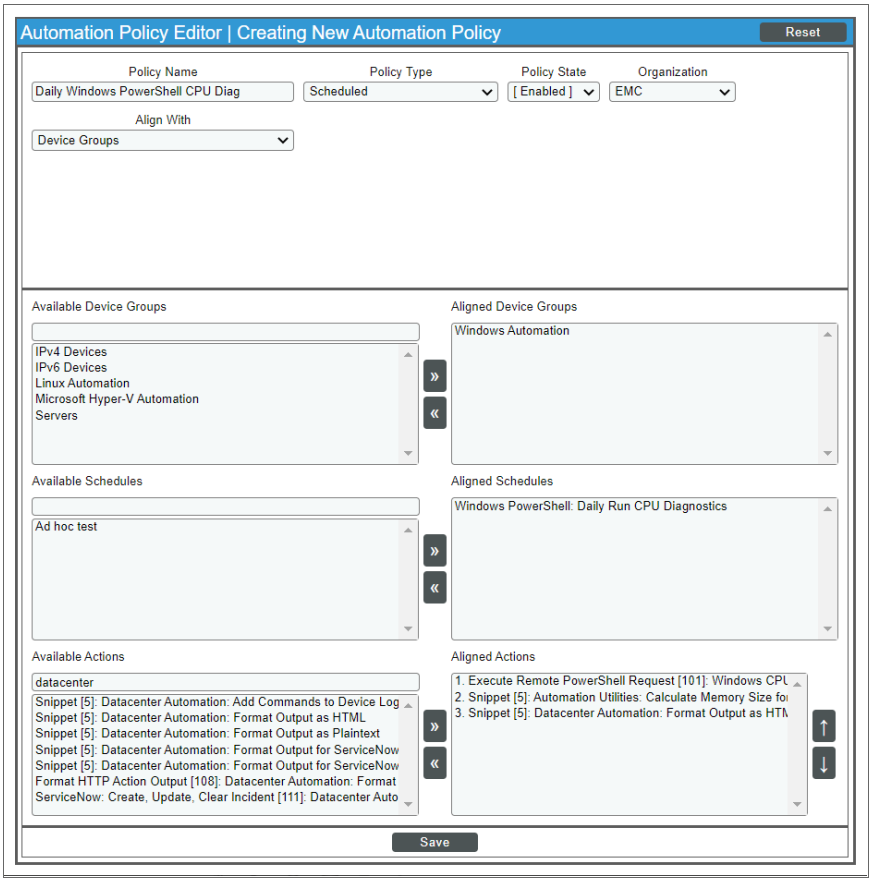
After you create an automation schedule on the **Automation Schedule Manager** page (Registry > Run Book > Schedules), you can create an automation policy to run on a schedule.

To create an automation policy that runs based on an automation schedule:

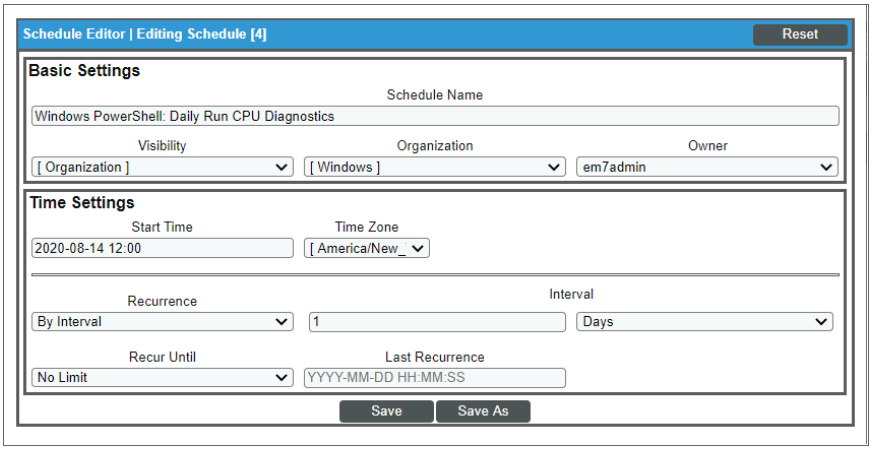
1. Navigate to the **Automation Policy Manager** page (Registry > Run Book > Automation).
2. Click the **[Create]** button. The **Automation Policy Editor** page appears.
3. On the **Automation Policy Editor** page, supply a value in each of the following fields:
    - ***Policy Name***. Name of the automation policy.
    - ***Policy Type***. Select *Scheduled*.
    - ***Policy State***. Specifies whether the automation policy can be executed (Enabled) or cannot be executed (Disabled).
    - ***Organization***. Organization associated with the automation policy. The organization should be the same as the one associated with the automation schedule you want to use.
    - ***Align With***. Specifies whether to align this automation policy with one or more devices, one or more device groups, or one or more organizations. Based on your setting, you must then specify the Devices, Device Groups, or Device in Organization in the field provided, as listed below.
        - *Devices*. The *Available Devices* field will appear below, allowing you to select devices to associate with the automation policy.
        - *Device Groups*. The *Available Device Groups* field will appear below, allowing you to select device groups to associate with the automation policy.
        - *Policy Organization*. The *Available Devices in Organization* field will appear below, allowing you to select one or more devices to associate with the automation policy. The list of devices comprises all devices in the organization specified in the Organization field.

- ***Available Schedules***. Displays a list of all automation schedules from the **Automation Schedule Manager** page that are not time-limited, that is, that do not have an set *End Time*. You can select one or more schedules in this field. To select an automation schedule, highlight it and click the right-arrow button.
- ***Aligned Schedules***. This pane displays a list of all automation schedules aligned with this automation policy. To de-select an automation schedule, highlight it and click the left-arrow button. To change the order in which one or more automation schedules are evaluated, highlight the action policy and use the up-arrow or down-arrow to move the policy within the list. If you selected multiple automation schedules in the automation policy, the automation schedules will be evaluated in the order specified in the ***Aligned Schedules*** field.

In the example below, we have created an automation policy to run daily CPU diagnostics on our "Windows Automation" device group using the Windows PowerShell: Daily Run CPU Diagnostics schedule we created. We selected Aligned Actions based on the tasks we want to happen during the automation run.



The automation schedule we created runs every day at 12:00 (America/NewYork timezone) with no end date. This automation schedule allows our automation policy to continue until we disable it.



Manually Executing a Scheduled Automation Policy

When you create a scheduled automation policy, Skylar One will execute that automation policy according to the schedule, regardless of event status. However, you can also manually execute the scheduled automation policy at any time.

To manually execute a scheduled automation policy:

1. Navigate to the **Automation Policy Manager** page Registry > Run Book > Automation).
2. Locate the scheduled automation policy you want to manually execute and click its lightning bolt icon (). A confirmation message appears.
3. Click **[OK]** to continue. Depending on the policy, one of the following will happen:
    - If the scheduled automation policy is enabled and has at least one schedule aligned with it, the policy will execute immediately.
    - If the policy is disabled or does not have a schedule aligned with it, Skylar One displays an error message and the policy will not execute.

"Clear" Policies

In an automation policy, the ***Policy Type*** field specifies whether the policy will be evaluated against active events or against cleared events.

If you create an automation policy with a ***Policy Type*** of *Clear*:

- The automation policy will be evaluated only for cleared events.
- The automation policy will contain only options for matching severity (***Criteria Logic*** fields), matching ticket created or not created status (***Criteria Logic*** fields), and matching text in an event message (***Match Logic*** and ***Match Syntax*** fields).
- The automation policies will run only once (when the event is cleared) for any given event.

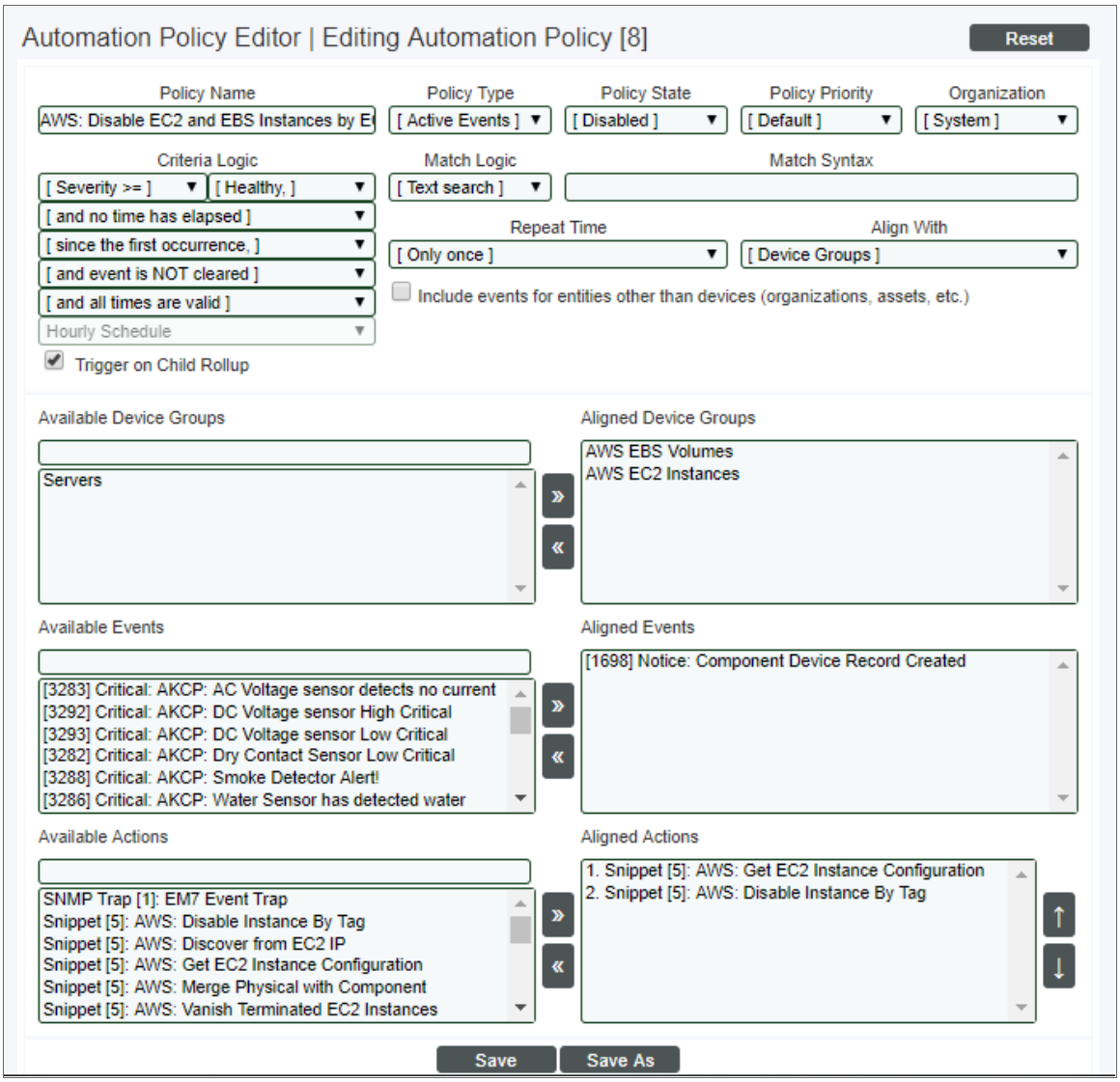
Aligning an Automation Policy with the System Organization

In an automation policy, the ***Organization*** field specifies the organization to associate with the policy and tells the automation policy which devices to evaluate. If you select the *System* organization in the ***Organization*** field, the behavior of the ***Available Devices*** field is affected.

- If you selected *Devices* in the ***Align With*** field, the ***Available Devices*** field is displayed in the **Automation Policy Editor** page.
- In the ***Available Devices*** field, you can select one or more devices. The selected event(s) and event criteria must occur on at least one of the selected devices before the automation policy will be executed.
    - ***If you do not select any devices***, the automation policy automatically evaluates all devices associated with the organization you selected in the ***Organization*** field.
    - ***If you do not select any devices and you selected System in the Organization field***, the automation policy automatically evaluates ***all devices in Skylar One***.

Ordering Actions in an Automation Policy

You can align multiple action policies with a single automation policy. In addition, you can specify the order in which the Skylar One system executes those aligned action policies.



Action policies can use the variable %_EM7_RESULT_% to retrieve the results from the previously executed action policy. Therefore, it is important that you understand the dependencies between action policies before you specify the order in which aligned action policies are executed.

For details on the variable %_EM7_RESULT_%, see *Using the Result of a Previous Action*.

Automation Policies and Event Masks and Event Correlation

In Skylar One, events can be grouped together in a suppression group using event correlation or event masks. These grouped events can affect run book criteria.

If you selected the checkbox ***Trigger on Child Rollup***, both the parent and all the child events in a suppression group can trigger the automation policy.

If you do not select the checkbox ***Trigger on Child Rollup***, the default behavior is:

- For event correlation, only the parent event can trigger the automation policy.
- For event masks, only the event with the highest severity can trigger the automation policy. If multiple events have the highest severity, only the event with the highest severity and the earliest timestamp can trigger the automation policy.

Events Not Displayed on the Events Page that Might Affect Automation Policies

There are four types of events that might not be displayed on the **Events** page. Two of them have an effect on Automation Policies:

- ***Topology Events***. In Skylar One, event correlation or topology suppression means the ability to build parent-child relationships between events. When events are correlated, only the parent event is displayed on the **Events** page. The child events are rolled up and nested under the parent event and are displayed only if you click on the magnifying-glass icon (🔍). For the parent event, the count column will be incremented to indicate the number of correlated child events.
- ***Event Masks***. In the **Device Properties** page for each device, you can define an Event Mask. When a device uses the Event Mask setting, events that occur on a single device within a specified span of time are grouped together. On the **Events** page, masked events are nested under the event with the highest severity. The magnifying-glass icon (🔍) appears to the left of the event. When you click on the magnifying-glass icon, the nested events are displayed.

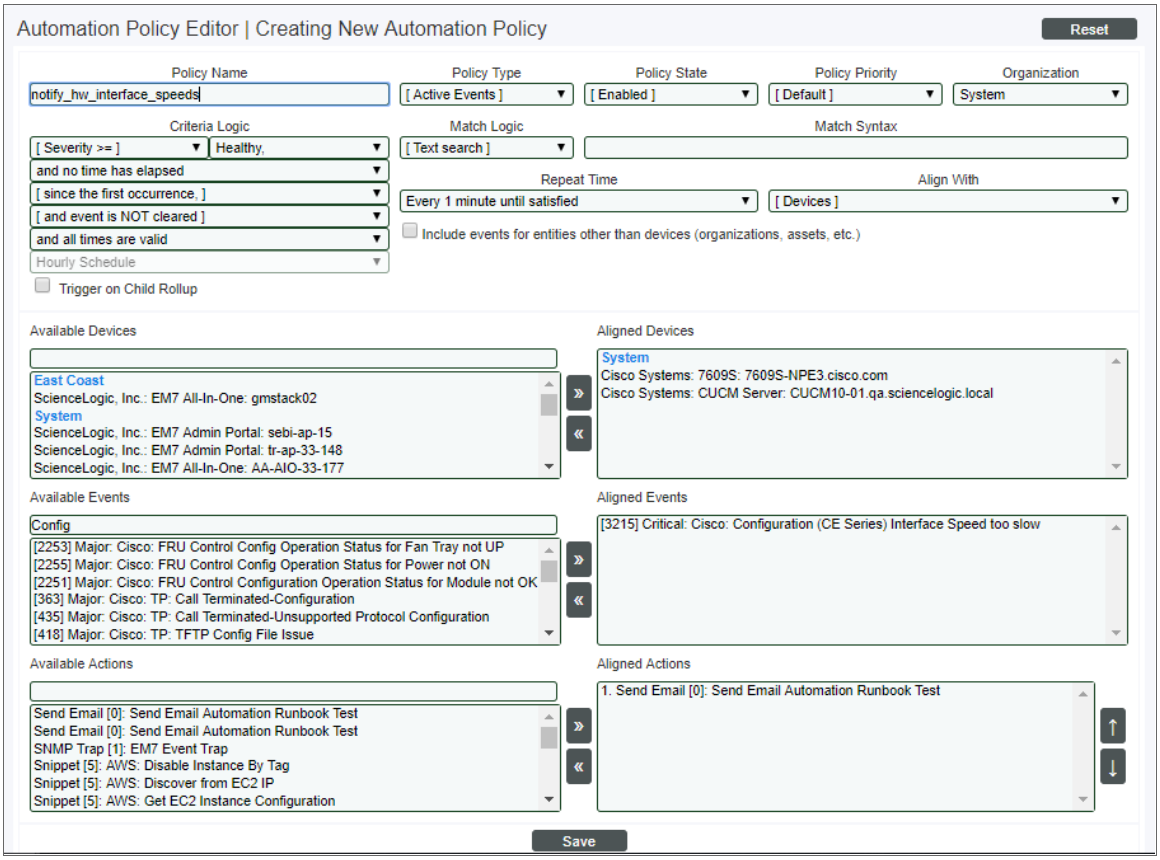
The first time an event triggers an automation policy, Skylar One will check to see if that event is the parent event of a suppression group due to topology events or an event mask. If the event is part of a suppression group, Skylar One will trigger the automation policy only if the event is the parent event in the suppression group. Only that single event will trigger the automation policy; other events in the suppression group will not trigger the automation policy. For all future instances, only that event with the highest severity will trigger the automation policy.

Example

- Suppose you have a high-security project that requires hardware to be extremely responsive.
- Suppose this project uses Cisco network hardware.
- Suppose you want to notify key personnel immediately if the Cisco network hardware is too slow.
- You could define an automation policy that specifies the Cisco hardware to monitor and the event that is triggered when speeds are too slow.
    - The event is called "Critical: Cisco: Configuration Interface Speed to Slow".

- You could align the automation policy with an action policy that sends an email to key personnel. The action policy could send these emails to the handheld devices for these key personnel.
- The action policy is called "Email_sysadmins".

Our example automation policy might look like this:



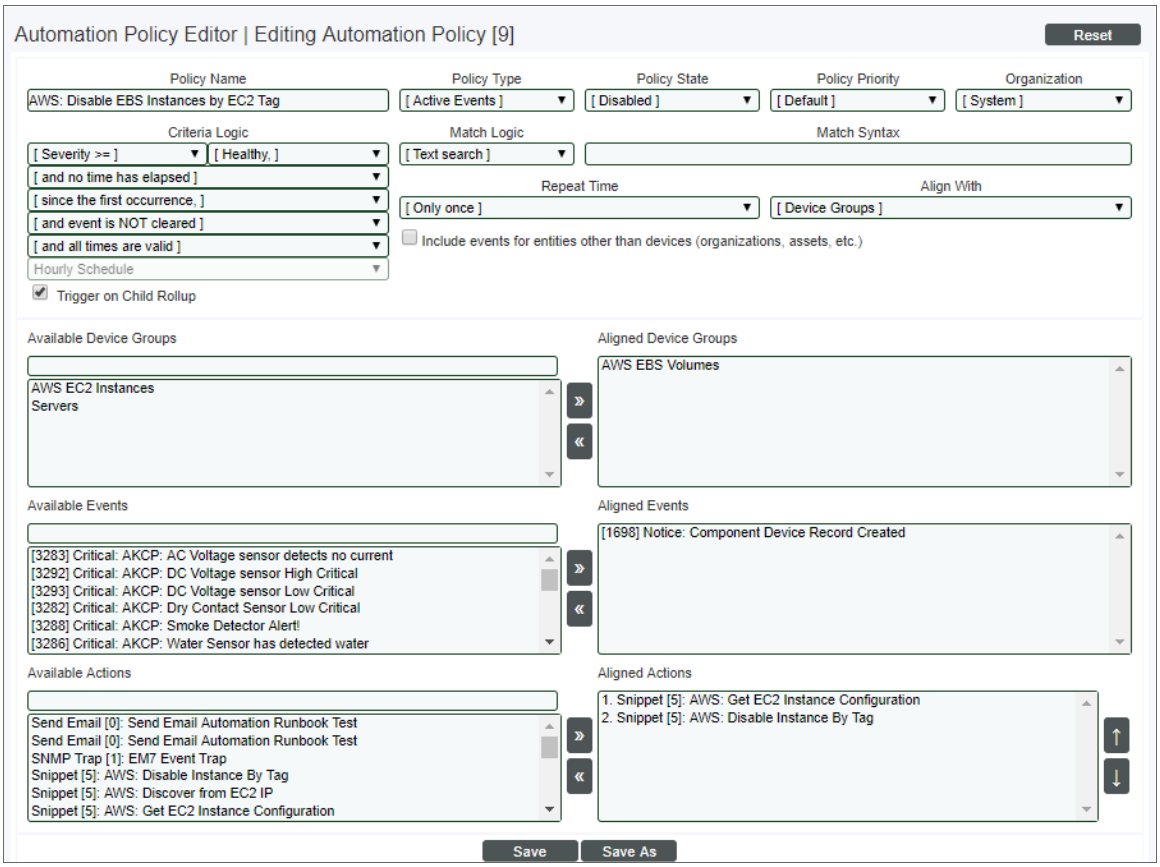
- We specified that the automation policy:
  - Should act upon active events.
  - Is enabled.
  - Is associated with the organization "System".
  - Will be triggered by the specified event when the event has a severity greater than "Healthy".
  - Will be triggered as soon as the specified event occurs.
  - The policy will continue to trigger the action every 1 minute until the event is cleared.
  - Will be triggered when the selected event occurs on at least one of the selected Cisco devices.
  - Will be triggered when the event "Critical: Cisco: Configuration Interface Speed to Slow" occurs on the selected Cisco devices.
  - Will be triggered at all times.

- We specified that when all the criteria in the automation policy are met, the action policy "Send Email" will be executed.

Editing an Automation Policy

You can edit any parameters of an existing automation policy. To do so:

1. Navigate to the **Automation Policy Manager** page (Registry > Run Book > Automation). In the **Automation Policy Manager** page, find the automation policy you want to edit. Click its wrench icon ( ).
2. The **Automation Policy Editor** modal page appears, populated with values from the selected automation policy.



3. You can edit the values in one or more fields. For a description of each field, see the previous section on *creating an automation policy*.
4. Click the **[Save]** button to save your changes to the automation policy.

Bulk Actions for Automation Policies

From the **Automation Policy Manager** page (Registry > Run Book > Automation), you can enable, disable, or delete one or more policies as a bulk action.

1. Navigate to the Automation Policy Manager page (Registry > Run Book > Automation).
2. In the Automation Policy Manager page, select the checkbox next to any policy you want to enable, disable, or delete. You can select more than one policy.
3. Go to the ***Select Action*** field in the lower right corner of the page. Select the action you want to take: *Delete Policies*, *ENABLE these Automation Policies*, or *DISABLE these Automation Policies*.
4. Click the **[Go]** button.

Updating Run Book Engine Settings

The process that executes Run Book tasks is automated, and the default Run Book Automation executor settings are appropriate for most Skylar One systems. However, users can change their Run Book Automation settings to support extremely large Skylar One systems. More specifically, users can adjust the settings for the executor count respective to their primary task's queue preference.

Users can configure the automation policies to be of a specific type of queue. The policy settings that a user chooses determines which task queue the automation engine will group to.

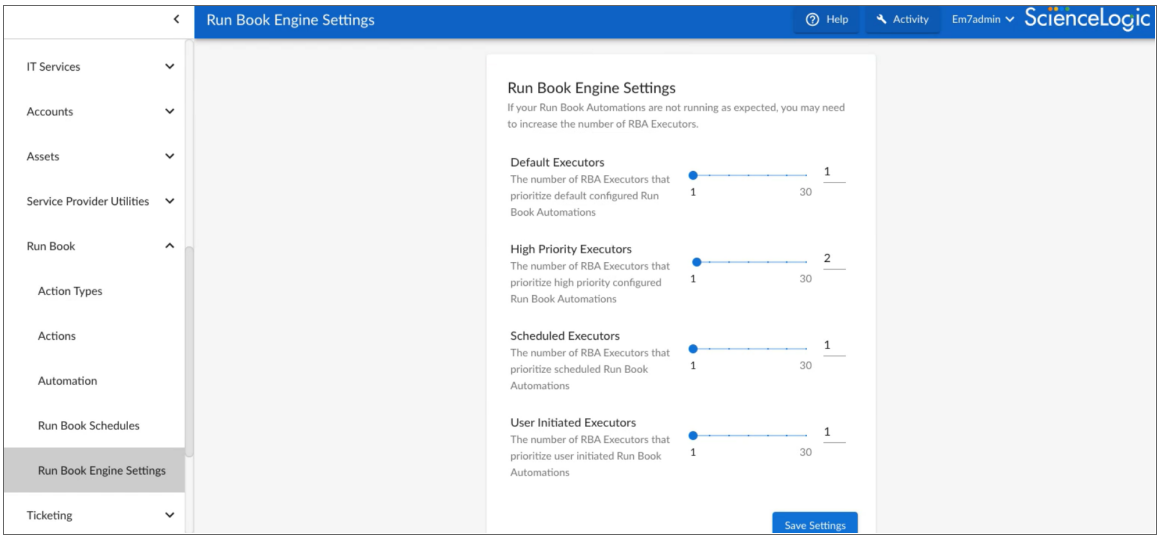
An ***RBA Executor*** is an individual process that executes automation tasks. If your Run Book Automations are not running as expected, you might need to increase the number of RBA Executors.

On the **Run Book Engine Settings** page, you can edit the following settings related to RBA Executors:

- **Default Executors**. The number of RBA Executors that prioritize default configured Run Book Automations. The default is 1.
- **High Priority Executors**. The number of RBA Executors that prioritize high priority configured Run Book Automations. the defalut is 2.
- **Scheduled Executors**. The number of RBA Executors that prioritize scheduled Run Book Automations. The default is 1.
- **User-Initiated Executors**. The number of RBA Executors that prioritize user-initiated Run Book Automations. The default is 1.

To adjust the number of executors on this page:

1. Slide the dial next to the desired task queue to the appropriate number. Or enter the number in the text box located to the right of sliding dial.
2. Slide the dial next to the desired task queue to the appropriate number. Or enter the number in the text box located to the right of sliding dial.
3. Click **[Save Settings]** after you make any edits to this page.



NOTE: All of the queues are shared by the automation engine. Meaning, that each executor does not have different queues containing different data. Also, depending on the settings' configuration, the executors will consume the queues according to the selected automation group. If there is no task present in the primary queue, the automation will move to the next queue.

To assure timely job execution for respective queue tasks loads, the executor count for those queue should be sized appropriately. For example, if a considerable number of your Automation policy executions are configured to utilize the High Priority Queue, a single executor may not be sufficient to handle the queue on its own and rely on other queue workers to assist (if they're available). By setting executor counts respective to expected queue tasks loads expected a Skylar One administrator can ensure executors prioritize accordingly.

For any major setting adjustments, please contact ScienceLogic Customer Support.

Chapter 3

Action Policies

Overview

An ***action policy*** is an action that can be automatically triggered in Skylar One when certain criteria are met. The triggers are defined in an automation policy (Registry > Run Book > Automation). For details on automation policies, see the section on *Automation Policies*.

An action policy can perform one of the following tasks:

- Send an email message to a pre-defined list of users and/or external contacts.
- Send an SNMP trap from Skylar One to an external device.
- Write an SNMP value to an existing SNMP object on an external device.
- Create a new ticket (using ticket templates defined in the **Ticket Templates** page [Registry > Ticketing > Templates]).
- Update an existing ticket. An action policy can change the status and/or severity of an existing ticket and/or add a note to an existing ticket. For this action policy to trigger successfully, a ticket must be associated with the event that triggered the action.

> **NOTE**: For more details on ticket templates, see the chapter on ticket templates in the ***Ticketing*** manual.

- Query a database.
- Run a custom python script, called a ***snippet***.
- Send a Message to a Topic ARN (Amazon Resource Name). All subscribers to the Topic ARN will receive the message.

- Run an Integration Service application.
- Send an HTTP request.

This chapter will describe how to create each type of action policy.

- If you want to trigger multiple actions when certain event criteria are met, you can define your automation policy to include multiple action policies.
- In an automation policy that will trigger multiple actions, you can specify the order in which the action policies are executed.
- In addition, the result of each action is available to the next executed action policy and can be accessed with the variable %_EM7_RESULT_%. You can define an action policy that uses the results of the previous action policy.

What is an Action Policy?

This chapter describe how to create an action policy. An ***action policy*** is an action that can be automatically triggered in Skylar One when certain criteria are met.

