

Building Modular Alerts w/ Add-On Builder

Micah Kemp - 2017-02-13

<https://github.com/micahkemp>

Modular Alerts - What are they?

Used to perform customized actions from events returned by savedsearches

What good is that?!

Send alerts to Slack/Hipchat/PagerDuty/whatever

Supplement available data once deemed important (reverse DNS, for instance)

Open tickets in your favorite ticketing system

Sounds great, but how hard is it?

Splunk Add-on Builder makes it very easy to get started.

Why bother putting more here, I'll just show you!

<https://splunkbase.splunk.com/app/2962/>

Create Add-on



* Add-on Name:

Demo

Author:

Micah Kemp

Add-on Folder Name: **TA-demo**

[Edit](#)

Version:

1.0.0

Description:

TA for demo

Visible:

☐

Icon:



Upload my icon

No file selected

Theme Color:



Cancel

Create



Create Alert Actions

Click to create alert actions.

Alert Action Properties

Enter the properties for this alert action, and choose whether to use the Adaptive Response feature of Splunk Enterprise Security. [Learn more](#) 

*Label

*Name

Description

Logo



Upload my icon

No file selected



presentation_action

*Ticket Title



Assign To



Property Editor

Text Input

Display label

Ticket Title

Internal name

ticket_title

Default value

optional

Help text

Optional. Max 200 characters

☒ Required

*Ticketing Username

×

*Ticketing Password

×

Property Editor

Password

Display label

Ticketing Password

Internal name

ticketing_password

Default value

Help text

Optional. Max 200 characters

☒ Required


```

1
2 # encoding = utf-8
3 # Always put this line at the beginning of this file
4 import ta_demo_declare
5
6 import os
7 import sys
8
9 from alert_actions_base import ModularAlertBase
10 import modalert_presentation_action_helper
11
12 class AlertActionWorkerpresentation_action(ModularAlertBase):
13
14     def __init__(self, ta_name, alert_name):
15         super(AlertActionWorkerpresentation_action, self).__init__(ta_name, alert_name)
16
17     def validate_params(self):
18
19         if not self.get_global_setting("ticketing_username"):
20             self.log_error('ticketing_username is a mandatory setup parameter, but its value is None.')
21             return False
22
23         if not self.get_global_setting("ticketing_password"):
24             self.log_error('ticketing_password is a mandatory setup parameter, but its value is None.')
25             return False
26
27         if not self.get_param("ticket_title"):
28             self.log_error('ticket_title is a mandatory parameter, but its value is None.')
29             return False
30         return True
31
32     def process_event(self, *args, **kwargs):
33         status = 0
34         try:
35             if not self.validate_params():
36                 return 3
37             status = modalert_presentation_action_helper.process_event(self, *args, **kwargs)
38         except (AttributeError, TypeError) as ae:
39             self.log_error("Error: {}. Please double check spelling and also verify that a compatible version of Splunk_SA_CIM is installed.".format(ae.message))
40             return 4
41         except Exception as e:
42             msg = "Unexpected error: {}."
43             if e.message:
44                 self.log_error(msg.format(e.message))
45             else:
46                 import traceback
47                 self.log_error(msg.format(traceback.format_exc()))
48             return 5
49         return status
50
51 if __name__ == "__main__":
52     exitcode = AlertActionWorkerpresentation_action("TA-demo", "presentation_action").run(sys.argv)
53     sys.exit(exitcode)

```

```

1
2 # encoding = utf-8
3
4 def process_event(helper, *args, **kwargs):
5     """
6     # IMPORTANT
7     # Do not remove the anchor macro:start and macro:end lines.
8     # These lines are used to generate sample code. If they are
9     # removed, the sample code will not be updated when configurations
10    # are updated.
11
12    [sample_code_macro:start]
13
14    # The following example gets the setup parameters and prints them to the log
15    ticketing_username = helper.get_global_setting("ticketing_username")
16    helper.log_info("ticketing_username={}".format(ticketing_username))
17    ticketing_password = helper.get_global_setting("ticketing_password")
18    helper.log_info("ticketing_password={}".format(ticketing_password))
19
20    # The following example gets the alert action parameters and prints them to the log
21    ticket_title = helper.get_param("ticket_title")
22    helper.log_info("ticket_title={}".format(ticket_title))
23
24    assign_to = helper.get_param("assign-to")
25    helper.log_info("assign_to={}".format(assign_to))
26
27
28    # The following example adds two sample events ("hello", "world")
29    # and writes them to Splunk
30    # NOTE: Call helper.writeevents() only once after all events
31    # have been added
32    helper.addevent("hello", sourcetype="sample_sourcetype")
33    helper.addevent("world", sourcetype="sample_sourcetype")
34    helper.writeevents(index="summary", host="localhost", source="localhost")
35
36    # The following example gets the events that trigger the alert
37    events = helper.get_events()
38    for event in events:
39        helper.log_info("event={}".format(event))
40
41    # helper.settings is a dict that includes environment configuration
42    # Example usage: helper.settings["server_uri"]
43    helper.log_info("server_uri={}".format(helper.settings["server_uri"]))
44    [sample_code_macro:end]
45    """
46
47    helper.log_info("Alert action presentation_action started.")
48
49    # TODO: Implement your alert action logic here
50    return 0