This chapter covers the following topics:

Viewing the List of Action Policies

The **Action Policy Manager** page (Registry > Run Book > Actions) displays a list of all existing action policies.

NOTE: Users of type "user" can view only action policies that are aligned with the same organization (s) to which the user is aligned. Users of type "administrator" can view all action policies.

TIP: You can sort the list by a column's values by clicking its column heading. You can also filter the items on the page by typing or selecting filter options in one or more of the filters found above the columns. For more information, see the topic on *Filtering Inventory Pages*.

To view the list of action policies:

1. Navigate to the **Action Policy Manager** page (Registry > Run Book > Actions).
2. The **Action Policy Manager** page displays the following about each action policy:
    - ***Action Name***. Name of the action policy.
    - ***Action Type***. Action that will be executed by the action policy. Choices are:
        - *Send an Email Notification*. Sends an email message. You can specify the content of the message and the users to whom it will be sent.

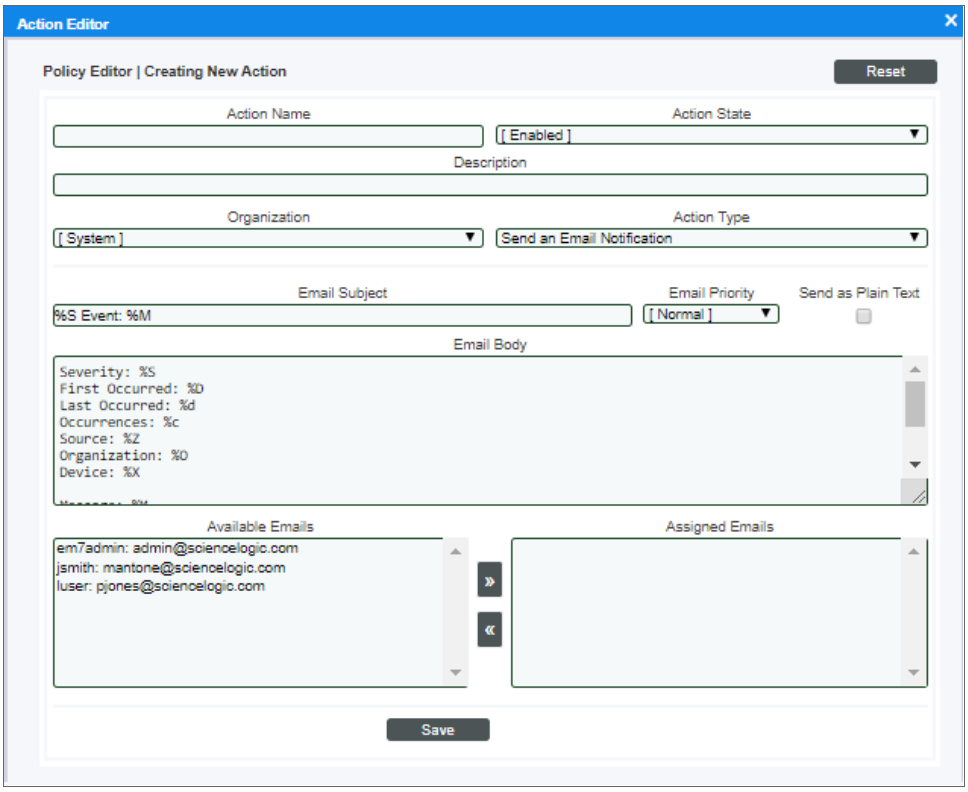
- *Send an SNMP Trap*. Sends an unsolicited SNMP message to an external system, using the ScienceLogic MIB files and predefined variables.
- *Create a New Ticket*. Creates a new ticket, using the Ticket Templates defined in Skylar One.
- *Send an SNMP Set*. Writes a value to an SNMP variable on an external device.
- *Run a Snippet*. Executes a snippet. A snippet is a custom program, written in Python.
- *Execute an SQL Query*. Either retrieve values from an external database or write a value to an external database. For distributed systems, the query can be sent from the Database Server or a Data Collector.
- *Update an Existing Ticket*. Updates an existing ticket. The action can add notes, change the severity, and change the status of the ticket.
- *Send an AWS SNS*. Sends an SNS Message to a Topic ARN (Amazon Resource Name). All subscribers to the Topic ARN will receive the message. This action type supports the use of a proxy, which must be configured in the AWS credential in Skylar One.
- *Run Integration Service Application*. Triggers an Integration Service integration application. When you create an Automation Action using this Action Type, you specify the credential for the Integration Service instance, the application to run, and the parameters to include in the request. This Action Type is available in the *Integration Service Action Type* PowerPack.
- *Send an HTTP request*. Sends an HTTP request to one or more devices. When you create an Automation Action using this Action Type, you specify the credential or Dynamic Appliaction GUID, a relative URL, and the request payload. This Action Type is available in the *HTTP Action Type* PowerPack.
- *Custom Action Type*. A Custom Action Type executes a reusable snippet. Unlike the Action Type "Snippet", a Custom Action Type can accept input parameters (in a JSON format) and create output (in a JSON format). A Custom Action Type allows a single snippet to be used in multiple Action Policies, each time with different inputs and different outputs.

- ***ID***. Unique numeric identifier, automatically assigned by Skylar One to each action policy.
- ***Action State***. Specifies whether the policy can be executed by an automation policy (enabled) or cannot be executed (disabled).
- ***Organization***. Organization associated with the action policy.
- ***Edit User***. User who created or last edited the action policy.
- ***Edit Date***. Date and time the action policy was created or last edited.

Creating an Action Policy

To create an action policy:

1. Navigate to the **Action Policy Manager** page (Registry > Run Book > Actions).
2. In the **Action Policy Manager** page, click the **[Create]** button.
3. The **Action Policy Editor** modal page appears.



4. In the **Action Policy Editor** page, supply a value in each field.
5. For all types of action policies, the first four fields are the same.
   - ***Action Name***. Specify the name for the action policy.
   - ***Action State***. Specifies whether the policy can be executed by an automation policy (enabled) or cannot be executed (disabled).
   - ***Description***. Allows you to enter a detailed description of the action.
   - ***Organization***. Organization to associate with the action policy.
   - **Action Type**. Type of action that will be executed. Your choices are:
     - *Send an Email Notification*. Sends an email message. You can specify the content of the message and the users to whom it will be sent.
     - *Send an SNMP Trap*. Sends an unsolicited SNMP message to an external system, using the ScienceLogic MIB files and predefined variables.
     - *Create a New Ticket*. Creates a new ticket, using the Ticket Templates defined in Skylar One.
     - *Send an SNMP Set*. Writes a value to an SNMP variable on an external device.

- *Run a Snippet*. Executes a snippet. A snippet is a custom program, written in Python.
- *Execute an SQL Query*. Either retrieve values from an external database or write a value to an external database. For distributed systems, the query can be sent from the Database Server or a Data Collector.
- *Update an Existing Ticket*. Updates an existing ticket. The action can add notes, change the severity, and change the status.
- *Send an AWS SNS Message*. Sends an SNS Message to a Topic ARN (Amazon Resource Name). All subscribers to the Topic ARN will receive the message. This action type supports the use of a proxy, which must be configured in the AWS credential in Skylar One.
- *Run Integration Service Application*. Triggers an Integration Service integration application. When you create an Automation Action using this Action Type, you specify the credential for the Integration Service instance, the application to run, and the parameters to include in the request. This Action Type is available in the *ScienceLogic: Integration Service Action Type* PowerPack.
- *Send an HTTP request*. Sends an HTTP request to one or more devices. When you create an Automation Action using this Action Type, you specify the credential or Dynamic Appliaction GUID, a relative URL, and the request payload. This Action Type is available in the *HTTP Action Type* PowerPack.
- *Custom Action Type*. A Custom Action Type executes a reusable snippet. Unlike the Action Type "Snippet", a Custom Action Type can accept input parameters (in a JSON format) and create output (in a JSON format). A Custom Action Type allows a single snippet to be used in multiple Action Policies, each time with different inputs and different outputs.

- **[Save]**. Saves a new action policy or saves changes to an existing policy.
- **[Save As]**. If you supply a new value in the ***Action Name*** field, saves the current action policy, including any edits, as a new policy with a new name.

6. The remaining fields will vary, depending upon the value you selected in the ***Action Type*** field.

Creating a Custom Action Type

A ***Custom Action Type*** executes a reusable snippet. Unlike the Action Type "Snippet", a Custom Action Type can accept input parameters (in a JSON format) and create output (in a JSON format). A Custom Action Type allows a single snippet to be used in multiple Action Policies, each time with different inputs and different outputs.

A Custom Action Type is associated with an Execution Environment. An ***execution environment*** is an on-demand Python environment that includes the supporting modules, code, scripts, directories, and files (packaged in one or more ScienceLogic Libraries) required by the Custom Action Type. ***ScienceLogic Libraries*** are packages consisting of metadata and Python files that can be used by the Run Book Actions that use snippets.

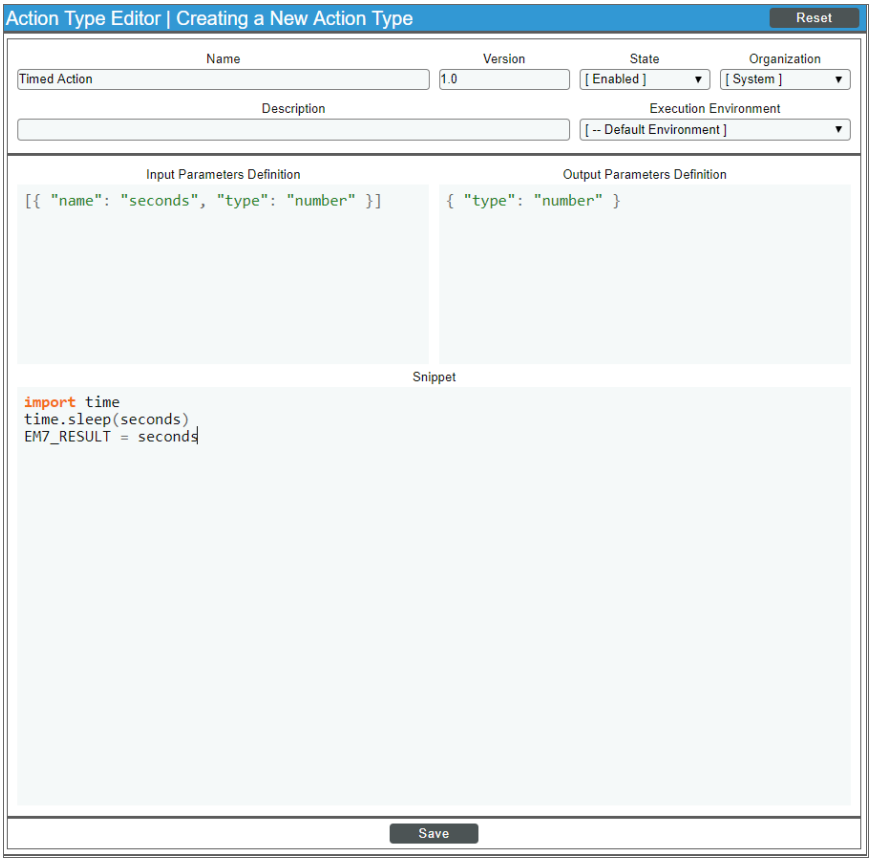
You can create and edit Custom Action Types on the **Action Type Manager** page Registry > Run Book > Action Types).

The **Action Type Manager** page displays a list of all Custom Action Types and any PowerPacks that include a Custom Action Type.

TIP: To create a new Custom Action Type based on an existing Custom Action Type, you can change the values in one or more of the following fields, supply a new name for the edited Custom Action Type, and click the **[Save As]** button.

To create a New Custom Action Type:

1. Navigate to the **Action Type Manager** page (Registry > Run Book > Action Types).
2. On the **Action Type Manager** page, click the **[Create]** button. The Action Type Editor page appears:



3. Complete the following fields:
   - ***Name***. Specify the name of the Custom Action Type. Can be any combination of alpha-numeric characters, up to 255 characters in length.
   - ***Version***. Version number for the Custom Action Type. Can be any combination of alpha-numeric characters, up to 64 characters in length.

- ***State***. Specifies whether the Custom Action Type can be executed by an action policy (Enabled) or cannot be executed (Disabled).
- ***Organization***. The organization associated with the Custom Action Type.
- ***Description***. A description of the action type. Can be any combination of alpha-numeric characters, up to 255 characters in length.
- ***Execution Environment***. Select from the list of available Execution Environments. The default execution environment is *System*.This list also shows the Python version of each selectable execution environment
- ***Input Parameters Definition***. A JSON structure that specifies each input parameter. Each parameter definition includes its name, data type, and whether the input is optional or required for this Custom Action Type. For example:

```
[{"name": "param", "type": "string", "required": true}]
```

NOTE: Input parameters must be defined as a JSON structure, even if only one parameter is defined.

NOTE: Currently, the only supported input data types are "number", "string", and "boolean".

- ***Output Parameters Definition***. A JSON structure that specifies the output parameter. The parameter definition must includes its name and data type. For example:

```
[{"name": "success", "type": "boolean"}]
```

NOTE: The output parameter definition is limited to a single parameter.

- ***Snippet***. Specify the python code that will be executed when Skylar One runs the action policy associated with this Custom Action Type. For example:

```
EM7_RESULT = {"success": True, "data": param}
```

4. Click the **[Save]** button to save the new action type.

Examples:

Here is an example Custom Action Type called "Timed Action":

- ***Name***. Timed Action.
- ***Version***. 1.0
- ***State***. Enabled
- ***Organization***. System
- ***Description***.

- ***Execution Environment***. System
- ***Input Parameters Definition***.

```
[{ "name": "seconds", "type": "number" }]
```

- ***Output Parameters Definition***.

```
{ "type": "number" }
```

- ***Snippet***.

```
import time
```

```
time.sleep(seconds)
```

```
EM7_RESULT = seconds
```

Here is an Action Policy called "Sleep for 2 Seconds" that uses the Custom Action Type called "Timed Action":

- ***Action Name***. Sleep For 2 Seconds
- ***Action State***. Enabled
- ***Description***.
- ***Organization***. System
- ***Action Type***. Timed Action (1.0)
- ***Execution Environment***. Default
- ***Action Run Context***. Database
- ***Input Parameters***.

```
{ "seconds": 2 }
```

Creating an Action Policy that Sends an Email Notification

In the **Action Policy Editor** page, if you selected the ***Action Type*** of *Send an Email Notification*, the new action policy will send an email message. You can specify the content of the email message and the users to whom the email message will be sent. If the action is aligned with an automation policy (i.e., if the action policy is included in the definition of an automation policy), and the criteria in the automation policy are met, the email message will be sent.

An action policy that sends an email notification is useful when you must immediately inform key personnel about event conditions.

NOTE: When an automation policy executes actions, the time stamps for the actions will use the time zone defined in the ***System Timezone*** field (System > Settings > Behavior page). However, the timestamp for the Event Action Log window will display the user's local time zone, as defined in the **Account Information** page (Preferences > Account > Information). For more information, see "My Contact Information" section of the ***Introduction to Skylar One*** manual.

NOTE: In the ***Email Subject*** and ***Email Body*** fields, you can use one or more of the variables from the *list of variables*. The value of each variable will be retrieved from the event that triggered the automation policy.

NOTE: When action policies send email notifications, they use the "BCC" email field by default. Optionally, you can configure action policies to send email notifications using the "To" email field instead. To do so, go to the **Database Tool** page (System > Tools > DB Tool). The **Database Tool** page is available only in versions of Skylar One prior to 12.2.1 and displays only for users that have sufficient permissions to access the page.

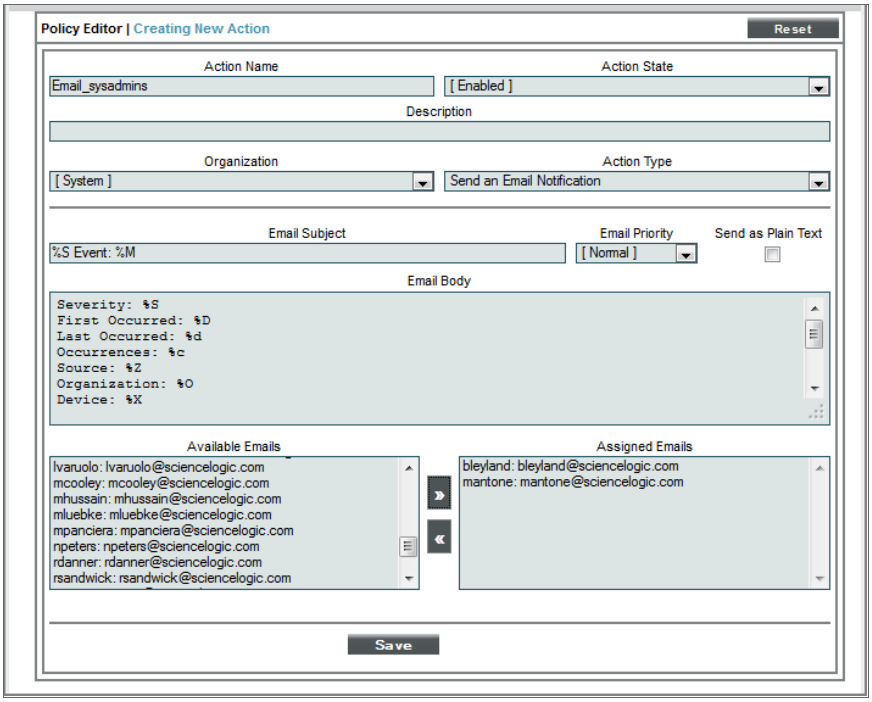
Enter the following in the ***SQL Query*** field, replacing `<value>` with either 'bcc' or 'to':

```
UPDATE
   master.system_settings_com
SET
   outbound_to_flag =< value >
```

Changing this setting affects all action policies that send email notifications.

NOTE: In the **Behavior Settings** page (System > Settings > Behavior), make sure that the value in the ***Interface URL*** does not include a trailing forward slash (/). When Skylar One generates URLs for tickets or events (and includes those URLs in email messages), the trailing forward slash causes problems with the generated URL.

To define an action policy that sends an email notification, you must supply values in the general fields, as specified in the *section on Creating an Action Policy* and also supply values in the following fields:



- ***Email Subject***. This will be the subject text in the outgoing email message. By default, the subject will be:

  %S Event: %M.

  where %S is the event's severity and %M is the message that appears on the **Events** page when the event occurs.

- ***Email Priority***. You can select *High*, *Normal*, or *Low*. However, be aware that email clients each handle priority differently.
- ***Send as Plain Text***. Select this checkbox if you want the email sent as plain text, without any special formatting.

- ***Email Body***. The body of the outgoing email message. You can include additional variables from the *list of variables* in the email body. By default, the body will be:

  Severity: %S

  First Occurred: %D

  Last Occurred: %d

  Occurrences: %c

  Source: %Z

  Organization: %O

  Device: %X

  Message: %M

  Sent by Automation Action: %N

  View this event at: %H

> **NOTE:** If you want to specify that the email be sent as an HTML message, include "<html><body>" at the very beginning and "</body></html>" at the very end of the ***Email Body*** field.

- ***Available Emails***. List of all email addresses associated with users and external contacts. You can select one or more email addresses to align with the action. To select an email address, highlight and then click the right-arrow button. The email address will then appear in the ***Assigned Emails*** pane. If the action is associated with an automation policy, and the criteria in the automation policy are met, the Skylar One system will send an email to the users and external contacts in the ***Assigned Emails*** pane.
- ***Assigned Emails***. If the action is associated with an automation policy, and the criteria in the automation policy are met, Skylar One will send an email message to the selected email address (es).

Creating an Action Policy that Sends an SNMP Trap

In the **Action Policy Editor** page, if you selected the ***Action Type*** of *Send an SNMP Trap*, the new action policy will send an unsolicited SNMP message to a device. If the action is associated with an automation policy, and the criteria in the automation policy are met, the Skylar One system sends the SNMP trap to the specified device. When you select this type action type, you must manually build the trap that will be sent. You do so in the **Action Policy Editor**.

An SNMP trap is a message is initiated by a network device or network application and sent to a network management system. For example, a router could send a message if one of its redundant power supplies fails or a printer could send an SNMP trap when it is out of paper.

An action policy that sends an SNMP trap is useful when you want to integrate Skylar One with an existing network management system. When certain event conditions are met (as defined in the automation policy), Skylar One can build an SNMP trap to pass the event information to another network management system.

Using the Default ScienceLogic MIBs to Build an SNMP Trap

When you create an action policy that sends an SNMP trap, you must manually build the trap that will be sent. You build the trap in the **Action Policy Editor** page. In the action policy, you assign an OID number to the trap. One or more variables can be included in the trap. These variables are called ***varbinds***. A varbind is referenced by an OID number, has a data type, and stores a dynamic value. You also define the varbinds in the **Action Policy Editor** page. For each varbind in the trap, you define the OID number, data type, and value.

If the receiver of the trap will perform actions based on the trap, best practice is to define a MIB file and send it to the receiver. This allows the receiver to decode and act upon the trap.

Default Traps from Skylar One

In most cases, you can use the default ScienceLogic MIB files to build SNMP traps from Skylar One. When you use the ScienceLogic MIB files, you are not required to define your own MIB files. You can simply export the ScienceLogic MIB files and send them to the trap receiver. When building traps in the **Action Policy Editor** page, you can then use the trap OIDs and varbind OIDs defined in the ScienceLogic MIB files, and the receiver will know how to decode each trap.

You can view the MIB files in the **MIB Compiler** page (System > Tools > MIB Compiler).

- ***SCIENCELOGIC-COMMON-MIB***. Defines the root OID for ScienceLogic.com (19567) and the products associated with Skylar One.
- ***SCIENCELOGIC-EVENT-MIB***. Defines varbinds for all the event information that can be included in a trap.
- ***SCIENCELOGIC-TRAP-MIB***. Defines two basic types of traps, severity-based traps, or event-type traps. Both types of traps can contain one, multiple, or all varbinds from the SCIENCELOGIC-EVENT-MIB.

If you choose to use the default ScienceLogic MIB files, you must configure the external system to receive traps from Skylar One. The MIB file SCIENCELOGIC-TRAP-MIB defines two types of event-based traps:

- ***Severity-based traps***. These traps specify that an event of a certain severity has occurred. The trap contains details on the event, including the event message and the element associated with the event.
- ***Event Type-based Traps***. These traps specify the event's policy ID. The trap contains details on the event, including the event message, event severity, and the element associated with the event. This type of trap allows you to define a unique trap OID for each event definition in Skylar One.

You must configure the receiving system to look for the traps.

- If you will send ***event severity-based*** traps:
    - You must configure the receiving system to look for traps with the following OIDs:

Event Severity	OID
Critical event	.1.3.6.1.4.1.19567.2.1.0.0.**1**
Major event	.1.3.6.1.4.1.19567.2.1.0.0.**2**
Minor event	.1.3.6.1.4.1.19567.2.1.0.0.**3**
Notice event	.1.3.6.1.4.1.19567.2.1.0.0.**4**
Healthy event	.1.3.6.1.4.1.19567.2.1.0.0.**5**

    - You must then define your traps (in the **Action Policy Editor** page) using these OIDs. When you specify the ***Trap OID***, use these OIDs.
- If you will send ***event type-based*** traps:
    - You must configure the receiving system to look for traps with the following OIDs:

      .1.3.6.1.4.1.19567.2.1.0.2.1.*event_policy_ID*

    - If you want the receiving system to accept and act on all of these traps, you can tell the receiving system to look for all traps that begin with the OID .1.3.6.1.4.1.19567.1.0.2.1.
    - If you want the receiving system to perform different actions depending upon the type of event, you can use the *event_policy_ID* at the end of each trap OID to sort and separate the traps by type of event.
    - You must then define your traps (in the **Action Policy Editor** page) using the OIDs .1.3.6.1.4.1.19567.2.1.0.2.1.*event_policy_ID*. When specifying the ***Trap OID***, you can use the %3 variable like this:

      .1.3.6.1.4.1.19567.2.1.0.2.1.%3

      Skylar One will append the current event's policy ID to the trap OID. (The current event will be the event that triggered the action policy. This event is specified in the automation policy.)

Varbinds

If you want to use an already defined MIB file and already defined OIDs, you can use the ScienceLogic MIB files SCIENCELOGIC-TRAP-MIB and the SCIENCELOGIC-EVENT-MIB and then dynamically assign values to the OIDs in those files. You can view the MIB files in the **MIB Compiler** page (System > Tools > MIB Compiler).

If you use the ScienceLogic MIB files, specifically the SCIENCELOGIC-EVENT-MIB files, you can include one or more of the following variables (called ***varbinds***) in each outgoing trap. You can assign values to these variables using the event variables described in the appendix on *Variables*.

Description	OID	Type	Associated Event Variable
Event ID	.1.3.6.1.4.1.19567.2.1.1.1.**1**	Integer	%e
Severity of the event, in numeric format. Possible values are 0 = healthy 1 = notice 2 = minor 3 = major 4 = critical	.1.3.6.1.4.1.19567.2.1.1.1.**2**	Integer	%s
Source of the event. Possible values are: syslog=1 internal=2 trap=3 dynamic=4 email=7 other=8	.1.3.6.1.4.1.19567.2.1.1.1.**3**	Integer	%z

Description	OID	Type	Associated Event Variable
Type of element that this event is tied to. Possible values are: organization=0 device=1 asset=2 network=4 interface=5 vendor=6 account=7 virtual interface=8 device group=9 IT service=10 ticket=11	.1.3.6.1.4.1.19567.2.1.1.1.**4**	Integer	%1 (one)
Unique element ID. For example, if the elementType is device, the elementID corresponds to the ScienceLogic device ID.	.1.3.6.1.4.1.19567.2.1.1.1.**5**	Integer	%x
Element name from Skylar One. Examples of element names are device hostname and organization name.	.1.3.6.1.4.1.19567.2.1.1.1.**6**	String	%X
Network address of an element. Typically this is an IP address.	.1.3.6.1.4.1.19567.2.1.1.1.**7**	String	N/A
Unique organization ID.	.1.3.6.1.4.1.19567.2.1.1.1.**8**	Integer	%o (lowercase "oh")
Organization Name	.1.3.6.1.4.1.19567.2.1.1.1.**9**	String	%O (uppercase "oh")
Event description (from event's definition)	.1.3.6.1.4.1.19567.2.1.1.1.**10**	String	%M

Description	OID	Type	Associated Event Variable
Type of sub-element that this event is tied to. Possible values for organizations are: news feed=0 Possible values for devices are: cpu=1 disk=2 filesystem=3 memory=4 swap=5 component=6 interface=7 software=8 process=9 port=10 service=11 content=12 mail=13	.1.3.6.1.4.1.19567.2.1.1.1.**11**	Integer	%2 (two)
Unique sub-element ID. For example, if the subElementType is disk, the subElementID corresponds to the disk ID.	.1.3.6.1.4.1.19567.2.1.1.1.**12**	Integer	%y
Name of sub-element associated with the event	.1.3.6.1.4.1.19567.2.1.1.1.**13**	String	%Y

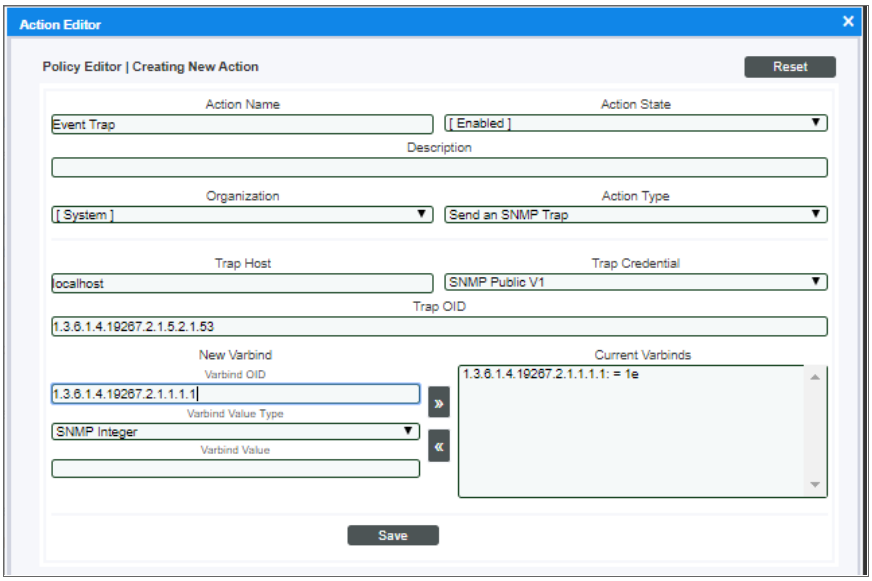
Example Trap

The following is an example of a trap that could be built with an action policy. This trap is event-type based (note the OID):

```
Trap Received: (.1.3.6.1.4.1.19567.2.1.0.2.1.217) | Trap Detail :
eventID: 32755; eventSeverity: 5; eventSource: 2; elementType: 1;
elementID: 119; elementName: webserver01; elementAddress: 192.168.11.30;
roaID: 0; roaName: System; eventMessage: CPU usage now below threshold
(load now: 2%); subElementType: 0; subElementID: 0; subElementName:;
```

Creating the Action Policy

To define an action policy that sends an SNMP trap to an external device, you must supply values in the general fields, as specified in the section on *Creating an Action Policy* and also supply values in the following fields:



- ***Trap Host***. IP address of the external device to which you want to send a trap.
- ***Trap Credential***. SNMP credential that allows Skylar One to send information to the external device. The list of credentials is filtered to include only those credentials to which you have access.

  If this field has already been set to a credential to which you do not have access, this field will display the value *Restricted Credential*. If you set this field to a different credential, the entry for *Restricted Credential* will be removed from the list in this field; you will not be able to re-align the device with the *Restricted Credential*.

NOTE: Your organization membership(s) might affect the list of credentials you can see in the ***Trap Credential*** field. For more information, see the ***Discovery and Credentials*** manual.

- ***Trap OID***. Object identifier for the trap. If you are using the default ScienceLogic MIB files to build traps, see *Default Traps from the ScienceLogic platform* to determine which OID to enter in this field.
- ***Varbind OID***. Object identifier (in dotted decimal notation) of the variable.

- ***Varbind Value Type***. Data type contained in the variable.
- ***Varbind Value***. Value to assign to the variable. You can use the event variables to assign values to the trap variables. This ensures that values from the event specified in the automation policy are included in the trap.
- Supply values in the ***Varbind OID***, ***Varbind Value Type,*** and ***Varbind Value***, then click the right-arrow button (**>>**) to add the varbind to the ***Current Varbinds*** pane. Repeat this step for each variable you want to include in the trap. If you are using the default ScienceLogic MIB files to build traps, see the section on *Varbinds* to determine the ***Varbind OID***, ***Varbind Value Type***, and ***Varbind Value***.
- Each defined variable will appear in the ***Current Varbinds*** pane. To edit a varbind, highlight it in the ***Current Varbinds*** pane and click the left-arrow button (**<<**). The ***Varbind OID***, ***Varbind Value Type***, and ***Varbind Value*** fields will be populated with values from the selected varbind.

NOTE: In the ***Trap OID*** field, ***Varbind OID*** field, and the ***Varbind Value*** field, you can use the variables described in the appendix on *Variables*. The value of each variable will be retrieved from the event that triggered the automation policy.

Creating an Action Policy that Creates a New Ticket

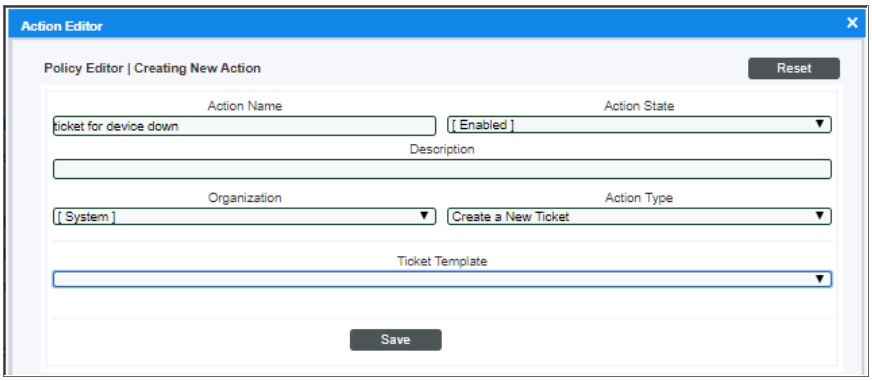
In the **Action Policy Editor** page, if you selected the ***Action Type*** of *Create a New Ticket*, the new action policy will generate a ticket in Skylar One. The value in each ticket field is supplied by a ticket template. Ticket templates are defined in the **Ticket Templates** page (Registry > Ticketing > Templates page.) If the action is associated with an automation policy, and the criteria in the automation policy are met, Skylar One will generate a ticket.

NOTE: For more details on ticket templates, see the chapter on ticket templates in the ***Ticketing*** manual .

An action policy that automatically generates a ticket is useful when you want to immediately assign a task based on event conditions. When certain event conditions are met (as defined in the automation policy), Skylar One can automatically create a ticket that describes the task to be performed and specifies who should perform that task.

To ensure that the generated ticket includes data from the event triggered in the automation policy, you can define a ticket template that uses event variables. These variables are described in the appendix on *Variables* and can be used in the ***Description*** and ***Notes*** fields of the ticket template.

To define an action policy that creates a ticket, you must supply values in the general fields, as specified in the section on *Creating an Action Policy*, and also supply values in the following fields:



- ***Ticket Template***. From this field, you can select from a list of ticket templates. Ticket templates are defined in the **Ticket Templates** page (Registry > Ticketing > Templates). All ticket templates defined with a ***Feature Use*** of *Automation* will appear in this drop-down list. Each of these ticket templates is listed in the ***Ticket Template*** field by ID and name. The ticket template will populate the fields for the ticket that is created by the action policy.

NOTE: For more details on ticket templates, see the chapter on ticket templates in the ***Ticketing*** manual.

Creating an Action Policy that Sends an SNMP Set

The Action Type of *Send an SNMP Set* writes a value to an SNMP variable on an external device. In the action policy, you can specify the variable to write to and the value to write. If the action policy is associated with an automation policy, and the criteria in the automation policy are met, Skylar One will write a value to the variable on the external device.

In the **Action Policy Editor** page, you can specify the SNMP variable to change and the value to assign to the SNMP variable.

For increased flexibility and connectivity, you can specify whether the SNMP Set should be executed by the Database Server or by the Data Collector. In some cases, a device might not accept connections from the Database Server or may not be "visible" from the Database Server. In these situations, you can specify that the SNMP Set be executed by the Data Collector.

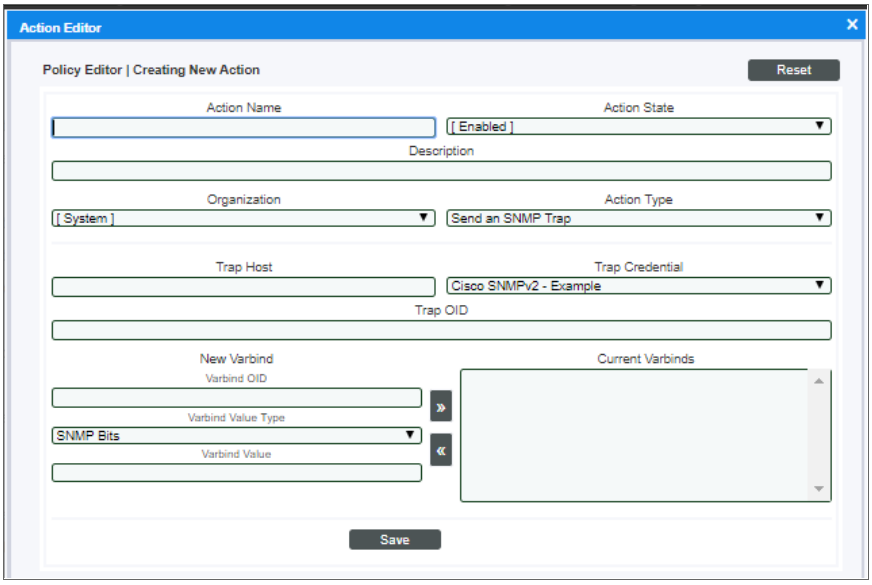
NOTE: For Skylar One systems that are using an All-In-One Appliance, you cannot choose to execute a policy on an Database Server or an Data Collector. All policies will be executed on the All-In-One Appliance.

An action policy that automatically changes the value of an SNMP variable on an external device is useful when you want to perform some automatic steps on the device to resolve a problem. For example, the external device could run a script that is triggered when the value of an SNMP variable is set to "5". You

could also use such an action policy to create a custom status or a custom message and store that custom status or custom message in an SNMP variable.

NOTE: Before you can write a value to an SNMP variable on an external device, you must be aware of the SNMP structure on the external device and the list of SNMP variables on the external device.

To define an action policy that changes an SNMP variable on an external device, you must supply values in the general fields, as specified in the section on *Creating an Action Policy*, and also supply values in the following fields:



- ***SNMP Host***. IP address of the external device where you want to write an SNMP value.
- ***SNMP Credential***. SNMP credential that allows Skylar One to send information to the external device. The list of credentials is filtered to include only those credentials to which you have access.

  If this field has already been set to a credential to which you do not have access, this field will display the value *Restricted Credential*. If you set this field to a different credential, the entry for *Restricted Credential* will be removed from the list in this field; you will not be able to re-align the device with the *Restricted Credential*.

NOTE: Your organization membership(s) might affect the list of credentials you can see in the ***SNMP Credential*** field. For details, see the ***Discovery and Credentials*** manual .

- ***Action Run Context***. This option is not available onAll-In-One Appliances. Specifies whether the action will be executed on the Database Server or on the Data Collector. The Choices are:
    - *Database*. Execute the action from the Database Server.
    - *Collector*. Execute the action from the Data Collector associated with the device. This is useful when a device doesn't accept connections from the Database Server or may not be "visible" from the Database Server.

> **NOTE**: If the triggering event (that is, the event specified in the automation policy that triggered this action policy) is not aligned with a device, and you select *Collector* in the ***Action Run Context*** field, Skylar One will 1) Not execute the action policy; 2) Create a log entry in the audit log for the organization aligned with the triggering event, noting that the criteria in the automation policy were met, but that the action policy was not executed.

- ***SNMP OID***. Object identifier for the variable on the external device to which you want to write a value.
- ***SNMP Value Type***. Data type contained in the variable.
- ***SNMP Value***. Value to assign to the variable.

> **NOTE**: In the ***SNMP Host*** field, the ***SNMP OID*** field, and the ***SNMP Value*** field, you can use one or more of the variables described in the appendix on *Variables*. The value of each variable will be retrieved from the event that triggered the automation policy.

Creating an Action Policy that Executes an SQL Query

In the **Action Policy Editor** page, if you selected the ***Action Type*** of *Execute an SQL Query*, the new action policy will execute an SQL query against an external database on an external device. The SQL query can either retrieve values from an external database or write values to an external database. If the action policy is aligned with an automation policy (i.e., if the action policy is included in the definition of an automation policy), and the criteria in the automation policy are met, Skylar One will execute the query.

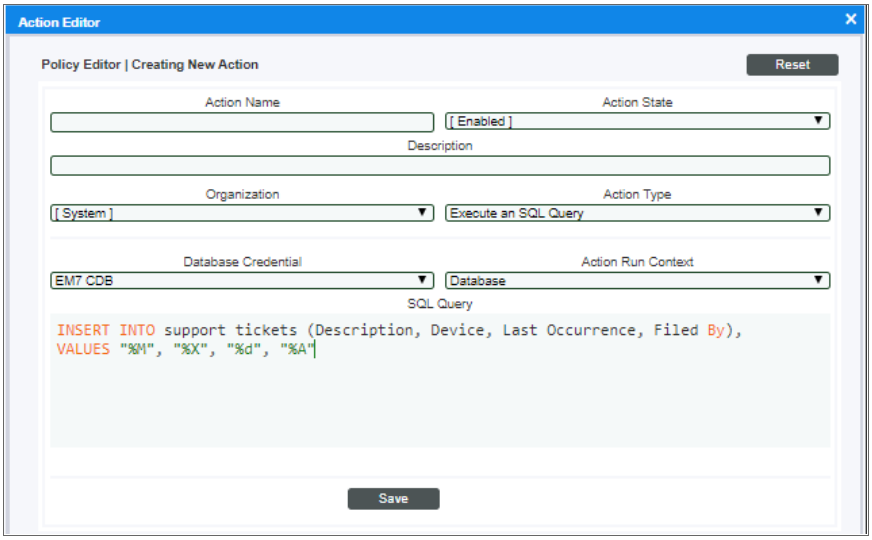
In the **Action Policy Editor** page, you specify the database you want to query and the SQL query to execute.

An action policy that automatically executes an SQL query is useful when you want to integrate event information from Skylar One with an external application that is database-based. For example, suppose you want an event to trigger a ticket on an external ticketing system. Suppose the ticketing system is database-based. If you know the database and table structure on the external ticketing system, you could use an action policy to manually create a ticket in the external database.

For increased flexibility and connectivity, you can specify whether the SQL query should be executed by the Database Server or by the Data Collector. In some cases, a device might not accept connections from the Database Server or may not be "visible" from the Database Server. In these situations, you can specify that the SQL query be executed by the Data Collector.

NOTE: For Skylar One systems that are using an All-In-One Appliance, you cannot choose to execute a policy on a Database Server or a Data Collector. All policies will be executed on the All-In-One Appliance.

To define an action policy that executes an SQL query on an external database, you must supply values in the general fields, as specified in the section on *Creating an Action Policy*, and also supply values in the following fields:



- ***Database Credential***. Credential that allows Skylar One to send a query to the external database. The database to query is specified in the credential. The list of credentials is filtered to include only those credentials to which you have access.

  If this field has already been set to a credential to which you do not have access, this field will display the value *Restricted Credential*. If you set this field to a different credential, the entry for *Restricted Credential* will be removed from the list in this field; you will not be able to re-align the device with the *Restricted Credential*.

NOTE: Your organization membership(s) might affect the list of credentials you can see in the ***Database Credential*** field. For details, see the ***Discovery and Credentials*** manual.

- ***Action Run Context***. This option is not available on All-In-One Appliances. Specifies whether the action will be executed on the Database Server or on the Data Collector. The choices are:

  - *Database*. Execute the action from the Database Server.
  - *Collector*. Execute the action from the Data Collector associated with the device. This is useful when a device doesn't accept connections from the Database Server or may not be "visible" from the Database Server.

- ***SQL Query***. SQL query to execute.

> **NOTE**: In the ***SQL Query*** field, you can use the variables described in the appendix on *Variables*. The value of each variable will be retrieved from the event selected in the automated policy.

> **NOTE**: Skylar One automatically performs an "auto-commit" action for each query, to save the change to the database. You are not required to create a separate "commit" clause for the queries in an action policy.

> **NOTE**: If you clicked the ***Code Highlighting*** in the **Account Preferences** page (Preferences > Account > Preferences), the code in the ***SQL Query*** field appears with syntax highlighting.

Creating an Action Policy that Updates an Existing Ticket

The Action Type of *Update an Existing Ticket* edits an existing ticket in Skylar One. The action can change the status, severity, and/or add a note to an existing ticket. The existing ticket must be associated with the event that triggers the automation policy that executes the action policy. This means that a user manually created the ticket from an instance of an event or that another Run Book Action Policy created the ticket. If the *Update an Existing Ticket* action is associated with an automation policy, and the criteria in the automation policy are met, Skylar One will edit the ticket.

An action policy that automatically edits a ticket is useful when you want to automate tasks in your escalation processes. For example, you could define an automation policy that specifies if an event is still active after a certain time period (that is, the event has not been cleared), increase the severity of the ticket. Conversely, you could define an automation policy that automatically resolves the ticket associated with an event when that event is cleared.

In the **Action Policy Editor** page, if you selected the ***Action Type*** of *Update an Existing Ticket*, you must supply values in the fields, as specified in the section on *Creating an Action Policy*, and also supply values in the following fields:

- ***Set Ticket Status***. Specifies the status to assign to the ticket. Choices are:
  - *Don't Change Status*
  - *Open*
  - *Working*
  - *Pending*
  - *Resolved*
- ***Set Ticket Severity***. Specifies how the severity of the ticket will be modified, or a specific severity to assign to the ticket. Choices are:

- *Don't Change Severity*
- *Increment Severity*
- *Decrement Severity*
- *Healthy*
- *Notice*
- *Minor*
- *Major*
- *Critical*

- ***Add Ticket Note***. Specifies text to add to the ticket as a note, like notes added with the **Notepad Editor**.

NOTE: For details on Ticket Status, Ticket Severity, and adding a note to a ticket, see the chapter on *Creating and Editing Tickets* in the ***Ticketing*** Manual.

Creating an Action Policy that Sends an AWS SNS Message

The Action Type of "Send an AWS SNS message" sends an SNS (Amazon Simple Notification Service) message to a Topic ARN (Amazon Resource Name). All subscribers to the Topic ARN will receive the message.

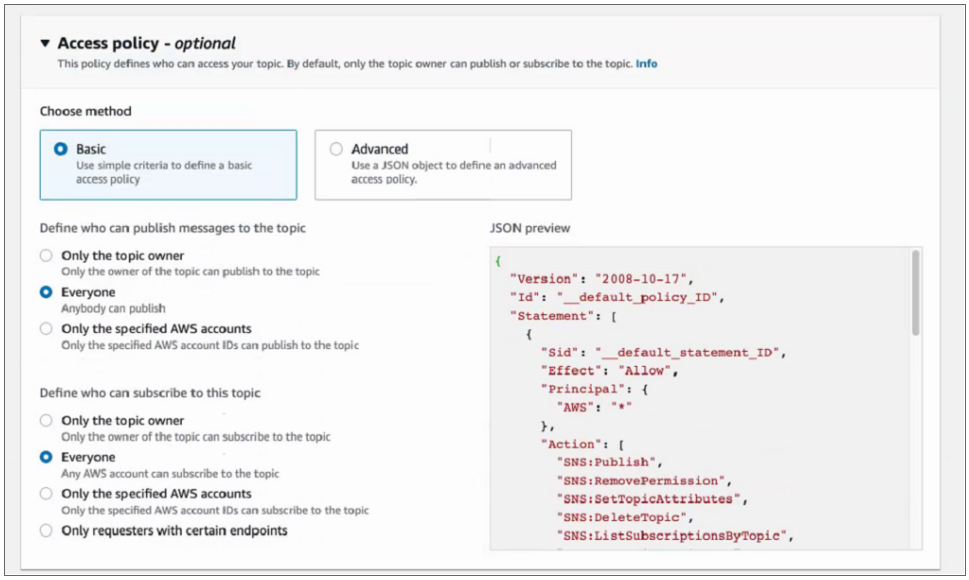
An action policy that sends an AWS SNS message is useful when you already have access to an AWS account through discovery and monitoring and would like to use the AWS ARN publish/subscribe model to send event notifications to users.

Setting up a Topic ARN in AWS

This information is provided as a courtesy to help you configure the action policy successfully. Use the following procedure to prepare an AWS Topic ARN. For details, refer to the Amazon Web Services documentation (https://docs.aws.amazon.com/).

1. From your AWS account, create a new SNS topic (**Amazon SNS > Topics > Create Topic**). Enter a *Topic Name*.

a. Set the access policy to the least restrictive for testing, as shown.



b. Click **Create Topic**. Note the *Topic ARN* ID, as you will need this to set up the message and the action policy.

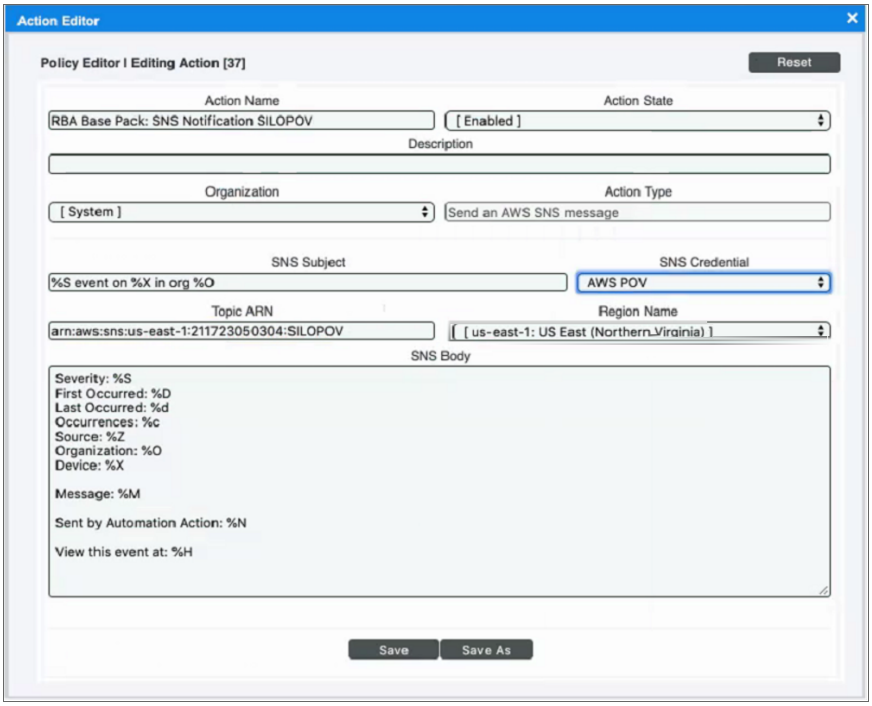
2. From the SNS console, create a subscription to the Topic ARN (**Amazon SNS > Subscriptions > Create Subscription**).
   a. Click **Create subscription**.
   b. Enter the **Topic ARN** ID.
   c. Select "Email" as the *Protocol*.
   d. Enter a valid email address as the *Endpoint*.
   e. Click **Create subscription**.
3. Confirm your subscription in the email confirmation sent from AWS.

Creating the Action Policy for AWS SNS

In the **Action Policy Editor** page, if you selected the ***Action Type*** of *Send an AWS SNS message*, you must supply values in the required fields, as specified in the section on *Creating an Action Policy,* and also supply values in the following AWS SNS fields:

- ***SNS Subject***. This field is optional. This field specifies the subject line for the SNS message. This field cannot exceed 100 characters and cannot contain newline characters or any special characters. You can include variables in this field.
- ***SNS Credential***. Select a credential of type "SOAP/XML" that will allow Skylar One to access the specified ***Topic ARN*** and ***Region***.
- ***Topic ARN***. The Topic ARN to which you want to send the SNS message. All subscribers to the Topic ARN will receive the SNS message.

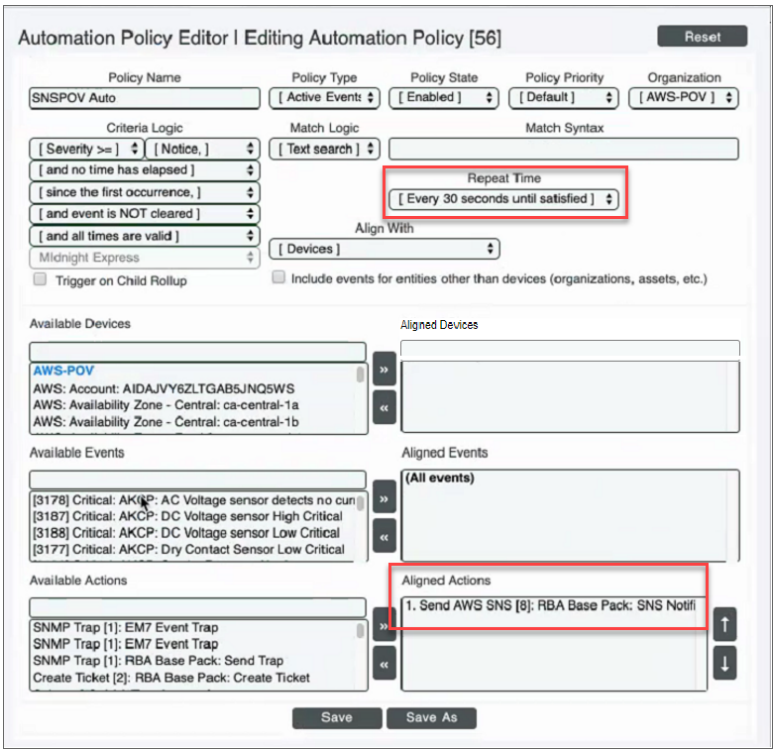
- ***Region Name***. AWS region with which the Topic ARN is associated.
- ***SNS Body***. The body of the SNS message. You can include variables in this field.



Using the Action Policy in an Automation Policy for AWS SNS

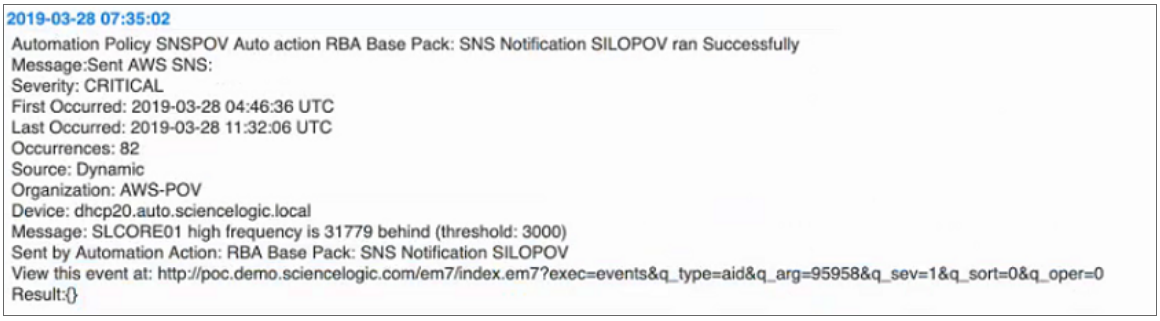
In the **Automation Policy Editor**, create an automation policy that runs the action policy you created for AWS SNS messages.

1. Enter a ***Policy Name*** that describes this automation.
2. Align the action policy you created with the automation policy in the ***Aligned Actions*** field.
3. Set the ***Repeat Time*** to "Every 30 seconds until satisfied" to test the automation.

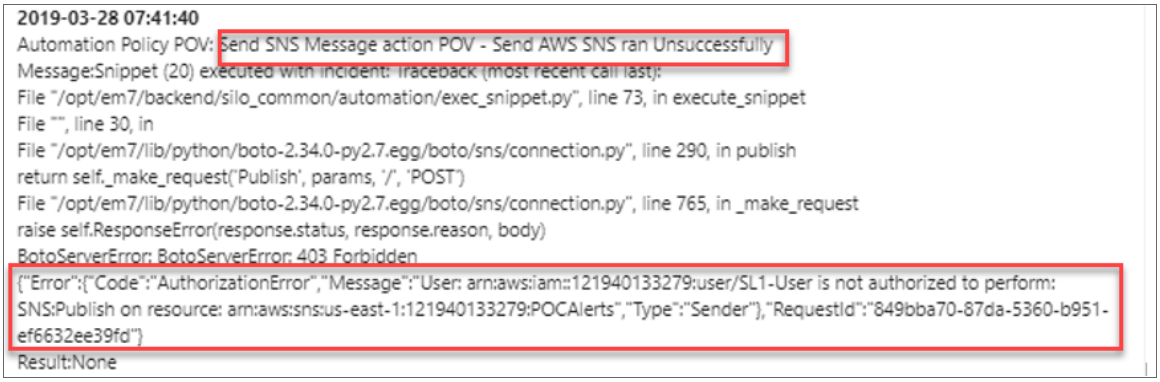


4. Click **[Save]**.
5. Check your email inbox for a message received from AWS SNS. You might also need to check your Spam or Junk mailboxes, if you do not receive the message in your inbox within a few minutes.

6. Check that the event appears in your Events viewer and that the automation policy ran successfully. An example is shown below:



If the automation policy did not run successfully, you might see something like the following in the event. Troubleshoot as necessary.