```

Configuration

Set up your add-on

Add-on Settings

Ticketing Username *

Ticketing Password *

Save

Trigger Actions

+ Add Actions ▾

When triggered



Presentation Action

Remove

Ticket
Title *

Demo Alert Triggered

Assign To

|

```

1
2 # encoding = utf-8
3
4 def process_event(helper, *args, **kwargs):
5
6     # The following example gets the setup parameters and prints them to the log
7     ticketing_username = helper.get_global_setting("ticketing_username")
8     helper.log_info("ticketing_username={}".format(ticketing_username))
9     ticketing_password = helper.get_global_setting("ticketing_password")
10    helper.log_info("ticketing_password={}".format(ticketing_password))
11
12    # The following example gets the alert action parameters and prints them to the log
13    ticket_title = helper.get_param("ticket_title")
14    helper.log_info("ticket_title={}".format(ticket_title))
15
16    assign_to = helper.get_param("assign_to")
17    helper.log_info("assign_to={}".format(assign_to))
18
19
20    # The following example adds two sample events ("hello", "world")
21    # and writes them to Splunk
22    # NOTE: Call helper.writeevents() only once after all events
23    # have been added
24    helper.addevent("hello", sourcetype="sample_sourcetype")
25    helper.addevent("world", sourcetype="sample_sourcetype")
26    helper.writeevents(index="summary", host="localhost", source="localhost")
27
28    # The following example gets the events that trigger the alert
29    events = helper.get_events()
30    for event in events:
31        helper.log_info("event={}".format(event))
32
33    # helper.settings is a dict that includes environment configuration
34    # Example usage: helper.settings["server_uri"]
35    helper.log_info("server_uri={}".format(helper.settings["server_uri"]))
36
37    helper.log_info("Alert action presentation_action started.")
38
39    # TODO: Implement your alert action logic here
40    return 0

```

2018-02-09 20:24:01,934 INFO pid=47341 tid=MainThread file=cim_actions.py:message:271 |
sendmodaction - signature="ticketing_username=AzureDiamond"
action_name="presentation_action" search_name="Demo Alert"
sid="scheduler_admin_search_RMD57a1631503e768f3b_at_1518207840_609" rid="0" app="search"
user="admin" action_mode="saved" action_status="success"

2018-02-09 20:24:01,934 INFO pid=47341 tid=MainThread file=cim_actions.py:message:271 |
sendmodaction - signature="ticketing_password=hunter2" action_name="presentation_action"
search_name="Demo Alert"
sid="scheduler_admin_search_RMD57a1631503e768f3b_at_1518207840_609" rid="0" app="search"
user="admin" action_mode="saved" action_status="success"

2018-02-09 20:24:01,934 INFO pid=47341 tid=MainThread file=cim_actions.py:message:271 |
sendmodaction - signature="ticket_title=Demo Alert Triggered"
action_name="presentation_action" search_name="Demo Alert"
sid="scheduler_admin_search_RMD57a1631503e768f3b_at_1518207840_609" rid="0" app="search"
user="admin" action_mode="saved" action_status="success"

2018-02-09 20:24:01,934 INFO pid=47341 tid=MainThread file=cim_actions.py:message:271 |
sendmodaction - signature="assign_to=" action_name="presentation_action" search_name="Demo
Alert" sid="scheduler_admin_search_RMD57a1631503e768f3b_at_1518207840_609" rid="0"
app="search" user="admin" action_mode="saved" action_status="success"

```
2018-02-09 20:24:01,935 INFO pid=47341 tid=MainThread file=cim_actions.py:message:271 |
sendmodaction - signature="event={'count': '344', 'rid': '0', 'sourcetype': 'audittrail',
'__mv_sourcetype': '', '__mv_count': ''}" action_name="presentation_action"
search_name="Demo Alert"
sid="scheduler__admin__search__RMD57a1631503e768f3b_at_1518207840_609" rid="0" app="search"
user="admin" action_mode="saved" action_status="success"
```

```
2018-02-09 20:24:01,935 INFO pid=47341 tid=MainThread file=cim_actions.py:message:271 |
sendmodaction - signature="event={'count': '84', 'rid': '1', 'sourcetype': 'gdf',
'__mv_sourcetype': '', '__mv_count': ''}" action_name="presentation_action"
search_name="Demo Alert"
sid="scheduler__admin__search__RMD57a1631503e768f3b_at_1518207840_609" rid="1" app="search"
user="admin" action_mode="saved" action_status="success"
```

```
2018-02-09 20:24:01,935 INFO pid=47341 tid=MainThread file=cim_actions.py:message:271 |
sendmodaction - signature="event={'count': '76', 'rid': '2', 'sourcetype': 'kvstore',
'__mv_sourcetype': '', '__mv_count': ''}" action_name="presentation_action"
search_name="Demo Alert"
sid="scheduler__admin__search__RMD57a1631503e768f3b_at_1518207840_609" rid="2" app="search"
user="admin" action_mode="saved" action_status="success"
```

A little more than the defaults may be needed

Add to `bin/ta_demo/modalert_presentation_action_helper.py`:

```
# show additional info available about the generating search
```

```
helper.addinfo()
```

```
helper.log_info("info={}".format(helper.info))
```

```
2018-02-10 06:10:02,778 INFO pid=58847 tid=MainThread file=cim_actions.py:message:271 |
sendmodaction - signature="info={ ... '_search_lt': '1518242940.000000000', '_search_et':
'1518242040.000000000' ... }" action_name="presentation_action" search_name="Demo Alert"
sid="scheduler__admin__search__RMD57a1631503e768f3b_at_1518242940_2811" rid="10"
app="search" user="admin" action_mode="saved" action_status="success"
```