Creating an Action Policy that Runs a PowerFlow Application

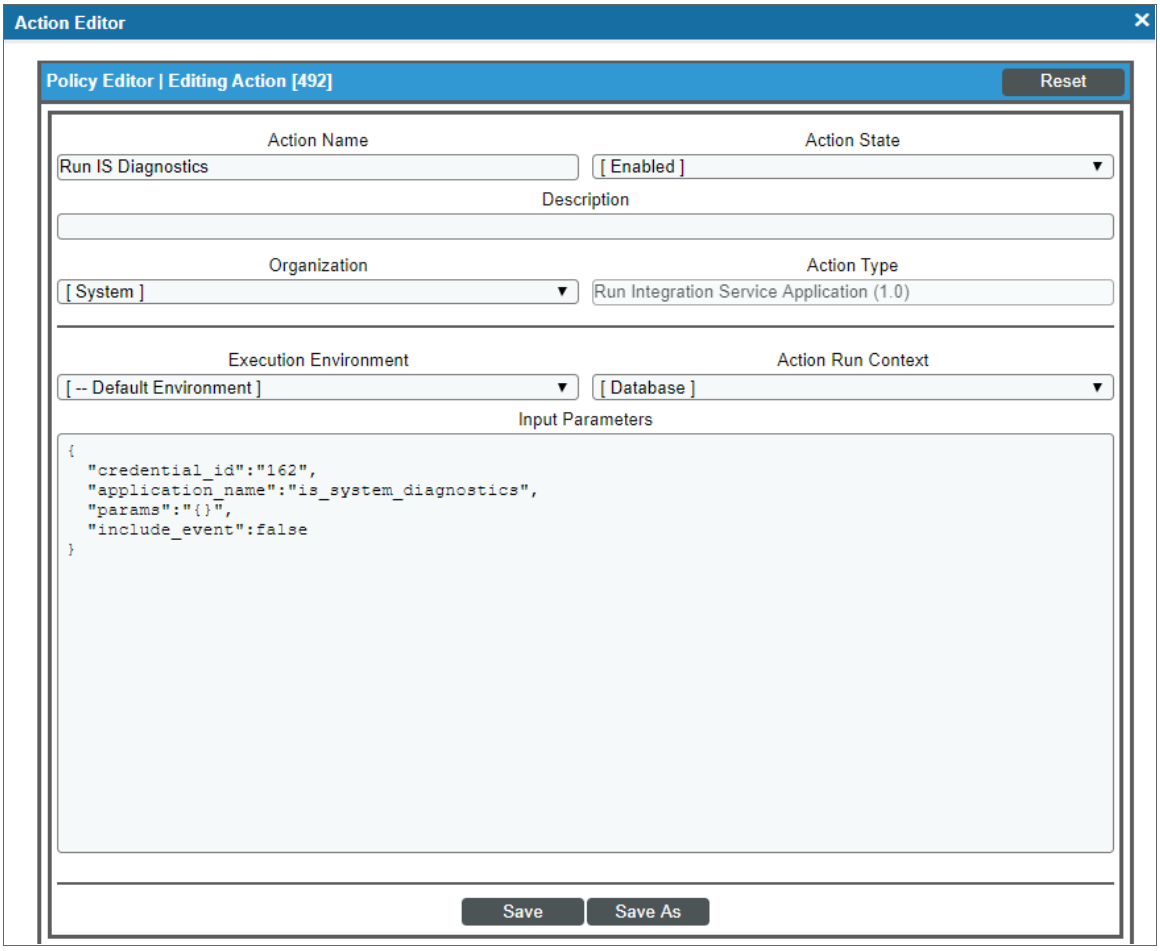
The "PowerFlow Action Type" PowerPack (formerly called the "Integration Service Action Type" PowerPack) includes the "Run Integration Service Application" action type. You can use this action type to build an automation action that triggers an application in Skylar One PowerFlow. When you create an automation action using this action type, you must specify the credential for the PowerFlow instance, the application to run, and the parameters to include in the request.

This automation action does not wait for the PowerFlow application to complete. If the PowerFlow API responds, the output of the automation action is the JSON response from the API, formatted as a string.

NOTE: This action type is not compatible with the formatting actions in the "Datacenter Automation Utilities" PowerPack.

You can create a new automation action that runs aPowerFlow application using the “Run Integration Service Application” action type. To do this, select “Run Integration Service Application” in the ***Action Type*** drop-down list when you create a new automation action.

The following figure shows an example of a custom action that uses the "Run Integration Service Application" action type to run PowerFlow diagnostics:



NOTE: The ***Execution Environment*** drop-down list displays the specific Python version of each execution environment.

This action accepts the following parameters in JSON:

Parameter	Input Type	Description
credential_id	integer	The ID of the credential that can be used to connect to the PowerFlow API.
include_event	boolean	Controls whether the EM7_VALUES dictionary is included as a parameter in the request to run the PowerFlow application.
application_name	string	Contains the name of the application to execute.

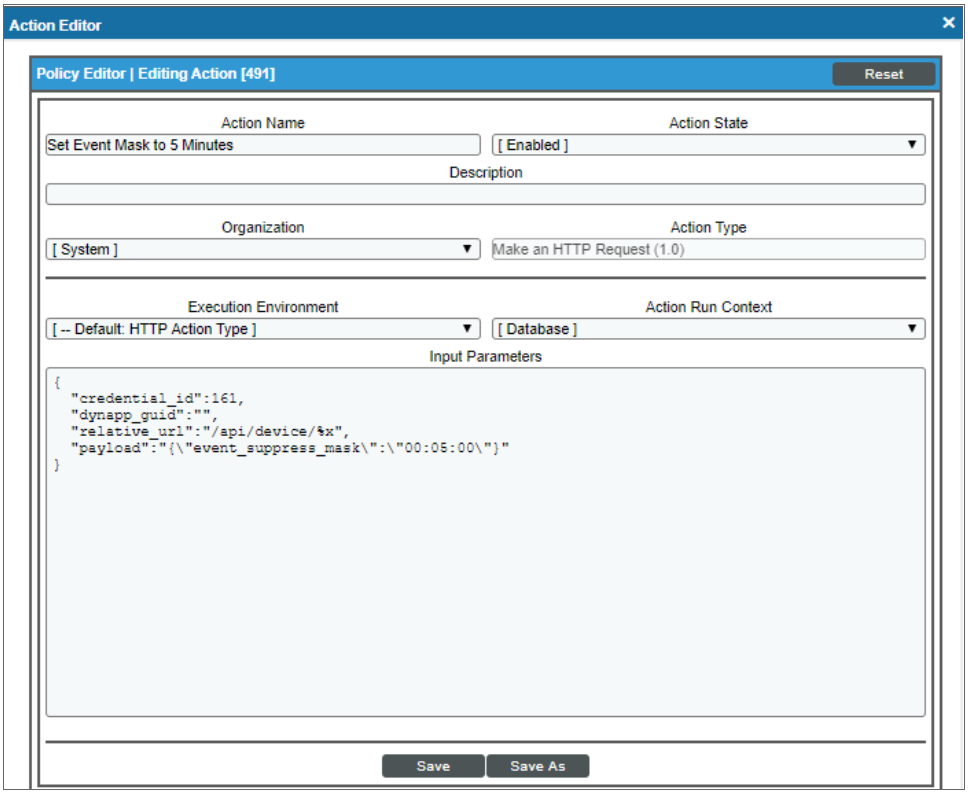
Parameter	Input Type	Description
params	string	A string representation of a JSON object. This string will be passed as additional parameters to the PowerFlow application. This field can contain substitution values that match the keys in EM7_VALUES.

Creating an Action Policy that Sends an HTTP Request

The "HTTP Action Type" PowerPack includes the "Make an HTTP Request" action type. You can use this action type to build an automation action that sends an HTTP request, using a SOAP/XML credential and specific to one or more devices.

You can create a new automation action that sends an HTTP request using the “Make an HTTP Request” action type. To do this, select “Make an HTTP Request” in the Action Type drop-down list when you create a new automation action.

The following figure shows an example of a custom action that uses the "Make an HTTP Request" action type to set the Event Mask to 5 minutes:



NOTE: The ***Execution Environment*** drop-down list displays the specific Python version of each execution environment.

This action accepts the following parameters in JSON:

Parameter	Input Type	Description
credential_id	integer	The ID of the SOAP/XML credential that will be used to make the request.
dynapp_guid	integer	If the credential_id parameter is set to zero, the dynapp_guid parameter is used. The credential that will be used is the one aligned to the specified Dynamic Application (using the GUID to look up the Dynamic Application) on the device associated with the triggering event.
relative_url	string	Specifies the relative portion of the URL. This string will be appended to the URL from the credential.
payload	string	A string representation of a JSON object. This JSON object will be sent as the body of the HTTP request. This field can contain substitution values that match the keys in EM7_VALUES.

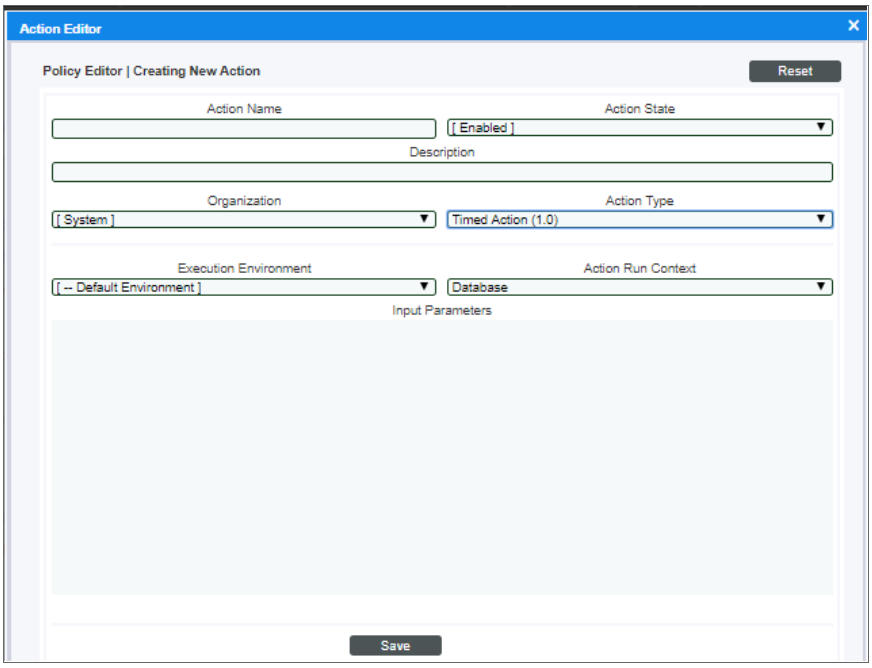
Creating an Action Policy that Uses a Custom Action Type

If your organization has defined Custom Action Types, you can select them when creating an Action Policy.

A ***Custom Action Type*** executes a reusable snippet. Unlike the Action Type "Snippet", a Custom Action Type can accept input parameters (in a JSON format) and create output (in a JSON format). A Custom Action Type allows a single snippet to be used in multiple Action Policies, each time with different inputs and different outputs.

A Custom Action Type is associated with an Execution Environment. An ***execution environment*** is an on-demand Python environment that includes the supporting modules, code, scripts, directories, and files (packaged in one or more ScienceLogic Libraries) required by the Custom Action Type. ***ScienceLogic Libraries*** are packages consisting of metadata and Python files that can be used by the Run Book Actions that use snippets.

In the **Action Policy Editor** page, if you selected a custom action type in the ***Action Type*** field, you must supply values in the fields, as specified in the section on *Creating an Action Policy*, and also supply values in the following fields:



- ***Execution Environment***. Specify the execution environment for the Action Policy. If you select "Default", the Action Policy will use the Execution Environment specified in the definition of the Custom Action Type. This list also shows the Python version of each selectable execution environment.
- ***Action Run Context***. This option is not available on ScienceLogic All-In-One appliances. Specifies whether the action will be executed on the ScienceLogic Database server or on the ScienceLogic Data Collection server. The choices are:
    - *Database*. Execute the action from the ScienceLogic Database Server.
    - *Collector*. Execute the action from the ScienceLogic Data Collection server associated with the device. This is useful when a device doesn't accept connections from the ScienceLogic Database Server or may not be "visible" from the ScienceLogic Database Server.
- ***Input Parameters***. Specifies the name and value of the input parameter, in *name:value* format:

```
{ "name_of_parameter": value}
```

Using the Results of a Previous Action

When you define an action policy, you can use the result from an action that was previously triggered by the same automation policy. To do this, you can use one of the following two variables:

- ***%_EM7_RESULT_%***. Action policies can include the variable %_EM7_RESULT_% to retrieve the results from the previously executed action policy. The value of the variable is available only to the very next action policy in an automation policy. For example, if an automation policy includes three action policies, the results from the first action policy are available only to the second action policy. The third action policy cannot access the results of the first action policy.

> **NOTE:** The EM7_RESULT variable can only pass data types that can be serialized by JSON, plus datetime objects.

- ***em7_result_list***. This variable allows you to include the results from any Action Policy that was executed within the same Automation Policy. For more information on how to use this variable, see *Using the em7_result_list Variable*.

> **NOTE:** You cannot use the ***em7_result_list*** variable with snippet actions. In this situation, use the ***EM7_LAST_RESULT_LIST*** variable for snippet actions. For more information, see *Using the Results of Previous Actions*.

You can use these two variables in the following fields:

- In the subject or body of an email message, sent with an action policy of type *Email Notification*.
- To populate an OID contained in an outbound trap, sent with an action policy of type *Send an SNMP Trap*.
- In the ***Description*** field or in a ***Note*** in a ticket template. The ticket template must be triggered by an action policy of type *Create a New Ticket*.
- To populate an OID contained in an SNMP Set command. The SNMP Set must be triggered by an action policy of type *Send an SNMP Set*.
- As part of an SQL query, triggered by an action policy of type *Execute an SQL Query*.
- In a ticket note added by an action policy of type *Update an existing ticket*.
- In an SNS Message to a Topic ARN (Amazon Resource Name).

Using the em7_result_list Variable

The variable ***em7_result_list*** allows you to include the results from a previous Action Policy in the current Action Policy. The value of the variable is available only to other actions in the same automation policy. For example, if an automation policy includes three action policies, you could include the ***em7_result_list*** variable in the third action policy and retrieve the results from the first action policy and use them in the third action policy. To specify the action policy for which you want to retrieve the results, you include the index number for that action policy. Index numbers start at zero ("0"). The syntax for the ***em7_result_list*** variable is:

{em7_result_list[*i*]}

where *i* represents the index number.

For example:

```
{em7_result_list[2]}
```

would display the results of the third action policy.

For all Action Policies except of type *Execute an SQL Query*, ***em7_result_list*** returns the result of the specified action.

For Action Policies of *Execute an SQL Query*, ***em7_result_list*** returns:

- returned data, if the query was a SELECT query.
- Row Count.
- Last Row ID (if cursor was used in query).
- Messages (if cursor was used in query).

You can include the ***em7_result_list*** variable:

- In the subject or body of an email message, sent with an action policy of type *Email Notification*.
- To populate an OID contained in an outbound trap, sent with an action policy of type *Send an SNMP Trap*.
- In the ***Description*** field or in a ***Note*** in a ticket template. The ticket template must be triggered by an action policy of type *Create a New Ticket*.
- To populate an OID contained in an SNMP Set command. The SNMP Set must be triggered by an action policy of type *Send an SNMP Set*.
- As part of an SQL query, triggered by an action policy of type *Execute an SQL Query*.
- In a ticket note added by an action policy of type *Update an existing ticket*.
- In an SNS Message to a Topic ARN (Amazon Resource Name).

NOTE: You cannot use the ***em7_result_list*** variable with snippet actions. In this situation, use the ***EM7_LAST_RESULT_LIST*** variable for snippet actions. For more information, see *Using the Results of Previous Actions*.

Chapter 4

Snippet Actions

Overview

This chapter describes how to create an action policy that executes a snippet, which is a custom-written Python program.

This chapter covers the following topics:

Creating an Action Policy that Executes a Snippet *78*

Writing Snippet Code *80*

Creating an Action Policy that Executes a Snippet

In the **Action Policy Editor** page, if you selected the ***Action Type*** of *Run a Snippet*, the new action policy will execute a custom-written Python program. If the action policy is aligned with an automation policy (i.e., if the action policy is included in the definition of an automation policy), and the criteria in the automation policy are met, Skylar One will execute the Snippet. Snippet action policies can only include object types that can be serialized and deserialized by JSON data formats.

For increased flexibility and connectivity, you can specify whether the Snippet should be executed by the Database Server or by the Data Collector. In some cases, a device might not accept connections from the Database Server or may not be "visible" from the Database Server. In these situations, you can specify that the Snippet be executed by the Data Collector.

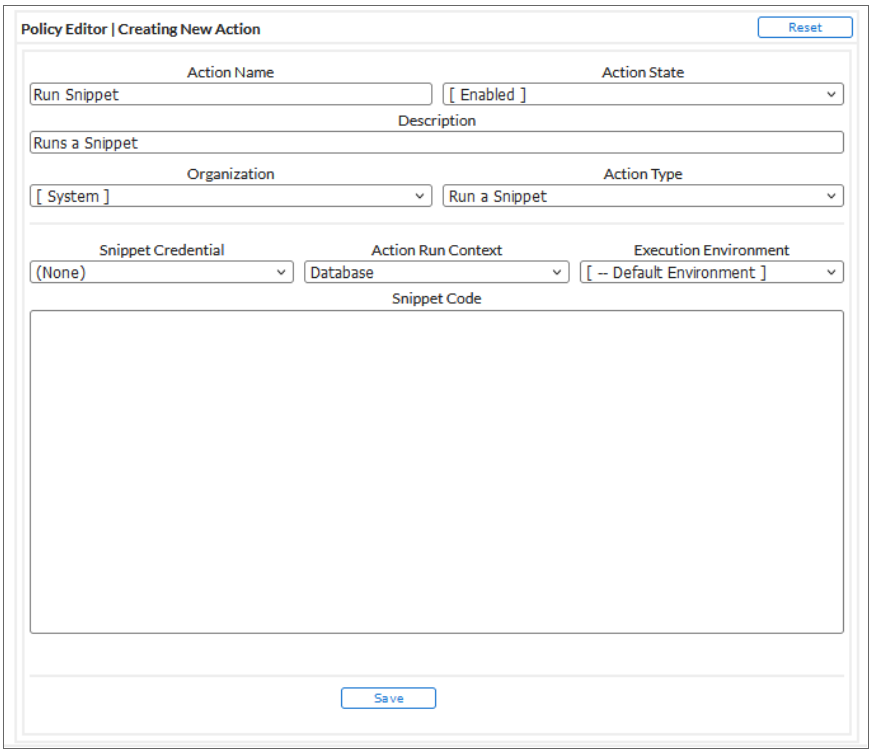
NOTE: For Skylar One systems that are using an All-In-One Appliance, you cannot choose to execute a policy on a Database Server or a Data Collector. All policies will be executed on the All-In-One Appliance.

An action policy that executes a Snippet is useful when you want to run detailed network diagnostics on a device. For example, if Skylar One generates an event saying that a device is not responding to ping, you could run a Snippet that performs a traceroute and specify that Skylar One execute the Snippet from the Data Collector server. You would then execute a traceroute from the Data Collector to the device, store the results in the variable %_ EM7_ RESULT_%, and use that variable to pass the results to another action policy.

An action policy that executes a Snippet is useful when you want to perform some automated steps on the device to resolve a problem. For example, when a specific event is triggered, you could run a Snippet that turns on debugging on the remote device and copies the logs to another remote device.

NOTE: Snippets are developed using the Python programming language. To create a Snippet Action Policy, you must be familiar with the programming techniques and data structures of the Python language.

In the **Action Policy Editor** page, if you select the ***Action Type*** of *Run a Snippet*, you must supply values in the fields specified in *Action Policies* and also in the following fields:



- ***Snippet Credential***. Credential that allows Skylar One to execute the Snippet code on the external device. Usually, these are credentials of type "Basic". The list of credentials is filtered to include only those credentials to which you have access.

  If this field has already been set to a credential to which you do not have access, this field will display the value *Restricted Credential*. If you set this field to a different credential, the entry for *Restricted Credential* will be removed from the list in this field; you will not be able to re-align the device with the *Restricted Credential*.

> **NOTE**: Your organization membership(s) might affect the list of credentials you can see in the ***Snippet Credential*** field.

- ***Action Run Context***. This option is not available on All-In-One Appliances. Specifies whether the action will be executed on the Database Server or on the Data Collector. The Choices are:
  - *Database*. Execute the action from the Database Server.
  - *Collector*. Execute the action from the Data Collector associated with the device. This is useful when a device doesn't accept connections from the Database Server or may not be "visible" from the Database Server.

- ***Execution Environment***. Select the execution environment to which you want to align the action. An execution environment is an on-demand Python environment that contains the supporting modules, code, scripts, directories, and files (packaged in ScienceLogic Libraries) for the snippet. An execution environment includes its own installation directories, doesn't share libraries with other environments, and allows granular control of dependencies, versions, and permissions. The default execution environment is *System*.This list also shows the Python version of each selectable execution environment. For more information, see the ***ScienceLogic Libraries*** manual.
- ***Snippet Code***. Python code for the Snippet.

NOTE: If you selected ***Code Highlighting*** in the **Account Preferences** page (Preferences > Account > Preferences), the code in the ***Snippet Code*** field appears with syntax highlighting.

Writing Snippet Code

The following sections describe the functions and variables that are available to python code for automation actions of type "snippet".

Snippet Functions

Skylar One uses Python's built-in logging using the standard `logging` module. Logger objects that begin with `silo` will automatically route to log files in the Skylar One filesystem. This module includes the following functions that you can use within your Snippet code:

- *import logging*

> **NOTE:** Best practices for logging in Dynamic Applications and run book automation snippets and ScienceLogic Libraries is to use basic pythonic logging under the **silo** namespace and to let the system handle routing and storage. Logging directly to the filesystem and using print statements should be avoided.

To log custom messages from a Dynamic Application or run book snippet:

```
To log to the default log file.
logger = logging.getLogger("silo")
logger.info("This will log to /var/log/sl1/sl1.log")
```

```
To log to a custom log file.
logger = logging.getLogger("silo.custom_logger")
logger.info("This will log to /var/log/sl1/custom_logger")
```

> **NOTE:** ScienceLogic recommends you avoid print statements and logging in Snippets using previously documented methods as both will be deprecated in the future. Logging using any other method may result in lost log records, files, or collected data.

- *em7_snippets.generate_alert*(***message, xid, xtype, yid, ytype, yname, value, threshold***).

  This function allows you to generate an alert from a Snippet action policy. You can define an event based on the alert; the event must have a ***Source*** of *API* and use pattern matching to match the alert. The arguments for the function are:

  - ***message***. Required argument. The message text for the alert.
  - ***xid***. Required argument. The entity to associate with the alert. Supply the numeric ID of an entity. For example, if you supply '1' in the ***xtype*** argument, supply a device ID in this argument.
  - ***xtype***. Specifies the type of ScienceLogic element associated with the alert. Supply one of the following integer values:
    - *0*. Organization
    - *1*. Device
    - *2*. Asset
    - *4*. Network
    - *5*. Interface
    - *6*. Vendor

- *7*. User Account
- *8*. Virtual Interface
- *9*. Device Group
- *10*. IT Service
- *11*. Ticket
- *12*. Quality of Service Object
- *13*. Discovery Session
- *14*. Business Service
- *15*. Business Service - IT Service
- *16*. Device Service
- *20*. Aggregate Service

◦ ***yid*** = *value*. The sub-entity to associate with the alert. Supply the numeric ID of a sub-entity. For example, if you supply '3' in the ***ytype*** argument, supply a file system ID in this argument.

◦ ***ytype*** = *value*. Optional argument. The type of sub-entity for which you specified an ID in the ***yid*** argument. Supply one of the following integer values:

- *9*. News Feed (if ***xtype*** is 0) or Process (if ***xtype*** is 1).
- *1*. CPU. Can be specified only if ***xtype*** is 1 (Device).
- *2*. Disk. Can be specified only if ***xtype*** is 1 (Device).
- *3*. File System. Can be specified only if ***xtype*** is 1 (Device).
- *4*. Memory. Can be specified only if ***xtype*** is 1 (Device).
- *5*. Swap. Can be specified only if ***xtype*** is 1 (Device).
- *6*. Hardware Component. Can be specified only if ***xtype*** is 1 (Device).
- *7*. Interface. Can be specified only if ***xtype*** is 1 (Device).
- *10*. Port. Can be specified only if ***xtype*** is 1 (Device).
- *11*. Windows Service. Can be specified only if ***xtype*** is 1 (Device).
- *12*. Web Content. Can be specified only if ***xtype*** is 1 (Device).
- *13*. Email Monitor. Can be specified only if ***xtype*** is 1 (Device).
- *14*. DNS. Can be specified only if ***xtype*** is 1 (Device).
- *15*. RSS. Can be specified only if ***xtype*** is 1 (Device).
- *16*. Quality of Service. Can be specified only if ***xtype*** is 1 (Device).

- ***yname*** = *value*. Optional argument. The name of the sub-entity for which you specified an ID in the ***yid*** argument.
- ***value***= *string*. Optional argument. A value that will be passed with the alert message. This value is available in the %V substitution character for event policies.
- ***threshold***= *string*. Optional argument. A threshold value that will be passed with the alert message. This threshold value is available in the %T substitution character for event policies.

For example:

```
em7_snippets.generate_alert('Attempted File System Cleanup',
'60', '1', '150', '3')
```

will generate an alert with the message "Attempted File System Cleanup" associated with the file system with ID 150 on the device with ID 60.

Snippet Variables

A Snippet can use the following global Snippet variables:

Global Snippet variables returned by your code:

- ***EM7_RESULT (str)***. An unstructured string variable returned by your code which is presented in the **Event Console** under the given event as *View Event Notification*. If this value is returned as a simple string, it will be coerced into a *dict entry* and passed to the next snippet in `EM7_LAST_RESULT`.
- ***EM7_RESULT (dict)***. A structured dict variable returned by your code which is presented in the **Event Console** under the given event as *View Event Notification*. The *dict* is used to store the results from the current Snippet Action Policy. This variable is used to populate the variable `%_EM7_RESULT_%`.

NOTE: The EM7_RESULT variable can only pass data types that can be serialized by JSON, plus datetime objects.

Provided global Snippet variables:

- ***EM7_LAST_RESULT: dict***. Provided at runtime, this variable contains the results from the previous action policy. The output of this variable is the same as the output of `EM7_LAST_RESULT_LIST` except it outputs a single object from the list.
- A Snippet can access the standard replacement variables by using the global dictionary ***EM7_VALUES***. See *Run Book Variables* for more information. The syntax is:

```
EM7_VALUES['variable']
```

For example, to access the variable that contains a device's IP address:

```
EM7_VALUES['%a']
```

- ***EM7_ACTION_CRED***. Variable that contains an element from the *dictionary of values* for the credential for this action policy, specified in the ***Snippet Credential*** field.
    - `host = EM7_ACTION_CRED['cred_host']`
    - `user = EM7_ACTION_CRED['cred_user']`
    - `password = EM7_ACTION_CRED['cred_pwd']`
- ***EM_7_DEVICE_CRED***. Variable that contains an element from the *dictionary of values* for the credential used to discover the device where the event occurred (that is, the event specified in the automation policy that triggered the current action policy). If the triggering event is not aligned with a device, this variable does not contain a value. These credentials are in the same format as found in a Dynamic Application. For example, a Basic/Snippet credential may look like this:

```
{'cred_host': '10.100.100.100',
                    'cred_id': 52,
                    'cred_name': 'My Aligned App Credentail',
                    'cred_port': 443,
                    'cred_pwd': 'my_example_password',
                    'cred_timeout': 1500,
                    'cred_type': 5,
                    'cred_user': 'my_example_user',
                    'sub_type': ''}
```

The following code will return all the variables in this section to a file in /tmp which you can use to verify fields, values, etc.

IMPORTANT: The credentials details saved will include sensitive or secret information, so ScienceLogic recommends using a sample credential of the same type to verify field, values, etc.

```
from pprint import pformat
filename = "/tmp/rba_dump"
output_dict = {
        "EM7_VALUES": EM7_VALUES,
        "EM7_ACTION_CRED": EM7_ACTION_CRED,
```

```
        "EM7_DEVICE_CRED": EM7_DEVICE_CRED,
        "EM7_DYNAMIC_APP_CREDS": EM7_DYNAMIC_APP_CREDS,
        "EM7_LAST_RESULT": EM7_LAST_RESULT

}


with open(filename, 'w') as f:

    f.write(pformat(output_dict))
```

- ***EM7_DYNAMIC_APP_CREDS['Dynamic_Application'_ID]***. Variable that returns an element from the *dictionary of values* for the credential associated with the specified Dynamic Application on the device (where the triggering event occurred). The syntax is:

```
EM7_DYNAMIC_APP_CREDS[DynamicAppId']
```

NOTE: All Dynamic Applications aligned to the device will be available in this dictionary. If you know the app ID of an aligned device, you can use this dictionary to get the credential dictionary of the credential used in the alignment.

For example, to access the dictionary of values for the credential assigned to the Dynamic Application with the ID of "17", you would enter:

```
EM7_DYNAMIC_APP_CREDS[DynamicAppId]
```

This credential is a dictionary in the same format as `em7_action_cred` and `em7_device_cred`. For example, a Basic/Snippet credential may look like:

```
{'cred_host': '10.100.100.100',
                'cred_id': 52,
                'cred_name': 'My Aligned App Credentail',
                'cred_port': 443,
                'cred_pwd': 'my_example_password',
                'cred_timeout': 1500,
                'cred_type': 5,
                'cred_user': 'my_example_user',
                'sub_type': ''}
```

This would return the dictionary of values for the credential that allows the Dynamic Application with an ID of "17" to run for the device where the triggering event occurred.

A returned value can be used in a run book automation snippet to set a value. For example, the following statement will set the value in the key "cred_user" to variable "user" within the snippet:

```
user = EM7_DYNAMIC_APP_CREDS[DynamicAppId]
```

Credential Dictionary Structure

Several elements in the credential dictionary are common to all credential types, and each credential type (other than Basic/Snippet) has unique elements that appear only in the credential dictionary for that credential type. The following elements are common to every type of credential dictionary:

- ***cred_id***. Integer. Unique credential ID.
- ***cred_type***. Integer. Type of credential .
    - 1 SNMP. *SNMP credentials allow Skylar One to access SNMP data on a managed device.*
    - 2 DB. *Database credentials allow Skylar One to access data on a database on a managed device.*
    - 3 SOAP/XML. *SOAP/XML credentials allow Skylar One to access a web server on a managed device.*
    - 4 LDAP. *LDAP or Active Directory credentials allow Skylar One to access data on an LDAP server or Active Directory server.*
    - 5 Snippet. *Basic/Snippet credentials define standard authentication parameters, but are not tied to a specific authentication protocol.*

    - 6 SSH. *SSH/Key credentials specifies the hostname or IP address of the system you want to monitor, the port number used to access that system, and the private key used for authentication.*
    - 7 PowerShell. *PowerShell credentials allow Dynamic Applications to retrieve data from Microsoft devices.*

> **NOTE**: For more information, see the ***Credential Management*** chapter in the ***Discovery and Credentials*** manual.

- ***cred_host***. String. Host name or IP address (%D substitution string).
- ***cred_port***. Integer. TCP/IP port for connections.
- ***cred_pwd***. String. Password (encrypted in the database, stored as clear text in the dictionary).
- ***cred_user***. String. Username.
- ***cred_timeout***. Integer. Timeout in milliseconds.

The following elements are unique for SNMP credentials:

- ***snmp_version***. Integer. SNMP version, values 1, 2, 3.
- ***snmp_ro_community***. String. Read-only community string.
- ***snmp_rw_community***. String. Read/Write community string.
- ***snmp_retries***. Integer. Number of retries.
- ***snmpv3_auth_proto***. String. V3 auth. protocol,. Can be either MD5 or SHA.
- ***snmpv3_sec_level***. String. V3 security. Can be noAuthNoPriv, AuthNoPriv, or AuthPriv.
- ***snmpv3_priv_proto***. String. V3 privacy protocol. Can be : DES or AES.
- ***snmpv3_priv_pwd***. String. V3 password encrypted in the database and stored as clear text in the dictionary.
- ***snmpv3_context***. String. V3 context.

The following elements are unique for Database credentials:

- ***db_type***. Integer.
    - 1 MySQL
    - 2 MSSQL
    - 3 Oracle
    - 4 Postgress
    - 5 DB2
    - 6 Sybase
    - 7 Informix
    - 8 Ingress).

- ***db_name***. String. Initial database name.
- ***db_sid***. String. Database SID (Oracle only).
- ***db_connect***. String. Database connect string (Oracle only).

The following elements are unique for SOAP/XML credentials:

- ***curl_url***. String. URL.
- ***curl_proxy_ip***. String. Proxy server IP address.
- ***curl_proxy_port***. Integer. Proxy server TCP/IP port.
- ***curl_proxy_acct***. String. Proxy server account.
- ***curl_proxy_passwd***. String. Proxy server password.
- ***curl_encoding***. String. Encoding method (eg text/xml).
- ***curl_post_or_get***. Integer. HTTP method 0 – GET, 1- POST.
- ***curl_http_version***. HTTP version: 10 = 1.0, 11 = 1.1.
- ***curl_request_sub_1***. String. Substitution value to substitute into Snippet code.
- ***curl_request_sub_2***. String. Substitution value to substitute into Snippet code.
- ***curl_request_sub_3***. String. Substitution value to substitute into Snippet code.
- ***curl_request_sub_4***. String. Substitution value to substitute into Snippet code.
- ***curl_headers***. List of Strings. Each string is a HTTP key/value pair.
- ***curl_opts***. Dictionary of Curl options comprising a series of pairs of string key and corresponding string value.

Using the Results of Previous Actions

EM7_LAST_RESULT_LIST Variable

The variable ***EM7_LAST_RESULT_LIST*** allows you to use the results from a previous Action Policy in the current Action Policy. The results of an action are available only to other actions within the same automation policy. For example, if an automation policy includes three action policies, you could pass the results from the first action policy to the third action policy. To specify the action policy for which you want to retrieve the results, you include the index number for that action policy. Index numbers start at zero ("0").

Each index in the ***EM7_LAST_RESULT_LIST*** variable is a list with the following structure:

```
('success', 'type', 'result', 'metrics', 'message')
```

Where:

- ***success***. Contains "True" if the specified Action Policy was successful and "False" if the specified Action Policy was not successful. To assign this value to a local variable, the syntax is:

  ```
  success = EM7_LAST_RESULT_LIST[i].success
  ```

  where *success* is the variable in which to store the returned value and *i* is the index number for the Action Policy, for example "1" for the second Action Policy.

- ***type***. Numeric ID for the action type. Possible values are:
  - 0. Send An Email Notification
  - 1. Send an SNMP Trap
  - 2. Create a New Ticket
  - 3. Send an SNMP Set
  - 5. Run a Snippet
  - 6. Execute an SQL Query
  - 7. Update an Existing Ticket

  To assign this value to a local variable, the syntax is:

  ```
  type = EM7_LAST_RESULT_LIST[i].type
  ```

  where *type* is the variable in which to store the returned value and *i* is index number for the Action Policy, for example "1" for the second Action Policy.

- ***result***. Returns the result of the specified Action Policy and is usually a Python ***dict*** object. To assign this value to a local variable, the syntax is:

  ```
  result = EM7_LAST_RESULT_LIST[i].result
  ```

  where *result* is the variable in which to store the returned value and *i* is the index number for the Action Policy, for example "1" for the second Action Policy.

- ***metrics***. Returns metrics about the specified Action Policy.
  - If the specified Action Policy is not of type "Run a Snippet", this value will be NONE.
  - If the specified Action Policy is of type "Run a Snippet", this value contains the following list structure:

    ```
    ('start_time', 'end_time', 'duration', 'mem', cpu_sys', 'cpu_user')
    ```

  To assign this value to a local variable, the syntax is:

  ```
  metrics = EM7_LAST_RESULT_LIST[i].metrics.end_time
  ```

  where *metrics* is the variable in which to store the returned value and *i* is the index number for the Action Policy, for example "1" for the second Action Policy.

  This syntax returns the "end_time" metric. To view another metric, substitute its name for "end_time". The name of each metric is listed above, in the description of the data structure.

- ***message***. An informational message. If the success parameter returns False, this parameter returns the error message. To assign this value to a local variable, the syntax is:

  ```
  message = EM7_LAST_RESULT_LIST[i].message
  ```

where *message* is the variable in which to store the returned value and *i* is the index number for the Action Policy, for example "1" for the second Action Policy.

For example, suppose we included the following Snippet code in an action of type "Run a Snippet". Suppose our current Action (the one that includes the code) is the fourth action in the Automation Policy. Suppose we want to gather information about the third action (which has an index of "2"). Suppose the third action created a new ticket. Suppose the snippet included the following local variable assignment statements:

```
success = EM7_LAST_RESULT_LIST[2].success

type = EM7_LAST_RESULT_LIST[2].type

result = EM7_LAST_RESULT_LIST[2].result

metrics = EM7_LAST_RESULT_LIST[2].metrics

message = EM7_LAST_RESULT_LIST[2].message
```

The contents of the local variables might be:

```
success: true

type: 2

result: {'tid': 814}

metrics: metrics is None

message: Created ticket 814
```

EM7_LAST_RESULT Variable

The ***EM7_LAST_RESULT*** variable allows you to use the last object returned from the previous Action Policy in the current Action Policy. With this variable, the results of the last action are available only to the next action. The objects available are the same as with EM7_LAST_RESULT_LIST but are presented as a single object and not as a list. EM7_LAST_RESULT returns the following:

```
('success', 'type', 'result', 'metrics', 'message')
```

Where:

- ***success***. Contains "True" if the specified Action Policy was successful and "False" if the specified Action Policy was not successful. To assign this value to a local variable, the syntax is as follows:

  ```
  success = EM7_LAST_RESULT.success
  ```

  where *success* is the variable in which to store the returned value

- ***type***. Numeric ID for the action type. Possible values are as follows:

- 0. Send An Email Notification
- 1. Send an SNMP Trap
- 2. Create a New Ticket
- 3. Send an SNMP Set
- 5. Run a Snippet
- 6. Execute an SQL Query
- 7. Update an Existing Ticket

To assign this value to a local variable, the syntax is as follows:

```
type = EM7_LAST_RESULT.type
```

where *type* is the variable in which to store the returned value.

- ***result***. Returns the result of the specified Action Policy and is usually a Python ***dict*** object. To assign this value to a local variable, the syntax is as follows:

```
result = EM7_LAST_RESULT.result
```

where *result* is the variable in which to store the returned value.

- ***metrics***. Returns metrics about the specified Action Policy.
  - If the specified Action Policy is not of type "Run a Snippet", this value will be NONE.
  - If the specified Action Policy is of type "Run a Snippet", this value contains the following list structure:

```
('start_time', 'end_time', 'duration', 'mem', cpu_sys', 'cpu_user')
```

To assign this value to a local variable, the syntax is as follows:

```
metrics = EM7_LAST_RESULT.metrics.end_time
```

where *metrics* is the variable in which to store the returned value.

This syntax returns the "end_time" metric. To view another metric, substitute its name for "end_time". The name of each metric is listed above, in the description of the data structure.

- ***message***. An informational message. If the success parameter returns False, this parameter returns the error message. To assign this value to a local variable, the syntax is as follows:

```
message = EM7_LAST_RESULT.message
```

where *message* is the variable in which to store the returned value.

For example, suppose we included the following Snippet code in our last action, which is of type "Run a Snippet". Suppose the snippet included the following local variable assignment statements:

```
success = EM7_LAST_RESULT.success

type = EM7_LAST_RESULT.type

result = EM7_LAST_RESULT.result

metrics = EM7_LAST_RESULT.metrics

message = EM7_LAST_RESULT.message
```

The contents of the local variables might be:

```
success: true

type: 2

result: {'tid': 814}

metrics: metrics is None

message: Created ticket 814
```

Chapter 5

Examples

Overview

This chapter contains examples of Run Book Automations that you can use as templates for your own automations.

This chapter covers the following topics:

Action Policy that Sends an Email Message 94

Action Policy that Sends an SNMP Trap to an External Server 96

Action Policy that Creates a Ticket 99

Action Policy that Writes an SNMP Value to an External Server 104

Action Policy that Sends an SQL Query to an External Server 104

Action Policy that Executes a Snippet and Triggers a New Alert 107

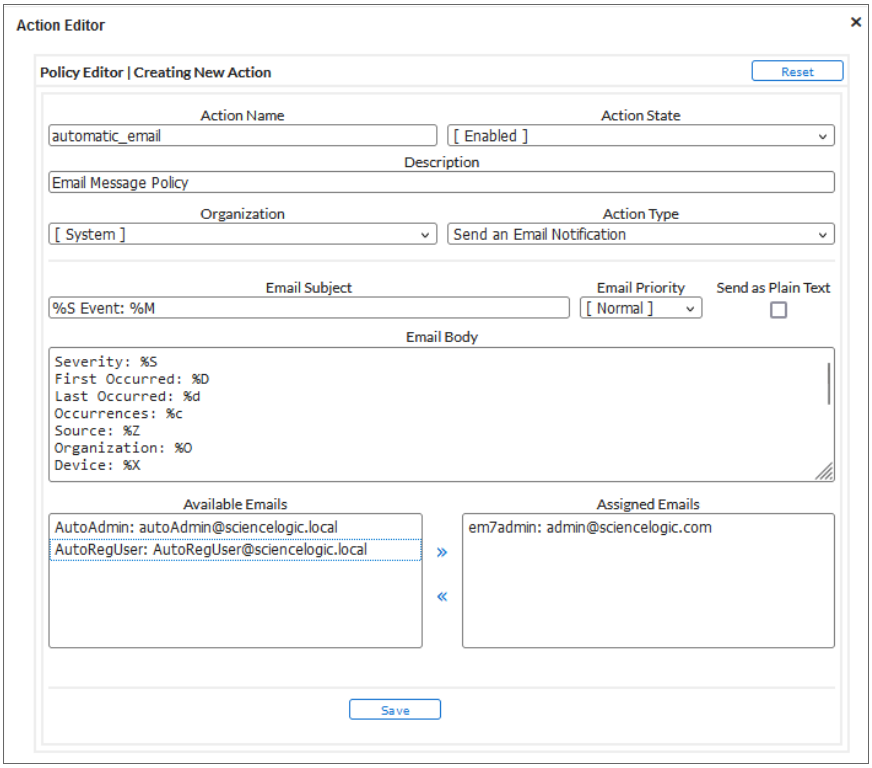
Action Policy that Executes a Snippet and Sends the Results to a Second Action Policy 110

Action Policy that Sends an Email Message

Action Policy

In this example, you will need to first create the action policy for sending the email.

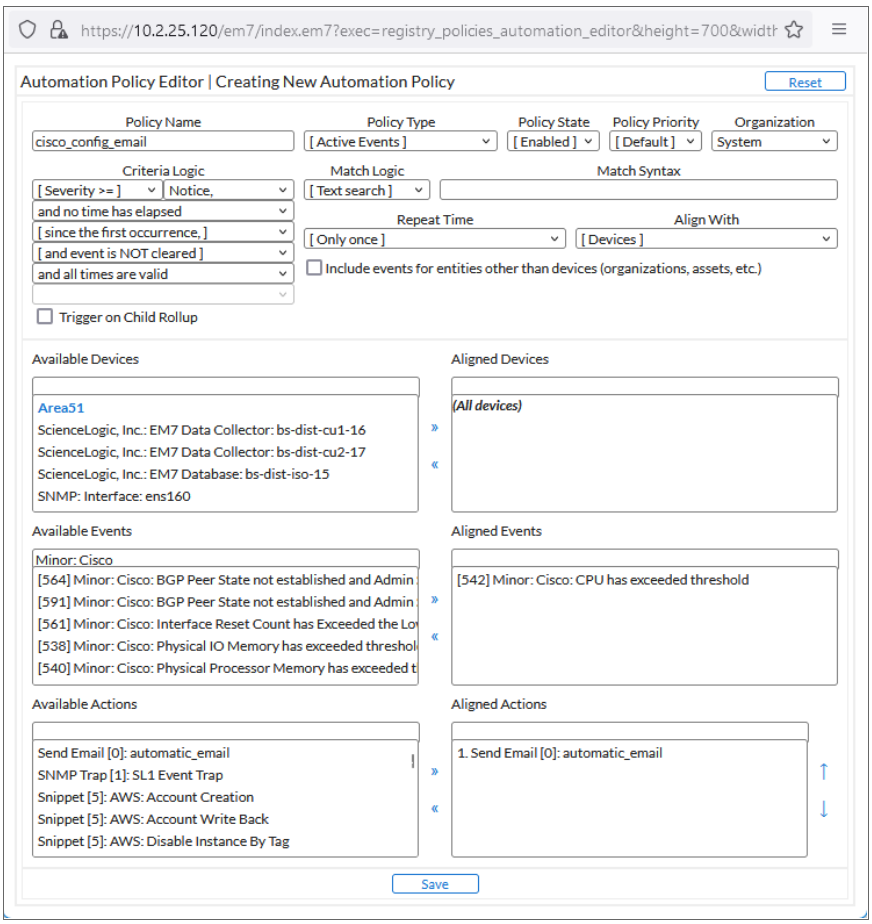
The example action policy called "automatic_email" looks like this:



- We specified that this action policy:
    - Is enabled.
    - Will act upon events and devices aligned with the System organization.
    - Will send an email notification in response to an automation policy.
    - Will include the default Email Subject and Email Body.
    - Will label email messages with Normal priority.
    - Will send an email message to admin@sciencelogic.com.

Automation Policy

For this example, our example automation policy might look like this:



- We specified that the automation policy:
    - Should act upon active events.
    - Is enabled.
    - Is associated with the organization "System".
    - Will be triggered when the specified event has a severity equal to or greater than "Notice".
    - Will be triggered as soon as the specified event occurs.
    - The policy will trigger the action only once for each instance of the event.
    - Will be triggered at any time.
    - Will be triggered when the selected event occurs on at least one of the selected Cisco devices.

    - Will be triggered when the event "Cisco: CPU has exceeded threshold" occurs on at least one of the selected Cisco devices.
- We specified that when all the criteria in the automation policy are met, the action policy "automatic_email" will be executed.

Sent Email

Suppose the criteria in our automation policy "cisco_config_email" was met and that the trigger event "Cisco: CPU has exceeded threshold" occurred on the device "CustB_2821-1.cisco.com".

Suppose our action policy "automatic_email" was successfully triggered and executed.

Our action policy will build and send an email message like this:

```
From: EM7 Event Notifier

Date: Wednesday, January 20, 2010 8:13 AM

Subject: MINOR Event: Configuration management trap received

Date: Wed, 20 Jan 2010 13:12:07 +0000

System Event [16285]

Severity: MINOR

Device/Context: CustB_2821-1.cisco.com

Message: CPU has exceeded threshold

First Occurred: 2010-01-15 22:13:13

Last Occurred: 2010-01-20 13:08:20

Impacted:

Cause and Resolution:

View this event at:
```

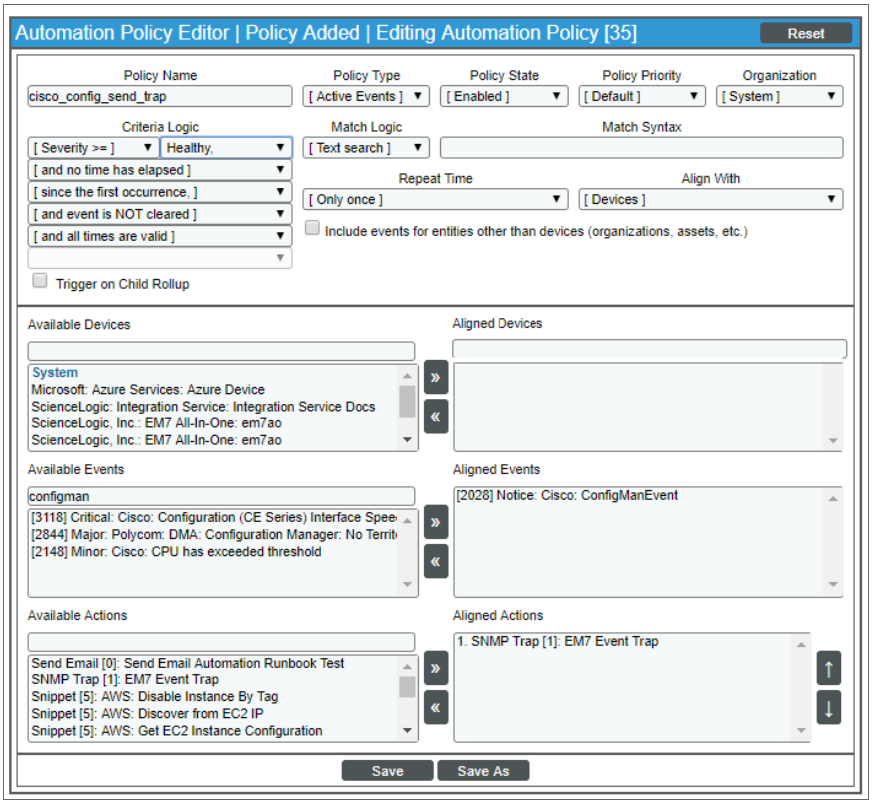
Action Policy that Sends an SNMP Trap to an External Server

Suppose Skylar One must integrate with an existing network management system. To do this, Skylar One must forward certain event information to the existing network management server. Skylar One could use

an SNMP trap to forward event information to another network management server. In this example, we'll use this scenario and send information about each instance of the event "Cisco: CPU has exceeded threshold".

Automation Policy

In this example, we'll use a modified version of the Automation Policy we described in *Automation Policies*.



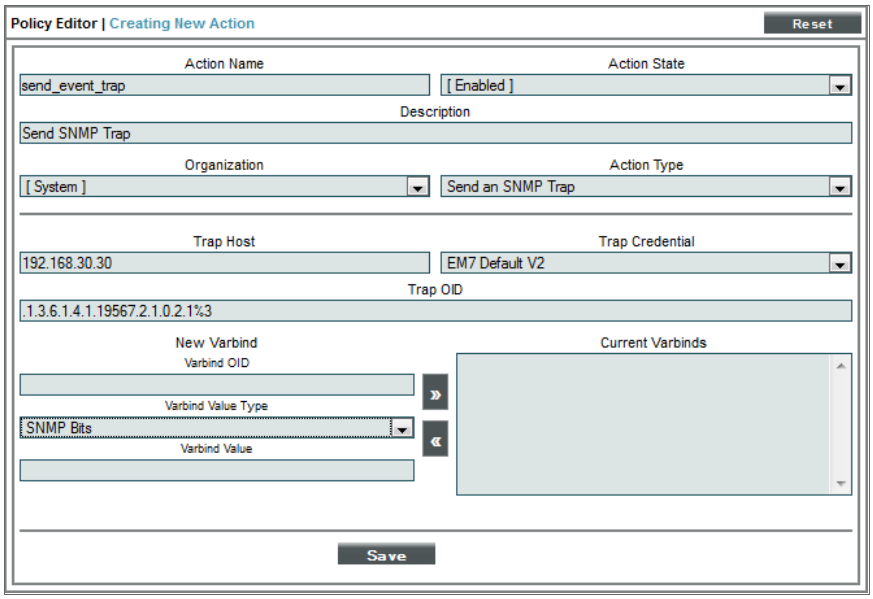
- We specified that the automation policy:
    - Should act upon active events.
    - Is enabled.
    - Is associated with the organization "System".
    - Will be triggered when the specified event has a severity greater than "Healthy".
    - Will be triggered as soon as the specified event occurs.
    - The policy will trigger the action only once.
    - Will be triggered at any time.
    - Will be triggered when the selected event occurs on at least one of the selected Cisco devices.

- Will be triggered when the event "Cisco: ConfigManEvent" occurs on at least one of the selected Cisco devices.

- We specified that when all the criteria in the automation policy are met, the action policy "send_event_trap" will be executed.

Action Policy

The action policy called "send_event_trap" looks like this:



- We specified that this action policy:

  - Is enabled.
  - Will act upon events and devices aligned with the System organization.
  - Will send an SNMP trap in response to an automation policy.
  - Will send the trap to the trap host at 192.168.30.30.
  - Will use the credential "EM7 Default V2" to send the trap to the trap host at 192.168.30.30.
  - Will send an *event type-based trap*, using the OID .1.3.6.1.4.1.19567.2.1.0.2.1.*event_policy_ID*. We use the variable %3, so that EM7 will append the current event's policy ID to the trap OID. (The current event will be the event that triggered the action policy. This event is specified in the automation policy.)
  - Includes all the *EM7 varbinds* in the trap.

Sent Trap

Suppose the criteria in our automation policy "cisco_config_send_trap" was met and that the trigger event "Cisco: ConfigManEvent" occurred on the device "CustB_2821-1.cisco.com".

Suppose our action policy "send_event_trap" was successfully triggered and executed.

Our action policy will build and send an event trap like this:

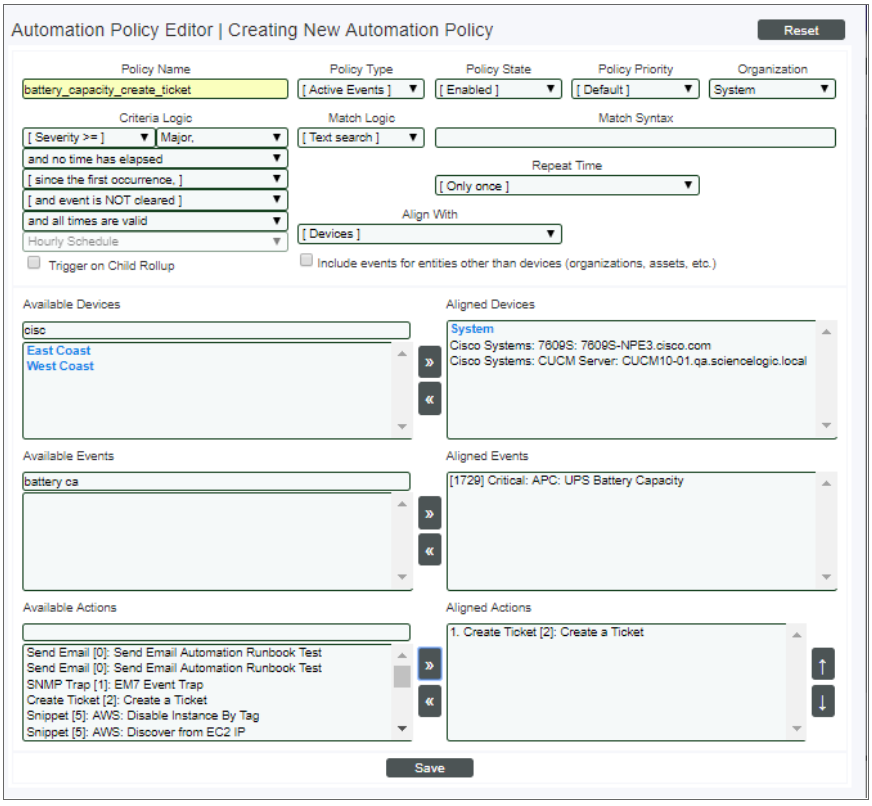
```
Trap Received: (.1.3.6.1.4.1.19567.2.1.0.2.1.403) | Trap Detail : eventID:
12500; eventSeverity: 2; eventSource: 3; elementType: 1; elementID: 48;
elementName: CustB_2821-1.cisco.com; elementAddress: 10.20.30.43; roaID:
0; roaName: System; eventMessage: Configuration management trap received;
subElementType: 0; subElementID:; subElementName:;
```

Action Policy that Creates a Ticket

Suppose we want to automatically create a ticket in response to a specific set of event conditions. We will use a modified version of the automation policy used in the examples above. Suppose that each time an event occurs, we immediately want to create a high priority ticket that specifies the emergency actions that must be performed. In this example, we'll automatically create a ticket about each instance of the event "Critical: APC: UPS Battery Capacity".

Automation Policy

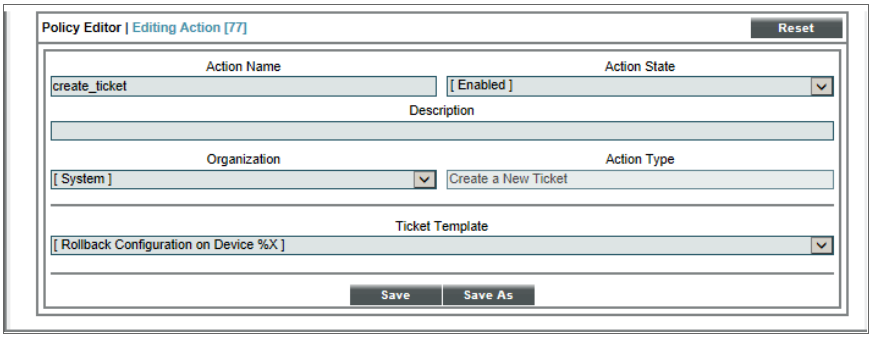
In this example, we'll use a modified version of the Automation Policy we described in *Automation Policies*.



- We specified that the automation policy:
    - Should act upon active events.
    - Is enabled.
    - Is associated with the organization "System".
    - Will be triggered when the specified event has a severity equal to or greater than "Major".
    - Will be triggered as soon as the specified event occurs.
    - The policy will trigger the action only once for each instance of the event.
    - Will be triggered at any time.
    - Will be triggered when the selected event occurs on the selected device.
    - Will be triggered when the event "Critical: APC: UPS Battery Capacity" occurs on the selected device.
- We specified that when all the criteria in the automation policy are met, the action policy "create_ ticket" will be executed.

Action Policy

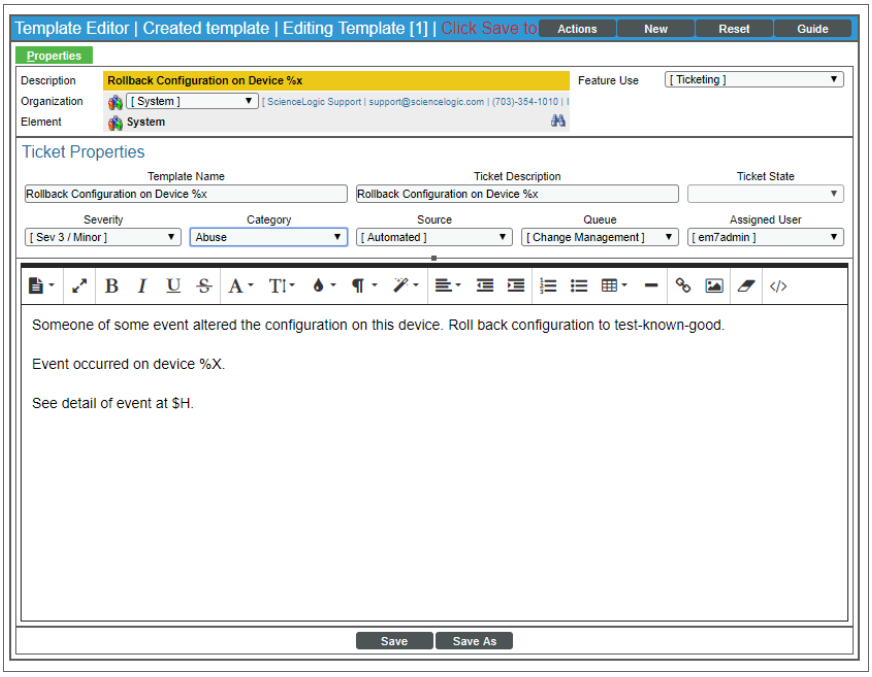
The action policy called "create_ticket" looks like this:



- We specified that this action policy:
    - Is enabled.
    - Will act upon events and devices aligned with the System organization.
    - Will create a new ticket in response to an automation policy.
    - Will use the ticket template "Rollback Configuration on Device %X" to create the ticket.

Ticket Template

The Ticket Template "Rollback Configuration on Device %X" looks like this:



- We specified that the ticket template :
    - Will create a ticket that includes the name of the affected device in the description.
    - Will create a ticket that is associated with the organization "System".
    - Will create a ticket that has a severity of "Minor".
    - Will create a ticket that will be placed in the "Monitoring" ticket queue.
    - Will create a ticket that will be assigned to the user "em7admin".
    - Will create a ticket that will have a category of "Abuse".

- Will create a ticket that will have a source of "Automation".
- Will appear as a choice in action policies.
- Will be triggered as soon as the specified event occurs.
- Will create a ticket that includes note text that reads:

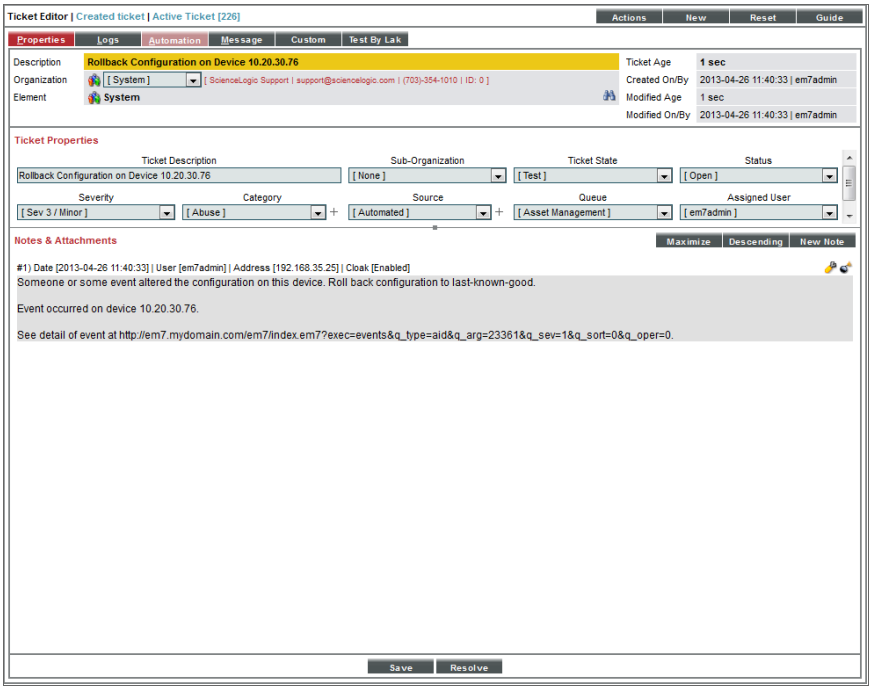
```
Someone or some event altered the configuration on this device.
Roll back configuration to last-known-good.
```

```
Event occurred on device device_name.
See detail of event at link for event.
```

Resulting Ticket

Suppose that the trigger event "UPS Battery Capacity has Degraded Below Threshold" occurred on the device "10.20.30.76".

Our action policy will build a ticket like this:



Action Policy that Writes an SNMP Value to an External Server

You can create an action policy that writes an SNMP value (using the SNMP Set command). You might want to use this type of action policy to perform the following types of tasks:

- ***Change the value of an OID in response to an event***. For example, we could use a Dynamic Application to create an alert. That alert could examine an OID for a specific value (for example, an OID that specifies whether a device will send traps or not). If the OID did ***not*** have a specific value, we could trigger an event. We could create an automation policy that looked for occurrences of the new event. We could define an action policy that performs an SNMPSet and writes the desired value to the OID (for example, assigns a value that allows the device to send traps). When the new event occurred, we could change the value of the OID.
- ***Trigger a script on an external device***. When a specified event occurs (for example, an event that informs us that a network device is not running), we could trigger an automation policy. This automation policy could trigger an action policy that performs an SNMPSet. We could change the value of an OID on the affected external device. The external device must include a script that is also monitoring the value of the changed OID. The script could be triggered when the OID changes. For example, the script might restart the device.

Action Policy that Sends an SQL Query to an External Server

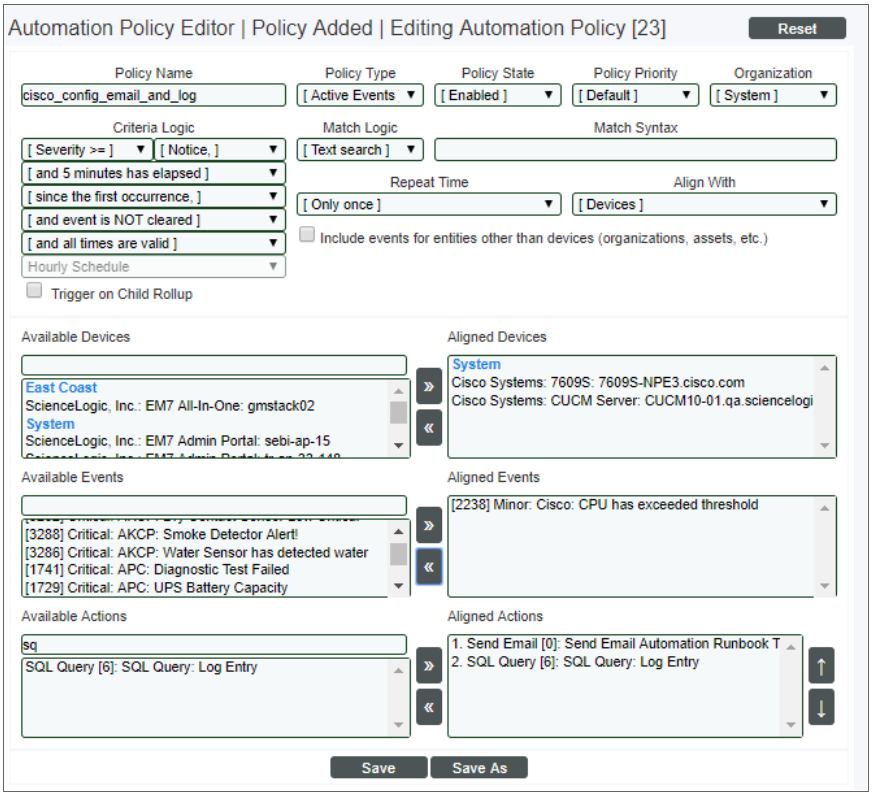
Suppose you want to create a custom Quick Report that displays the number of automation policies that are executed and the date and time each execute occurs. However, by default, Skylar One does not log this information to the system logs or access logs.

To solve this problem, you could create an SQL action that automatically creates a log entry in the audit logs in the database each time an automation policy is executed. You could then include this action in each automation policy, so that Skylar One automatically creates a log entry in the database each time an automation policy is executed.

You could later write a custom Quick Report to retrieve, format, and display the log entries from the database.

Automation Policy

For our example, we'll use a modification of the previous automation policy that sends an email ("cisco_config_email"). Our modification will include an email action and also include an action that use SQL to log the instance of the email action.

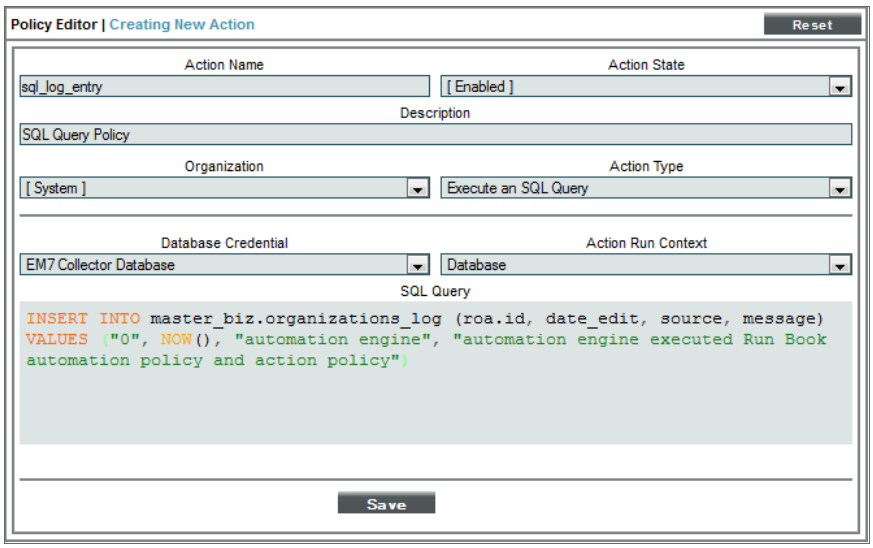


- We specified that the automation policy:
    - Should act upon active events.
    - Is enabled.
    - Is associated with the organization "System".
    - Will be triggered when the specified event has a severity equal to or greater than "Notice".
    - Will be triggered as soon as the specified event occurs.
    - The policy will trigger the action only once for each instance of the event.
    - Will be triggered at any time.
    - Will be triggered when the selected event occurs on at least one of the selected Cisco devices.
    - Will be triggered when the event "Cisco: CPU has exceeded threshold" occurs on at least one of the selected Cisco devices.

- We specified that when all the criteria in the automation policy are met, the action policy "automatic_ email" will be executed.
- We specified that when all the criteria in he automation policy are met, the action policy "sql_log_ entry" will be executed.

Action Policy

The action policy called "sql_log_entry" looks like this:



- We specified that this action policy:
  - is enabled.
  - will act upon events and devices aligned with the System organization.
  - will execute an SQL query.
  - will use the credential "MySQLWrite" to connect to the database.
  - will add a new row of data to the table organizations_log in the database master_biz, using following MySQL INSERT command:

```
INSERT INTO master_biz.organizations_log

(roa.id, date_edit, source, message)

VALUES ("0", NOW(), "automation engine", "automation engine executed Run Book automation policy and action policy")
```

Action Policy that Executes a Snippet and Triggers a New Alert

Skylar One includes a sample action policy that executes a Snippet. This example Snippet pings a device, stores the results in a variable, and makes an entry in a ScienceLogic database table. Skylar One will check the entries in this database table and try to match the messages to an existing event policy.

NOTE: To use this example action policy to trigger an event, you must define an event policy with an ***Event Source*** of *API* and a ***First Match String*** value that will match against the value in the ***Message*** column in the database **in_api**, in the table **messages**. When a new entry is made to the database **in_api**, in the table **messages**, this triggers Skylar One to check the value in the ***Message*** column against any existing event policies.

The code for the Snippet looks like this (line numbers were added for easy reference and are not included in the code):

```
1) import MySQLdb

2) import subprocess

3) CDB_IP = '192.168.9.90'

4) out, err = subprocess.Popen(['ping', '-c 5', EM7_VALUES['%a']],
stdout=subprocess.PIPE, stderr=subprocess.PIPE).communicate()

5) EM7_RESULT = out

6) if ' 0% packet loss' not in out:

7) conn = MySQLdb.connect(user='root', passwd='<password>', host=CDB_IP,
port=7706)

8) cur = conn.cursor()

9) cur.execute("""INSERT INTO 'in_api'.'messages' ('xtype', 'xid',
'message', 'value', 'message_time') VALUES (%s, %s, %s, '', NOW())""",
(EM7_VALUES['%1'], EM7_VALUES['%x'], 'Bad connection to %s' % EM7_VALUES
['%a']))

10) cur.execute("""COMMIT""")
```

The code performs the following:

- Line 1. Tells the code to use the code in the MySQLdb module. This module allows the code to connect to a MySQL database and execute SQL commands.
- Line 2. Tells the code to use the subprocess module to spawn processes, access stdin and stout for those processes, and retrieve return codes for those processes.
- Line 3. Defines the variable CDB_IP, the IP address of the Database Server (to use this example, supply the IP address of the Database Server in your network).
- Line 4. Uses the subprocess module to run the ping command.
    - Notice that the argument for the ping command is EM7_VALUES['%a']. EM7_VALUES is the global dict that allows a Snippet to access the substitution variables. The substitution variable %a contains the IP address for the device where the event occurred.
    - Notice that the results are stored in the variable **out**.
- Line 5. Stores the value of the variable **out** in the global Snippet variable **EM7_RESULT**. The global Snippet variable EM7_RESULT is used to populate the variable **%_EM7_RESULT_%.** The value of the variable **%_EM7_RESULT_%** can be accessed by the next Action Policy.
- Line 6. Defines the criteria for triggering a new event. The code says "If the variable **out** does not contain the value '0% packet loss' perform the following lines of code. If the variable **out** does contain the value '0% packet loss' do not perform the following lines of code."

> **NOTE**: The following lines will enter a row into the database **in_api**, in the table **messages**. This table allows external APIs to trigger an event. When a new entry is made in this database table, it triggers Skylar One to try to match the value in the ***Message*** column with an existing event policy.

- Line 7. Uses the MySQLdb module and the **connect** method to connect to the Database. The connect method passes the user ID, password, IP address of the Database Server, and the port to use to connect to the database.

> **TIP:** In the **connect** method, use the same username and password you would use to connect to the Database through the PHPMyAdmin interface, from the **Appliance Manager** page (System > Settings > Appliances).

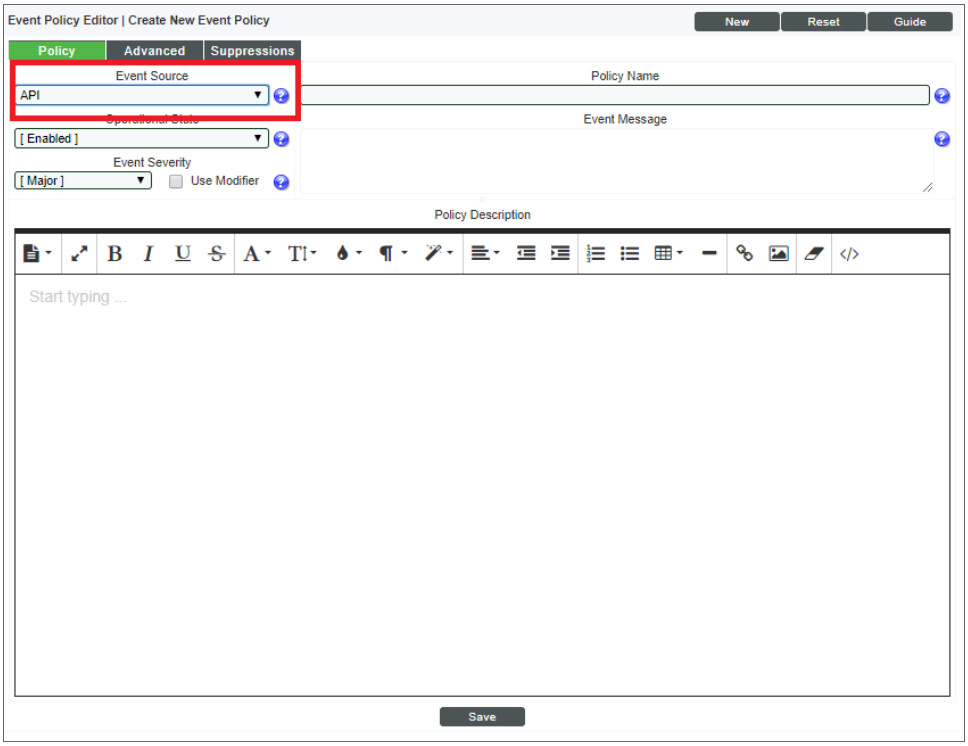
- Line 8. Uses the **cursor** method to create a cursor object for processing SQL statements.
- Line 9. Uses the **execute** method to execute an SQL statement. In this case, the SQL statement says:
    - Perform an INSERT in the database **in_api**, in the table **messages.**
    - Insert values into the following columns: ***xtype***, ***xid***, ***message***, ***value***, ***message time***.
    - For the specified columns, substitute three substitution values (%s in Python), a null value, and the value returned by the **NOW** command.

- Insert into the ***xtype*** column a substitution value, specifically the value variable **%1** (the entity type for the device).
- Insert into the ***xid*** column a substitution value, specifically the value of the variable **%x** (the device ID).
- Insert into the ***message*** column a substitution value, specifically the string 'Bad connection to %s', where the Python substitution value (%s) will be replaced with the value of the variable **%a** (the device's IP address).
- Insert a null value into the ***value*** column.
- Insert into the ***message time*** column the value returned by the **NOW** command (the current date and time).

- Line 10. Uses the **execute** method to execute an SQL statement, specifically to COMMIT the changes to the database **in_api**, in the table **messages**.

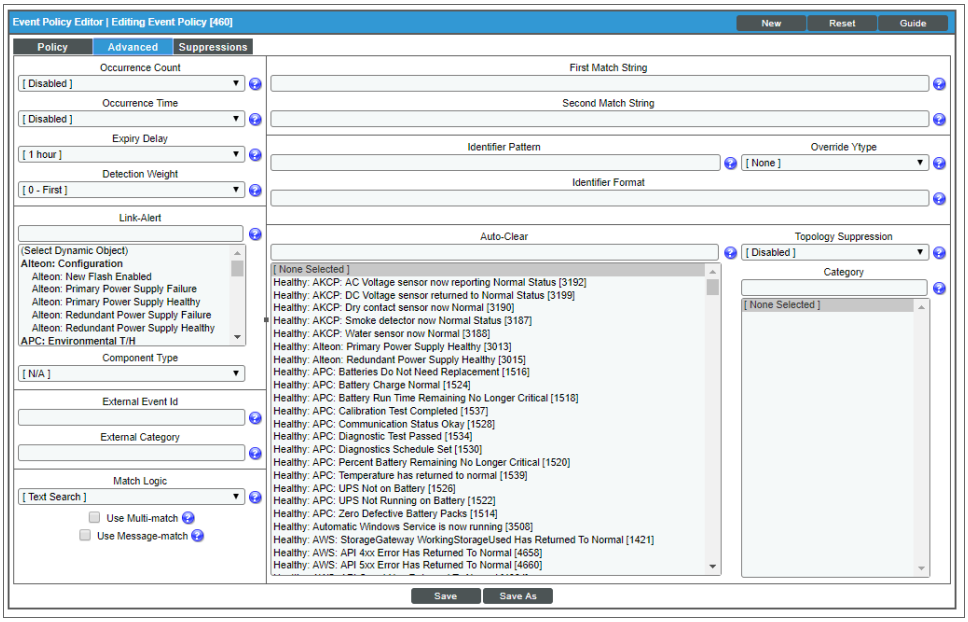
To define an event policy based on the alert (database entry) generated by this Snippet, you would perform the following:

1. Navigate to the **Event Policy Manager** page. (Events > Event Policies)
2. In the **Event Policy Manager** page, click the **[Create]** button.
3. The **Event Policy Editor** page is displayed.
4. In the **Event Policy Editor** page, in the **[Policy]** tab, provide the following values:



- ***Event Source***. Select *API*. This tells Skylar One to look for new entries in the **in_api.messages** table.

5. In the **Event Policy Editor** page, in the **[Advanced]** tab, provide the following values:



- ***First Match String***. Enter a search string that matches the text we entered into the message column of the database table. In this case, we would enter "*Bad connection to*".
- In the ***Match Logic*** field, we also selected *Text Search*, to tell Skylar One to search for the text string we entered in the ***First Match String*** field, and not a regular expression.

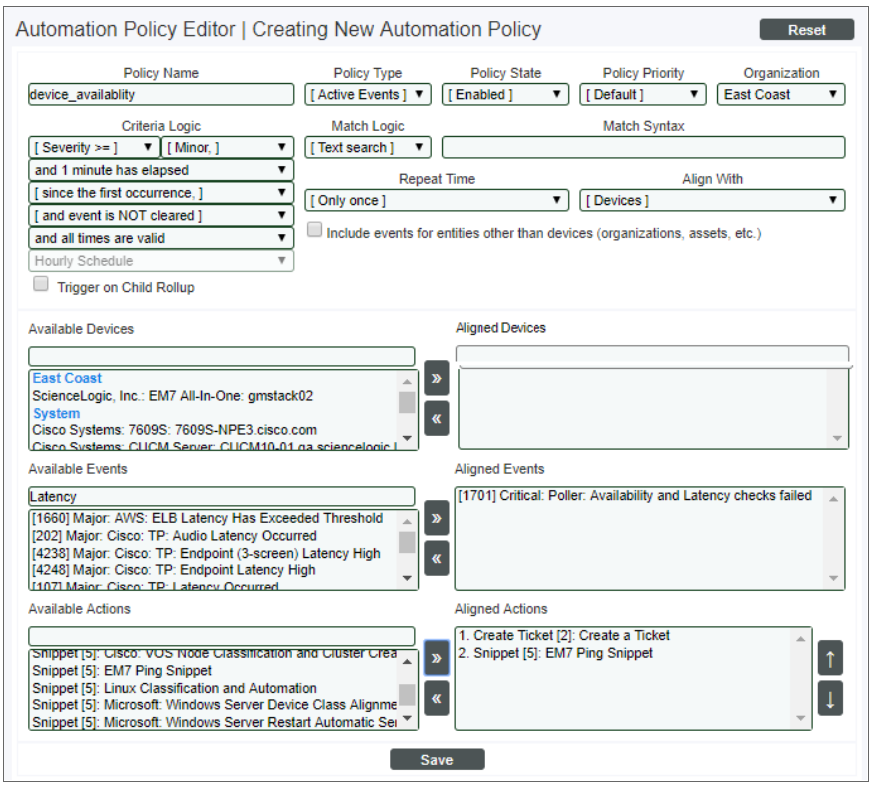
6. For additional details on the **Event Policy Editor** page and tabs and creating event policies, see the manual ***Events***.
7. Click the **[Save]** button to save your new event.
8. The event will be triggered each time a new entry is made to the database **in_api**, in the table **messages**, that contains the text "Bad connection to".

Action Policy that Executes a Snippet and Sends the Results to a Second Action Policy

Suppose that when Skylar One generates an event saying that a device is not available, we want to ping the device from a Data Collector. Suppose that we then want to create a ticket that contains the results of the ping, so we can troubleshoot the availability problem. To do this, we could create an automation policy that executes two action policies, one that executes the ping (a Snippet Action Policy) and one that creates a ticket (a Ticket Action Policy).

Automation Policy

Our automation policy would look like this:

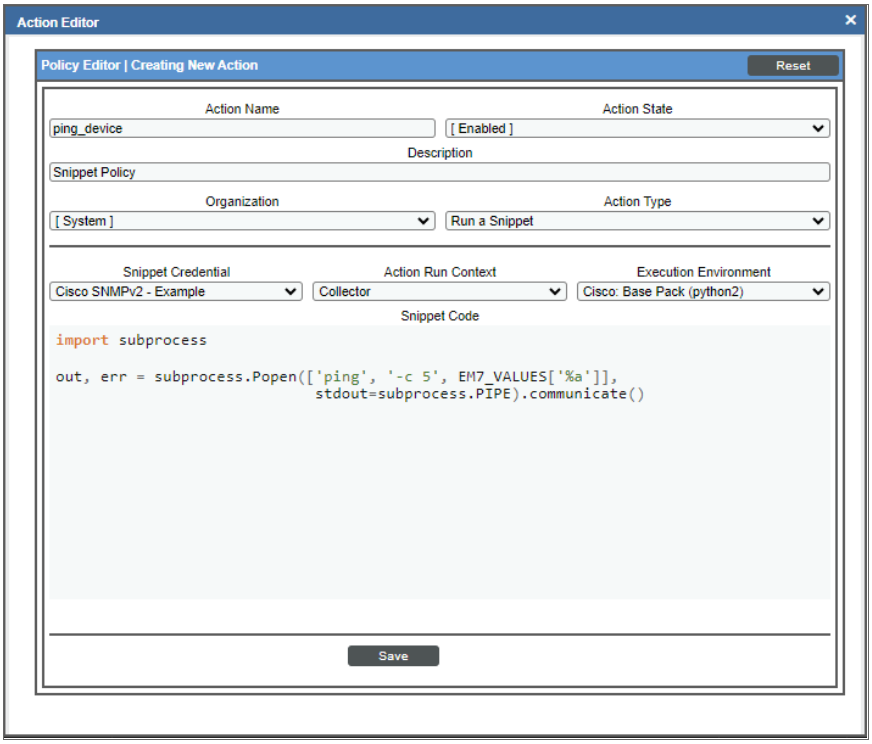


- We specified that the automation policy:
    - Should act upon active events.
    - Is enabled.
    - Is associated with the organization "System".
    - Will be triggered when the specified event has a severity equal to or greater than "Minor".
    - Will be triggered 1 minute after the event occurs and is not cleared.
    - The policy will trigger the action policies once for each occurrence of the event(s).
    - Will be triggered at any time.
    - Will be triggered when the selected event occurs on at least one of the selected devices.
    - Will be triggered when the event "Critical Poller: Availability and Latency checks failed" occurs on at least one of the selected devices (which in this case is all devices).

- We specified that when all the criteria in the automation policy are met, the action policy "ping_ device" and then the action policy "Create Ticket: Create Ping Ticket" will be executed, in the order specified.

Snippet Action Policy

The Snippet Action Policy would look like this:



NOTE: The ***Execution Environment*** field's drop-down list displays the specific Python version of each execution environment.

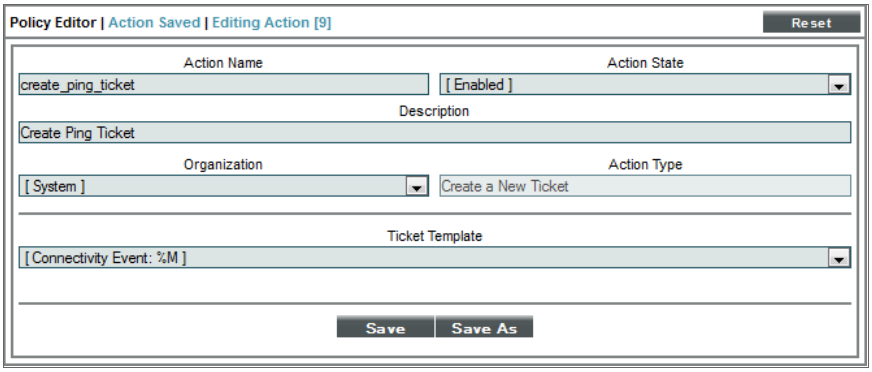
This action policy:

- Tells the code to use the **subprocess** module to spawn processes, access stdin and stout for those processes, and retrieve return codes for those processes.
- Uses the **subprocess** module to run the **ping** command.
    - Notice that the argument for the ping command is EM7_VALUES['%a']. EM7_VALUES is the global dictionary that allows a Snippet to access the substitution variables. The substitution variable ***%a*** contains the IP address for the device where the event occurred.
    - Notice that the results are stored in the variable **out**.

- The value of the variable **out** is stored in the global Snippet variable **EM7_RESULT**. The global Snippet variable EM7_RESULT is used to populate the variable **%_EM7_RESULT_%.** The value of the variable **%_EM7_RESULT_%** can be accessed by the next Action Policy.

Ticket Action Policy

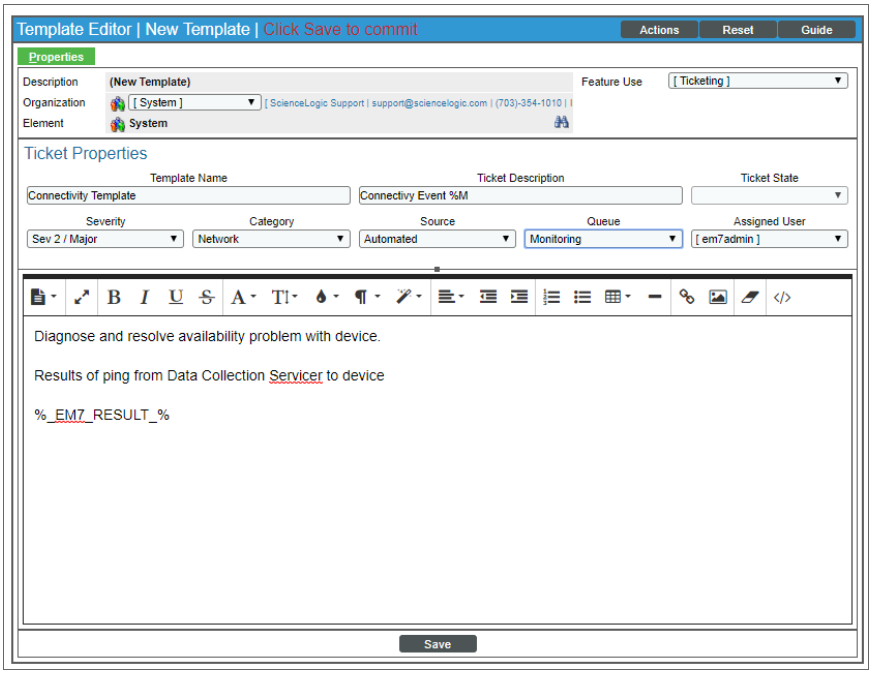
The Ticket Action Policy would look like this:



- We specified that this action policy:
  - Is enabled.
  - Will act upon events and devices aligned with the System organization.
  - Will create a new ticket in response to an automation policy.
  - Will use the ticket template "Connectivity Event: %M" to create the ticket.

Ticket Template

The Ticket Templates specified in the Create Ticket Action Policy would look like this:



- We specified that the ticket template:
    - Will create a ticket that includes the event description (%M) in the description.
    - Will create a ticket that has a severity of "Major".
    - Will create a ticket that is associated with the organization "System".
    - Will create a ticket with the category "Network".
    - Will create a ticket that will be placed in the "Monitoring" ticket queue.
    - Will create a ticket with the source "Automated".

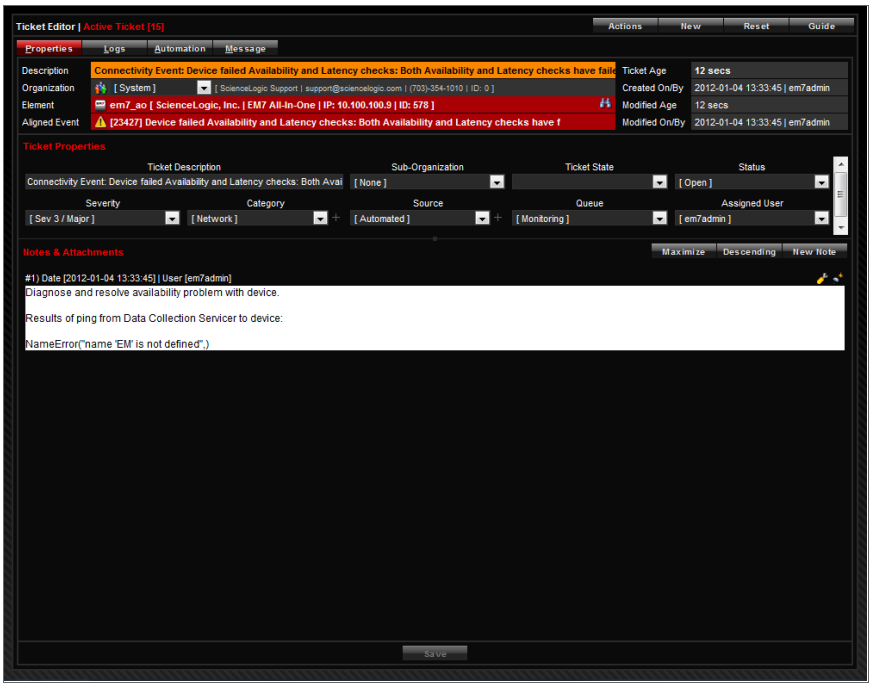
- Will create a ticket that will be assigned to the user "em7admin".
- Will appear as a choice in action policies.
- Will create a ticket that includes note text that reads:

```
Diagnose and resolve availability problem with device.

Results of ping from Data Collection Server to device:

%_EM7_RESULT_%
```

Where the variable %_EM7_RESULT_% will contain the results from the previous Snippet Action Policy. In this case, the variable %_EM7_RESULT_% will contain the results from a ping from the Data Collector to the device where the availability event occurred.

Resulting Ticket

The resulting ticket would look like this:



Appendix

A

Run Book Variables

Overview

This appendix defines the different variables you can use when creating an action policy.

This appendix covers the following topics:

This chapter covers the following topics:

Run Book Variables *117*

Run Book Variables

You can include variables when creating an action policy. These variables are listed in the table below.

- In an action policy of type ***Send an Email Notification***, you can include one or more of these variables in the fields ***Email Subject*** and ***Email Body***.
- In an action policy of type ***Send an SNMP Trap***, you can include one or more of these variables in the ***Trap OID*** field, ***Varbind OID*** field, and the ***Varbind Value*** field.
- In an action policy of type ***Create a New Ticket***, you can include one or more of these variables in the ***Description*** field or the ***Note*** field of the related Ticket Template.
- In an action policy of type ***Send an SNMP Set***, you can include one or more of these variables in the ***SNMP OID*** field and the ***SNMP Value*** field.
- In an action policy of type ***Run A Snippet***, you can access variables from the global dictionary ***EM7_VALUES***.
- In a policy of type ***Execute an SQL Query***, you can include one or more of these variables in the ***SQL Query*** field.

Variable	Source	Description
%A	Account	Username
%a	Entity	IP address
%F	Dynamic Alert	Alert ID for a Dynamic Application Alert
%g	Asset	Asset serial
%h	Asset	Device ID associated with the asset
%I (uppercase "eye")	Dynamic Alert	For events with a source of "dynamic", this variable contains the index value from SNMP. For events with a source of "syslog" or "trap", this variable contains the value that matches the ***Identifier Pattern*** field in the event definition.
%i (lowercase "eye")	Asset	Asset Location
%K	Asset	Asset Floor
%k	Asset	Asset Room
%L	Dynamic Alert	Value returned by the label variable in a Dynamic Application Alert.
%m	Automation	Automation policy note
%N	Action	Automation action name
%n	Automation	Automation policy name
%P	Asset	Asset plate
%p	Asset	Asset panel
%Q	Asset	Asset punch

Variable	Source	Description
%q	Asset	Asset zone
%T	Dynamic Alert	Value returned by the Threshold function in a Dynamic Application Alert.
%U	Asset	Asset rack
%u	Asset	Asset shelf
%v	Asset	Asset tag
%W	Asset	Asset make
%w	Asset	Asset model
%V	Dynamic Alert	Value returned by the Result function in a Dynamic Application Alert.
%_category_id	Entity	Device category ID associated with the entity in the event.
%_category_ name	Entity	Device category name associated with the entity in the event.
%_class_id	Entity	Device class ID associated with the entity in the event.
%_class_name	Entity	Device class description associated with the entity in the event.
%_parent_id	Entity	For component devices, the device ID of the parent device.
%_parent_ name	Entity	For component devices, the name of the parent device.
%_root_id	Entity	For component devices, the device ID of the root device.
%_root_name	Entity	For component devices, the name of the root device.
%_service_ investigator_url	Entity	The URL of the Business Service Investigator page for the event that triggered the automation (for run book actions that run against events aligned with business services).
%1 (one)	Event	Entity type. Possible values are: • 0. Organization • 1. Device • 2. Asset • 4. IP Network • 5. Interface • 6. Vendor • 7. Account • 8. Virtual Interface • 9. Device Group • 10. IT Service • 11. Ticket
%2	Event	Sub-entity type. Possible values for organizations are:

Variable	Source	Description
		• 9. News feed Possible values for devices are: • 1. CPU • 2. Disk • 3. File System • 4. Memory • 5. Swap • 6. Component • 7. Interface • 9. Process • 10. Port • 11. Service • 12. Content • 13. Email
%4	Event	Text string of the user name that cleared the event.
%5	Event	Date/time when event was deleted.
%6	Event	Date/time when event became active.
%7	Event	Event severity (1-5), for compatibility with previous versions of Skylar One. 1=critical, 2=major, 3=minor, 4=notify, 5=healthy. **NOTE**: When referring to an event, %7 represents severity (for previous versions of Skylar One). When referring to a ticket, %7 represents the subject line of an email used to create a ticket.
%c	Event	Event counter
%d	Event	Date/time when last event occurred.
%D	Event	Date/time of first event occurrence.
%e	Event	Event ID
%H	Event	URL link to event
%M	Event	Event message
%s	Event	severity (0 - 4). 0=healthy, 1=notify, 2=minor, 3=major, 4=critical.
%S	Event	Severity (HEALTHY - CRITICAL)
%_user_note	Event	Current note about the event that is displayed on the **Events** page.
%x	Event	Entity ID
%X	Event	Entity name

Variable	Source	Description
%y	Event	Sub-entity ID
%Y	Event	Sub-entity name
%Z	Event	Event source (Syslog - Group)
%z	Event	Event source (1 - 8)
%_ext_ticket_ref	Event	For events associated with an external Ticket ID, this variable contains the external Ticket ID.
%3	Event Policy	Event policy ID
%E	Event Policy	External ID from event policy
%f	Event Policy	Specifies whether event is stateful, that is, has an associated event that will clear the current event. 1 (one)=stateful; 0 (zero)=not stateful.
%G	Event Policy	External Category
%R	Event Policy	Event policy cause/action text
%_event_policy_name	Event Policy	Name of the event policy that triggered the event.
%B	Organization	Organization billing ID
%b	Organization	Impacted organization
%C	Organization	Organization CRM ID
%o (lowercase "oh")	Organization	Organization ID
%O (uppercase "oh")	Organization	Organization name
%r	System	Unique ID / name for the current Skylar One system
%7	Ticket	Subject of email used to create a ticket. If you specify this variable in a ticket template, Skylar One will use the subject line of the email in the ticket description or note text when Skylar One creates the ticket. **NOTE**: When referring to a ticket, %7 represents the subject line of an Email used to create a ticket. When referring to an event, %7 represents severity (for previous versions of Skylar One).
%t	Ticket	Ticket ID
%J	Ticket	Description field from the Skylar One ticket.

Appendix

B

Example: Adding a Weather Report to a Ticket

Overview

This example describes:

- A snippet action that requests weather data from a public API using the location of the associated device
- A test automation policy that triggers the snippet action and adds the output to an existing Skylar One ticket

This example covers the following topics:

Weather Report: Snippet Design 2

Weather Report: Snippet Code 2

Creating the Credential 4

Creating the Automation Action 5

Creating the Update Ticket Action 6

Creating the Automation Policy 7

Configuring a Test Event 9

Testing the Automation Policy 10

Configuring a Device Asset Record 10

Triggering a Test Event 11

Creating a Ticket for the Test Event 13

Possible Next Steps 15

Weather Report: Snippet Design

Snippet code uses the third-party Python Requests library, which is installed on Skylar One Database Server and Data Collector appliances, to perform an HTTP request. The documentation for the library is available at the link below:

http://docs.python-requests.org

This example uses OpenWeatherMap, a public API, to request a current weather report. To use the example code, you must request a free API key to include in the credential associated with the snippet action. The documentation for the API that provides current weather data is available at the link below:

https://openweathermap.org/current

The documentation for all available APIs and a link to sign up for an API key are available at the link below:

https://openweathermap.org/api

NOTE: ScienceLogic provides this documentation for the convenience of ScienceLogic customers. Some of the configuration information contained herein pertains to third-party vendor software that is subject to change without notice to ScienceLogic. ScienceLogic makes every attempt to maintain accurate technical information and cannot be held responsible for defects or changes in third-party vendor software. There is no written or implied guarantee that information contained herein will work for all third-party variants. See the End User License Agreement (EULA) for more information.

To use the API, you must provide the following data in the parameters of the HTTP request:

- The API key
- The location for which the current weather data is being requested

For example, to request the current weather in London, you would use the following URL, inserting the API key where indicated:

```
https://api.openweathermap.org/data/2.5/weather?q=London&units=metric&app
id=[API Key]
```

The snippet action uses the asset location of the device associated with the triggering event, which is provided as a variable to the snippet code.

Weather Report: Snippet Code

The following snippet code is used in this example and is explained in the section below:

```
1     import requests, json
```

```
if type(EM7_ACTION_CRED) == dict and EM7_ACTION_CRED['cred_type']
== 3:

  if EM7_VALUES['%i'] != "" and EM7_VALUES['%i'] is not None:

    uri = EM7_ACTION_CRED['curl_url']

    uri = uri.replace("%i",EM7_VALUES['%i'])

    try:

      response = requests.get(uri, timeout=EM7_ACTION_CRED['cred_
      timeout'])

    except requests.exceptions.RequestException as e:

      EM7_RESULT = "Automation Action Error: Failed request for
      URL: %s Exception information: %s" % (url, e)

    else:

      if response.status_code < 300:

        EM7_RESULT = "<b>Weather Report</b>:<br />%s" %

        (json.dumps(response.json(),indent=2).replace("\n","<br
        />").replace(" ", "  "),)

      else:

        EM7_RESULT = "Automation Action Error: Non-2XX status
        code: %s returned from request to URL %s" %

         (response.status_code,url)

  else:

    EM7_RESULT = "Automation Action Error: No location specified
    in asset record"

else:

  EM7_RESULT = "Automation Action Error: SOAP/XML credential not
  aligned to action"
```

The code performs the following steps:

- ***Line 1***. The necessary Python modules (requests and JSON) are imported.
- ***Line 3***. The credential associated with the automation action (**EM7_ACTION_CRED**) is evaluated to ensure a credential is aligned (i.e. **EM7_ACTION_CRED** is a dictionary) and that the credential is the expected type (SOAP/XML).
- ***Line 4***. The variable that contains the asset location for the device that triggered the automation (**EM7_VALUES['%i']**) is evaluated to ensure a valid location value is set (i.e. the variable is not None or an empty string).
- ***Lines 5 - 6***. The EM7_ACTION_CRED dictionary is used to populate the variable "uri" and the asset location is substituted where "%i" appears in the URL.
- ***Lines 7 - 10***. The requests module is used to perform an HTTP GET request on the URL using the timeout specified in the credential (`EM7_ACTION_CRED['cred_timeout']`). If the `requests.get` function generates an exception, the output of the action (the contents of the **EM7_RESULT** variable) is set to an error message that includes the URL and the exception text.
- ***Lines 12 - 13***. If the `requests.get` function executed correctly (i.e. it did not generate an exception), the HTTP status code is evaluated to determine whether the HTTP request was successful. If the request was successful (i.e. the HTTP status code is in the 200 range), the output of the action (the contents of the EM7_RESULT variable) is set to a string that contains the response. The following formatting functions are applied:
  - The API responds in JSON, so the `dumps` function in the JSON module is used to format the response as a multi-line string with indentation.
  - The Skylar One ticketing system is able to render notes in HTML format, so the unix-style line-endings in the string are replaced with HTML break tags.
  - To preserve the whitespace used to indent the JSON elements, each indentation (2 space characters) is replaced with two HTML non-breaking space characters.
  - The test "Weather Report" is added to the beginning of the string with basic HTML formatting.
- ***Lines 14 - 19***. If any of the following conditions occur, the output of the action (the contents of the EM7_RESULT variable) is set to an appropriate error message, as follows:
  - The received status code indicated the HTTP request was unsuccessful.
  - The asset record for the device that triggered the automation does not include a location value.
  - The automation action does not have a credential aligned or the aligned credential is of an incorrect type.

Creating the Credential

To create the credential, perform the following steps:

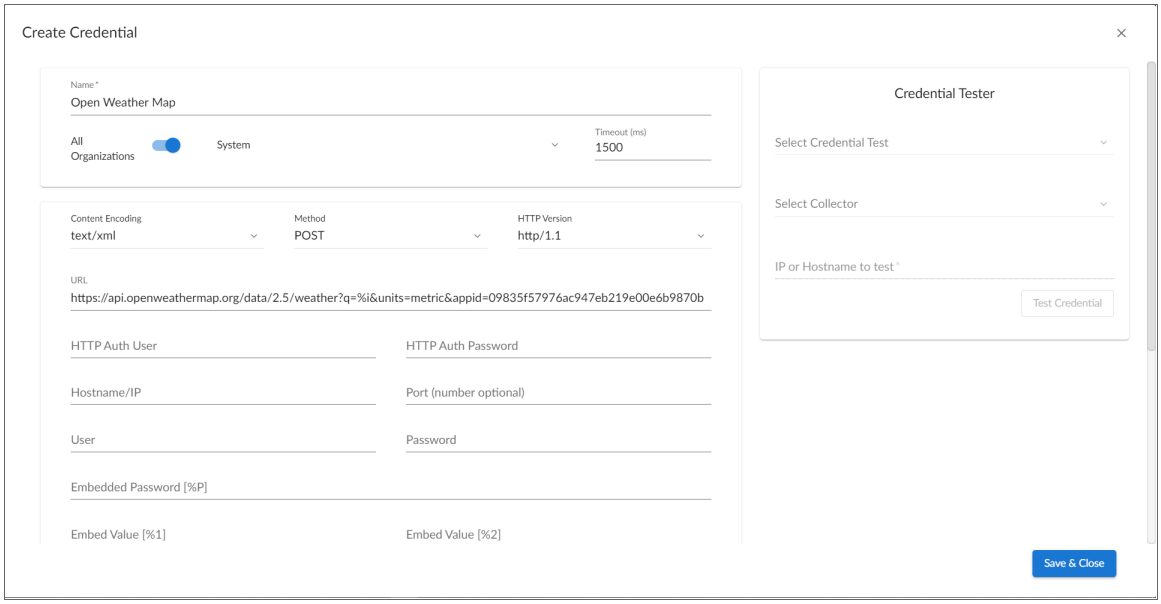
1. Go to the **Credential Management** page (Manage > Credentials).
2. Click **[Create New]** and then select ***Create SOAP/XML Host Credential***.
3. Supply values in the following fields. The default value can be used in all other fields in this page:

- ***Profile Name***. Enter a name for the credential. In this example, we entered "Open Weather Map".
- ***URL***. Enter the URL to use to request the current weather, including the "%i" substitution character as a placeholder for the asset location. The URL must use your API key. In this example we used the following URL, with the API key substituted where indicated:

```
https://api.openweathermap.org/data/2.5/weather?q=%i&units=metric&appid=[API Key]
```

For example, if your API key is 09835f57976ac947eb219e00e6b9870f, the URL would be:

```
https://api.openweathermap.org/data/2.5/weather?q=%i&units=metric&appid=09835f57976ac947eb219e00e6b9870b
```



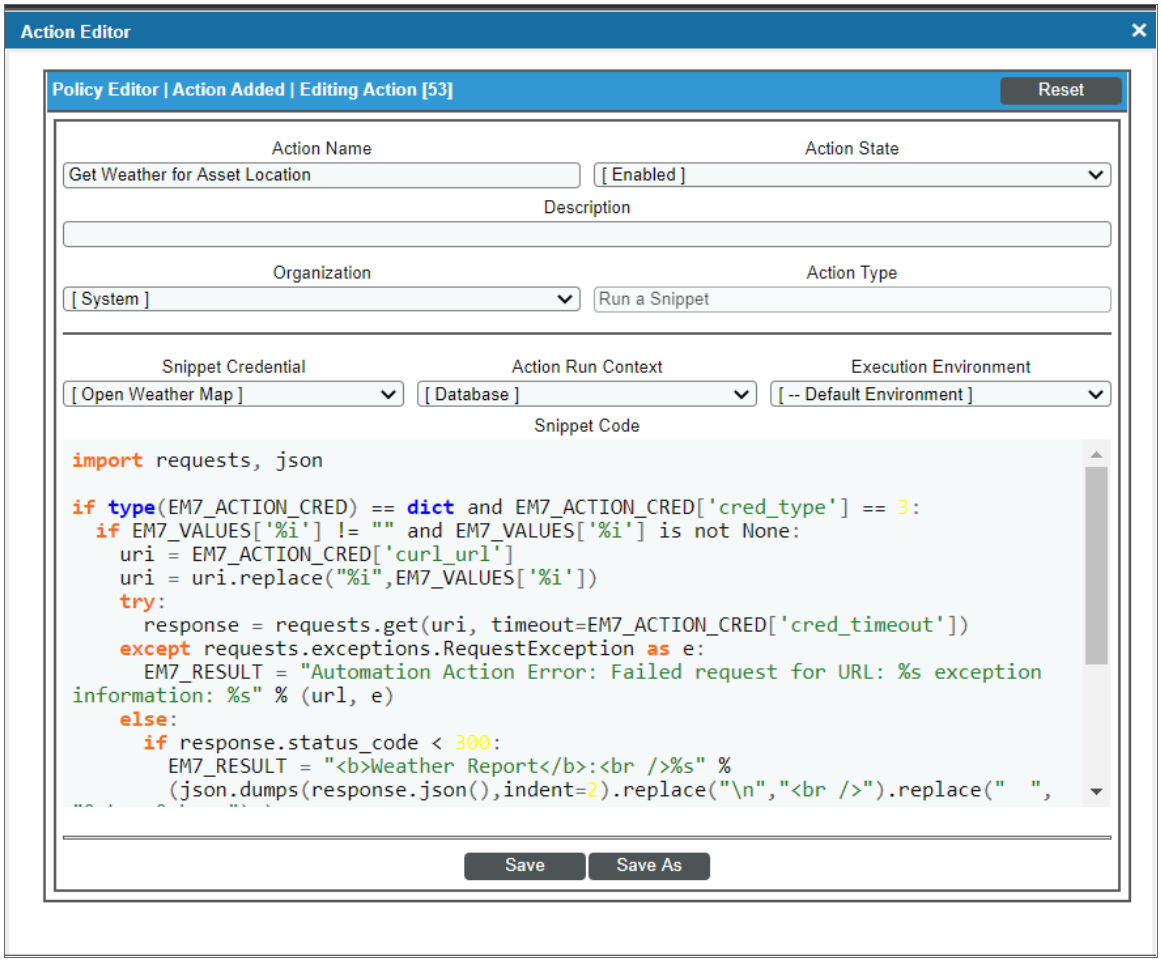
4. Click **[Save]**.

Creating the Automation Action

To create the automation action, perform the following steps:

1. Go to the **Action Policy Manager** page (Registry > Run Book > Actions).
2. Click **[Create]**.
3. Supply values in the following fields. The default value can be used in all other fields in this page:
   - ***Action Name***. Enter a name for the action. In this example, we entered "Get Weather for Asset Location".
   - ***Action Type***. Select *Run a Snippet*.
   - ***Snippet Credential***. Select the credential you created in the *Creating the Credential* section.

- ***Action Run Context***. This example assumes that the Database Servers or All-In-One Appliances in the Skylar One system have internet connectivity, so *Database* is selected in this field.
- ***Snippet Code***. Enter the snippet code in this field.



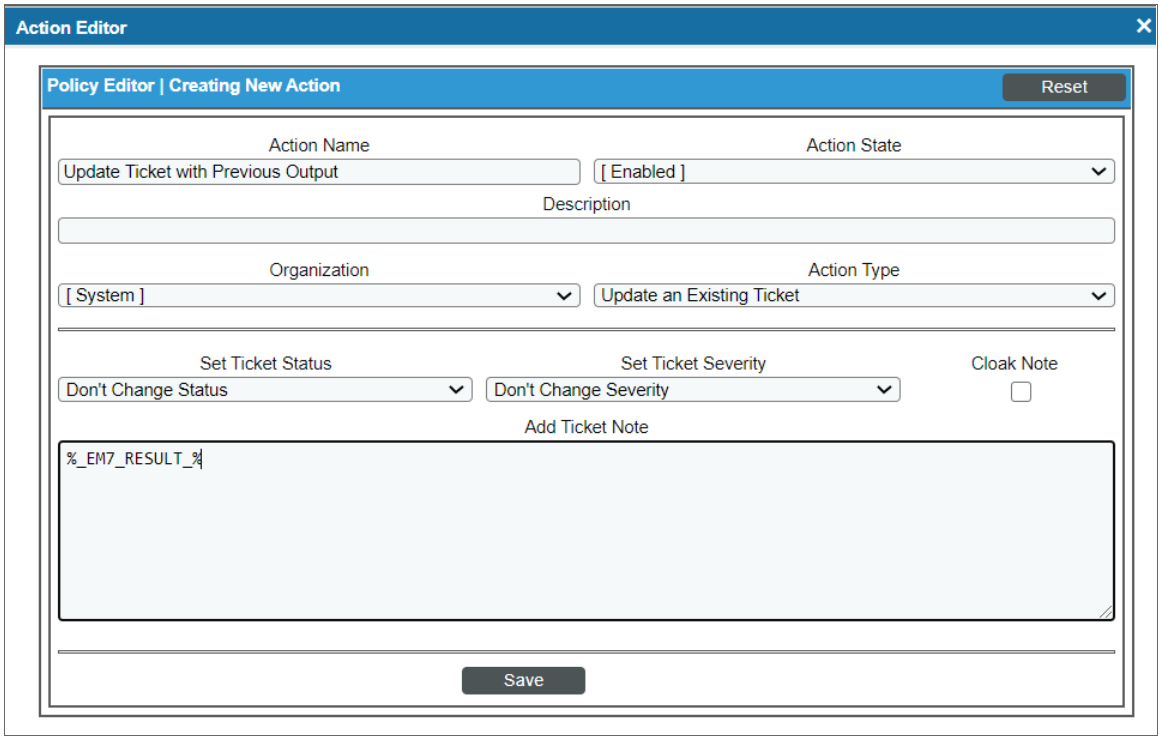
4. Click **[Save]**.

Creating the Update Ticket Action

In this example, the output of the snippet action is added as a note in the Skylar One ticket associated with the event. To do this, a second automation action is included in the automation policy. To create the second automation action, perform the following steps:

1. Go to the **Action Policy Manager** page (Registry > Run Book > Actions).
2. Click **[Create]**.
3. Supply values in the following fields. The default value can be used in all other fields in this page:

- ***Action Name***. Enter a name for the action. In this example, we entered "Update Ticket with Previous Output".
- ***Action Type***. Select *Update an Existing Ticket*.
- ***Add Ticket Note***. Enter the note text to add to the ticket, including substitution characters. In this example, we entered the substitution character for the result of the previous action, "%_EM7_RESULT_%".



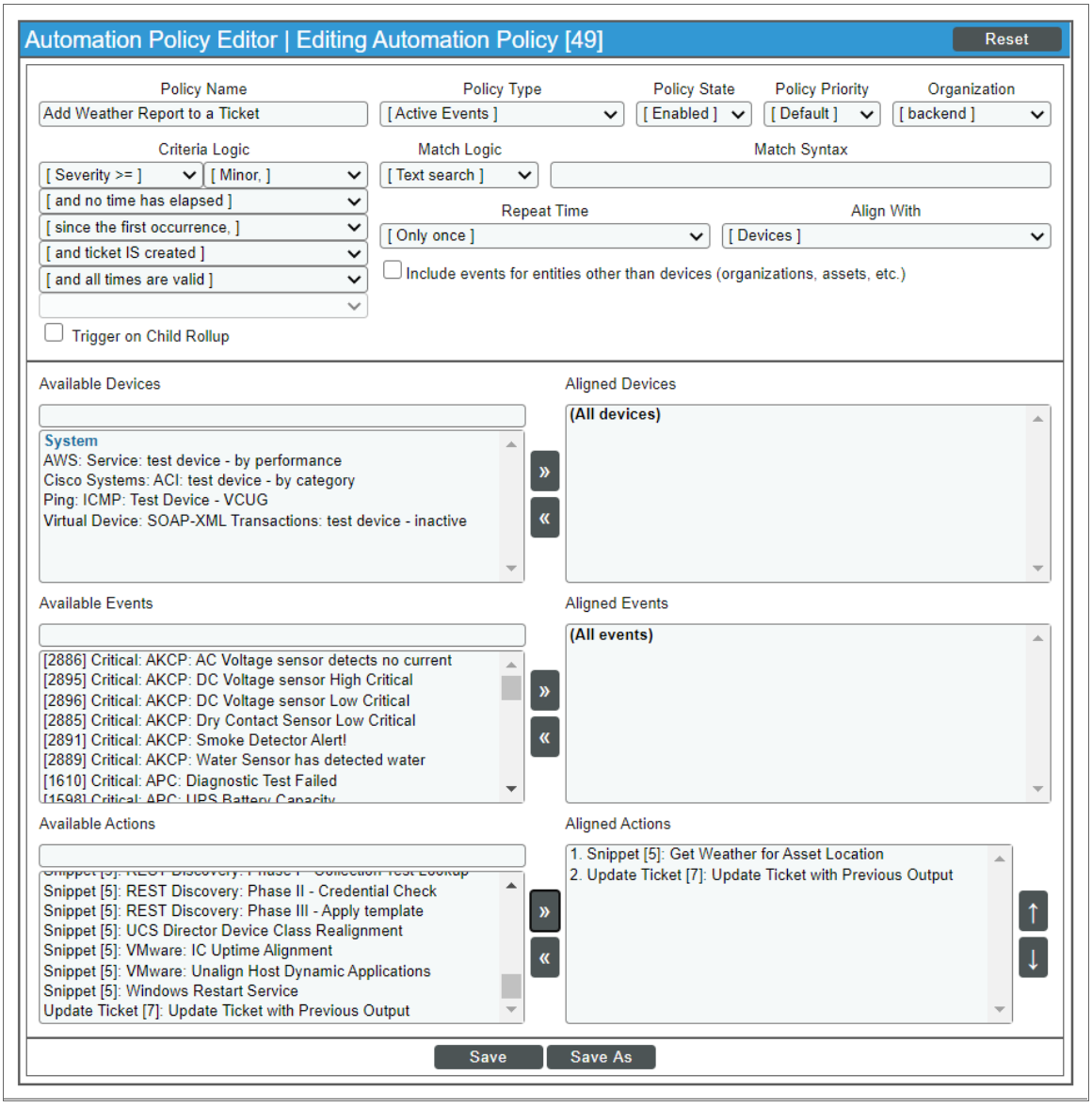
4. Click **[Save]**.

Creating the Automation Policy

To create the automation policy used in this example, perform the following steps:

1. Go to the **Automation Policy Manager** page (Registry > Run Book > Automation).
2. Click **[Create]**.
3. Supply values in the following fields. The default value can be used in all other fields in this page. However, using the default values causes any minor severity and above event on any device to trigger the policy. You might want to change the values in the ***Organization***, ***Align With***, ***Aligned Devices***, and ***Aligned Events*** fields to scope the policy based on your organization's needs:
   - ***Policy Name***. Enter a name for the policy. In this example, we entered "Add Weather Report to Ticket".

- ***Criteria Logic***. In this example, the automation policy is configured to trigger immediately when a ticket exists for any event with a severity of minor and above. The ***Criteria Logic*** fields are configured to read "Severity >= Minor and no time has elapsed since the first occurrence and ticket IS created and all times are valid".
- ***Aligned Actions***. Move the following actions from the Available Actions field to the Aligned Actions field in order:
    - Snippet: Get Weather for Asset Location.
    - Update Ticket: Update Ticket with Previous Output.

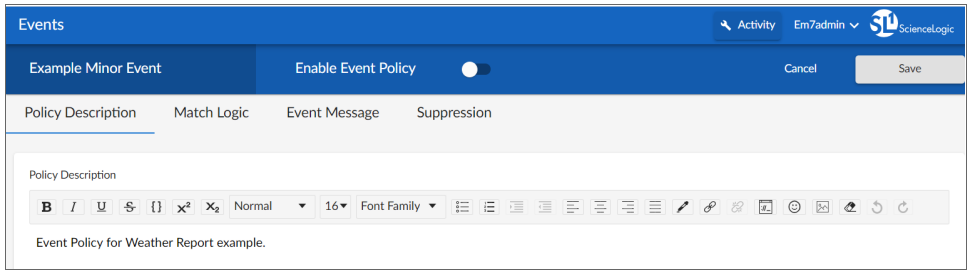


4. Click **[Save]**.

Configuring a Test Event

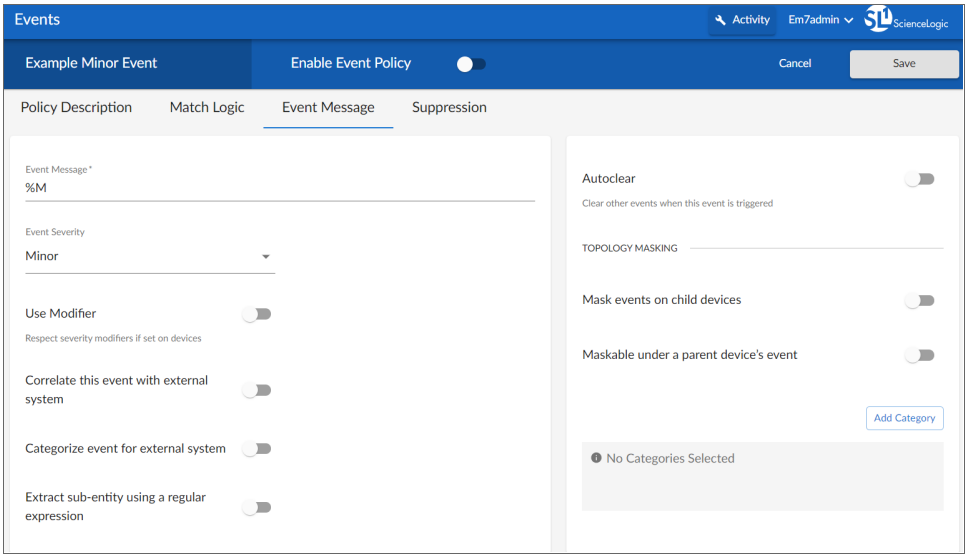
To test an automation policy, you can create an Event Policy of type "API" to manually trigger events on devices via the API browser. To create a test event policy:

1. Go to the **Event Policy Manager** page (Events > Event Policies).
2. Click **[Create Event Policy]**.
3. Supply values as follows. The default value can be used in all other fields in this page:
    - ***Policy Description Tab***. Set the fields on the **Policy Description** tab as follows:
        - ***Policy Name***. Enter a name for the policy. In this example, we entered "Example Minor Event".
        - ***Policy Description***. Enter a description for your policy. In this example, we entered "Event Policy for Weather Report example."



    - ***Match Logic Tab***. Set the fields on the **Match Logic** tab as follows:
        - ***Event Source***. Select *API*.
        - ***Match String***. Enter unique text that will appear in the message provided in the API request. In this example, we entered "Example Minor".
    - ***Event Message Tab***. Set the fields on the **Event Message** tab as follows:
        - ***Event Message***. In this example, we entered "%M" to use the message text provided in the API request.

- ***Event Severity***. Select a severity. In this example, we selected *Minor*.



4. Click **[Save]**.

NOTE: It can take up to five minutes for the event engine to start using a new event policy.

Testing the Automation Policy

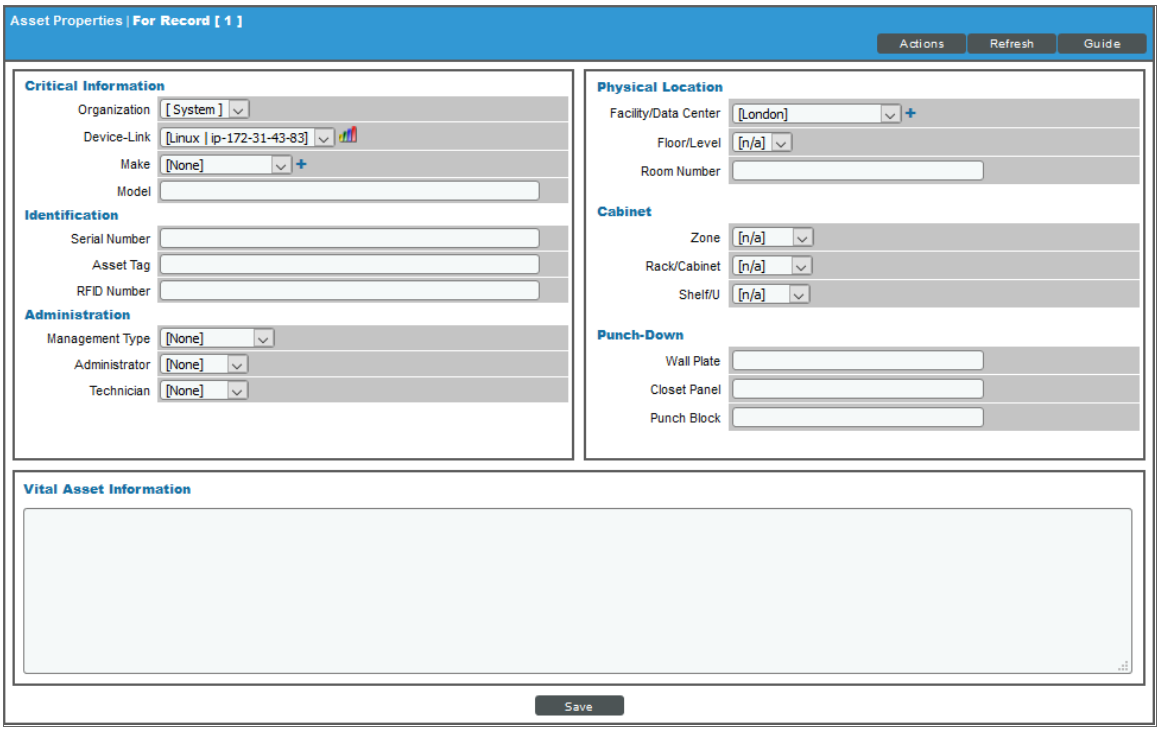
To test the automation policy used in this example, you must do the following:

- Configure the asset record of a test device to use a valid location value
- Trigger the test event
- Create a ticket for the test event

Configuring a Device Asset Record

To add a location value to a device asset record, perform the following steps:

1. Go to the Device Manager page (Devices > Classic Devices, or Registry > Devices > Device Manager in the classic user interface).
2. Locate a test device and click its asset icon ( ). The Asset Properties page appears:



3. Change the value in the ***Facility/Data Center*** field to a value that is valid for the weather API (i.e. a city name). You can add a new option to the drop-down list by clicking the plus icon ( ) and typing in the field. In this example, we added the value "London".
4. Click **[Save]**.

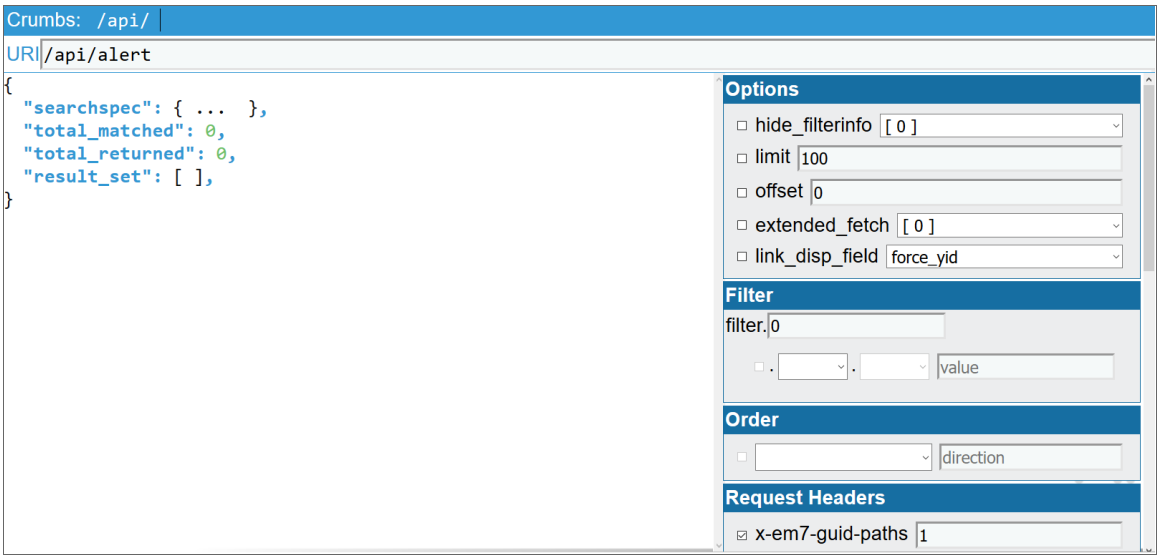
Triggering a Test Event

To trigger a test event, you will need:

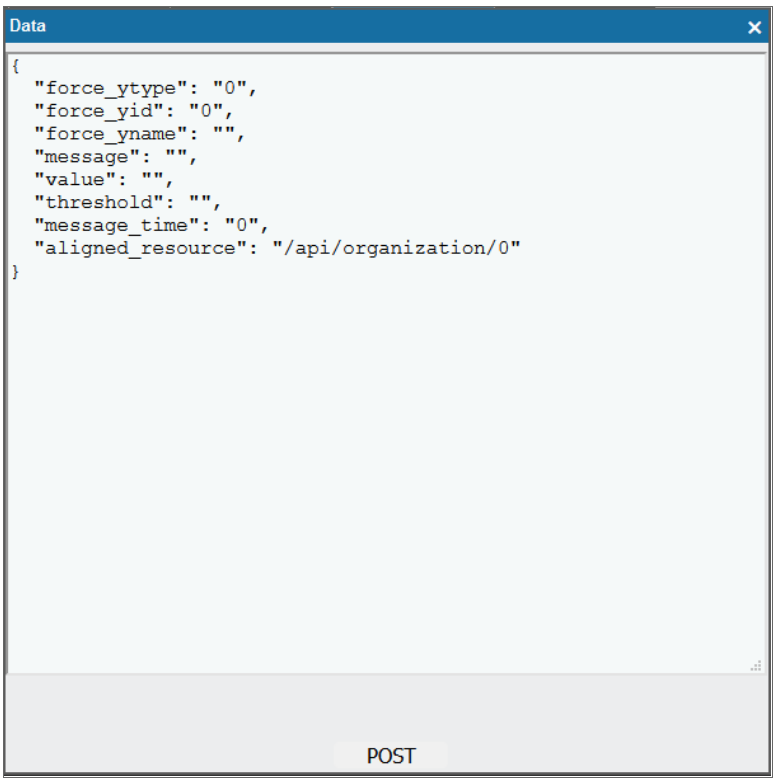
- The device ID of the test device. The device ID is displayed in the DID column in the Device Manager page (Devices > Classic Devices, or Registry > Devices > Device Manager in the classic user interface).
- The ***Match String*** value from the test event policy you created.

To trigger a test event, perform the following steps:

1. Go to the **API Browser** page (System > Tools > API Browser).
2. In the main pane, enter "/api/alert" in the URI field. The "/api/alert" resource is loaded:



3. In the lower right of the page, locate the **Actions** section and click **[POST]**. The **Data** modal appears:



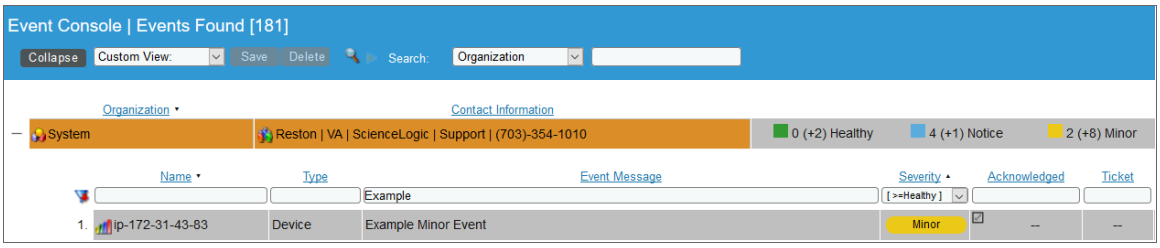
4. Change the following JSON attributes:
   - ***message***. Add a text string that includes the ***Match String*** value from the test event policy.
   - ***aligned_resource***. Change the value to "/api/device/*<DID>*", replacing *<DID>* with the device ID of the test device.

5. Click **[POST]**.

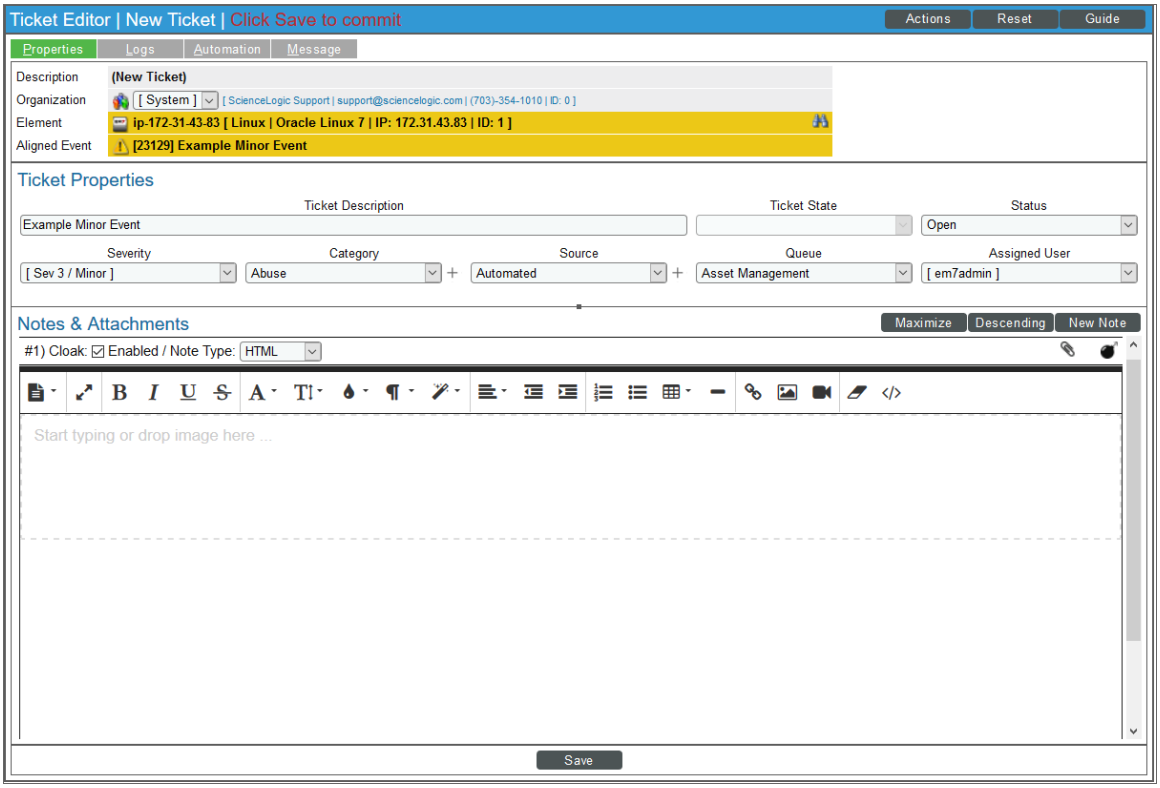
Creating a Ticket for the Test Event

To view the test event and create a ticket:

1. Go to the **Events** page.
2. In the field at the top of the "Message" column, enter the ***Match String*** value from the test event policy. The filtered list will include the test event you triggered:

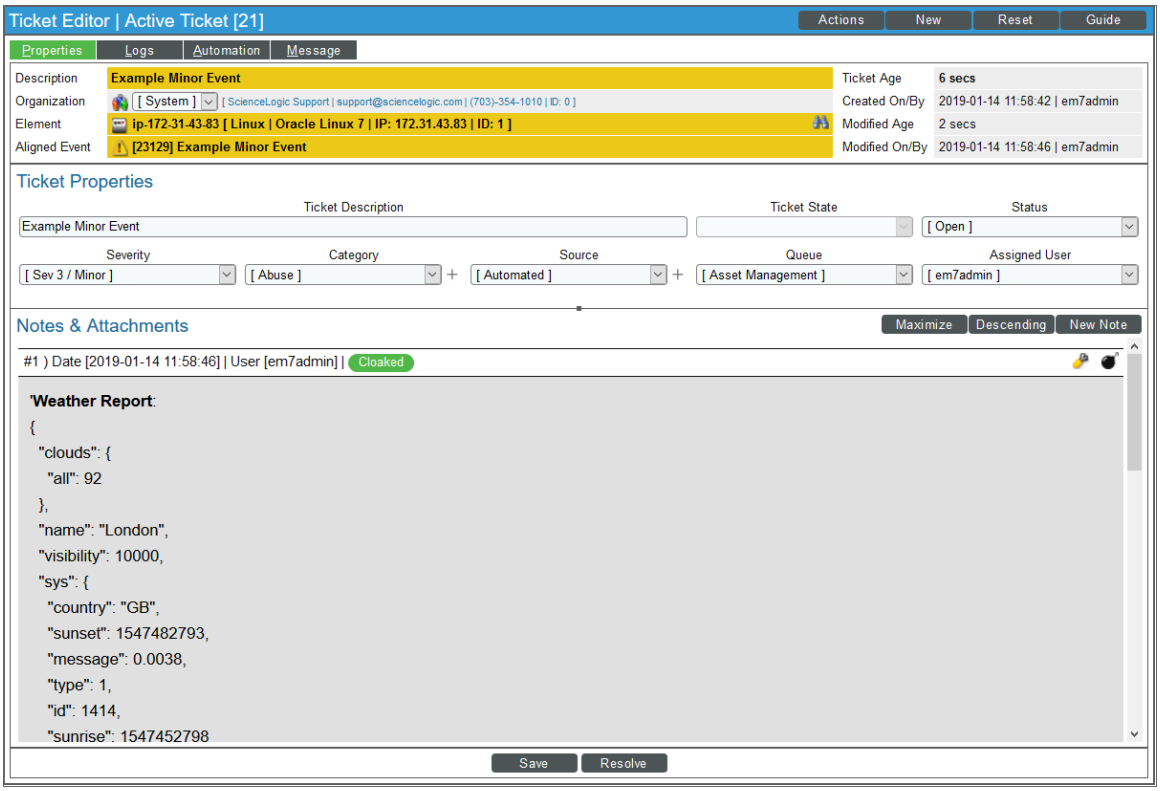


3. The automation policy we created for this example triggers only when a ticket exists for the event. Click the ticket icon () to create a ticket. The **Ticket Editor** page is displayed:

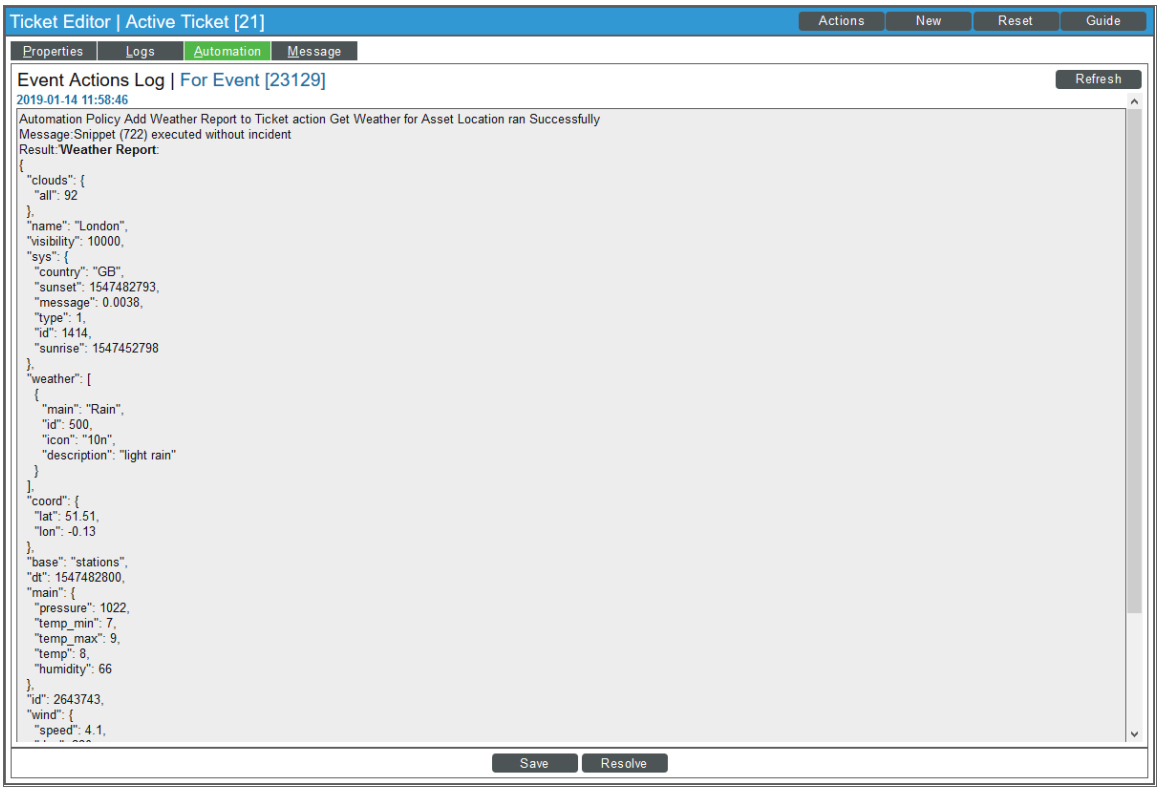


4. Click **[Save]** in the Ticket Editor to create the ticket.

5. The automation policy now matches and will trigger for the event. Click the **[Reset]** button in the **Ticket Editor** page to refresh the page. When the actions associated with the automation policy have completed, the weather report will appear in a new note:



6. To view the automation logs for the event, click the **[Automation]** tab:



Possible Next Steps

This example is intended to demonstrate the basic capabilities of the snippet action type with the smallest amount of code possible. You could expand the functionality of this example by making the following changes:

- This example assumes that the values in the "Facility/Data Center" field of the asset record are valid location names for the weather API. If your organization has already defined values in the "Facility/Data Center" field that are not valid location names, you could add a lookup function to translate the "Facility/Data Center" values to a location name.
- This example applies minimal formatting to the JSON response from the API. You could include additional processing of the API response, such as: adding human-readable labels for each field, specifying units, and removing data that is not relevant to the end user. If multiple formatting options are required, you could output the raw JSON response from the action and add separate actions that format the response.

© 2003 - 2026, ScienceLogic, Inc.

All rights reserved.

ScienceLogic™, the ScienceLogic logo, and ScienceLogic's product and service names are trademarks or service marks of ScienceLogic, Inc. and its affiliates. Use of ScienceLogic's trademarks or service marks without permission is prohibited.

ALL INFORMATION AVAILABLE IN THIS GUIDE IS PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED. SCIENCELOGIC™ AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

Although ScienceLogic™ has attempted to provide accurate information herein, the information provided in this document may contain inadvertent technical inaccuracies or typographical errors, and ScienceLogic™ assumes no responsibility for the accuracy of the information. Information may be changed or updated without notice. ScienceLogic™ may also make improvements and / or changes in the products or services described herein at any time without notice.



800-SCI-LOGIC (1-800-724-5644)

International: +1-703-354-1010