# 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

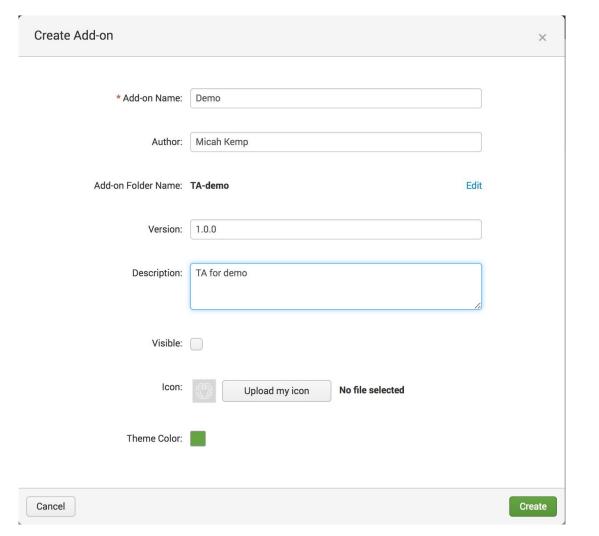
Supplement available data once deemed important (reverse DNS, for instance)

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!



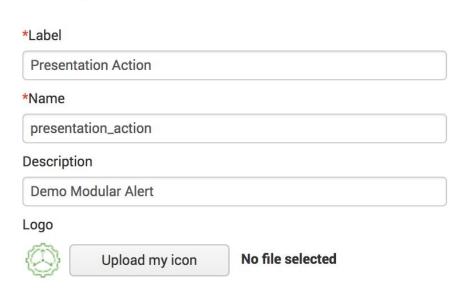


## **Create Alert Actions**

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





#### presentation\_action

*Ticket Title	×
Assign To	×
	1
	1 1 1 1 1

Property Editor
Text Input
Display label
Ticket Title
Internal name
ticket_title
Default value
optional
Help text
Optional. Max 200 characters
✓ Required

	Property Editor
*Ticketing Username	× Password
	Display label
*Ticketing Password	Ticketing Password
	Internal name
	ticketing_password
	Default value
	Help text
	Optional. Max 200 characters
	Required

```
2 # encoding = utf-8
3 # Always put this line at the beginning of this file
 4 import ta_demo_declare
 6 import os
 7 import sys
 9 from alert_actions_base import ModularAlertBase
 10 import modalert_presentation_action_helper
12 class AlertActionWorkerpresentation_action(ModularAlertBase):
       def __init__(self, ta_name, alert_name):
           super(AlertActionWorkerpresentation_action, self).__init__(ta_name, alert_name)
       def validate_params(self):
          if not self.get_global_setting("ticketing_username"):
               self.log_error('ticketing_username is a mandatory setup parameter, but its value is None.')
               return False
          if not self.get_global_setting("ticketing_password"):
    self.log_error('ticketing_password is a mandatory setup parameter, but its value is None.')
               return False
           if not self.get_param("ticket_title"):
               self.log_error('ticket_title is a mandatory parameter, but its value is None.')
               return False
           return True
       def process_event(self, *args, **kwargs):
           status = 0
               if not self.validate_params():
               status = modalert_presentation_action_helper.process_event(self, *args, **kwargs)
           except (AttributeError, TypeError) as ae:
               self.log_error("Error: {}. Please double check spelling and also verify that a compatible version of Splunk_SA_CIM is installed.".format(ae.message))
               return 4
           except Exception as e:
               msg = "Unexpected error: {}."
               if e.message:
                   self.log_error(msg.format(e.message))
               else:
                   import traceback
                   self.log_error(msg.format(traceback.format_exc()))
               return 5
           return status
51 if __name__ == "__main__":
       exitcode = AlertActionWorkerpresentation_action("TA-demo", "presentation_action").run(sys.argv)
       sys.exit(exitcode)
```

```
4 def process_event(helper, *args, **kwargs):
        # IMPORTANT
        # Do not remove the anchor macro:start and macro:end lines.
        # These lines are used to generate sample code. If they are
        # removed, the sample code will not be updated when configurations # are updated.
         [sample_code_macro:start]
        # The following example gets the setup parameters and prints them to the log
ticketing_username = helper.get_global_setting("ticketing_username")
helper.log_info("ticketing_username={}".format(ticketing_username))
ticketing_password = helper.get_global_setting("ticketing_password")
         helper.log_info("ticketing_password={}".format(ticketing_password))
18
         # The following example gets the alert action parameters and prints them to the log
ticket_title = helper.get_param("ticket_title")
         helper.log_info("ticket_title={}".format(ticket_title))
         assign_to = helper.get_param("assign_to")
helper.log_info("assign_to={}".format(assign_to))
         # The following example adds two sample events ("hello", "world")
         # and writes them to Splunk
         # NOTE: Call helper.writeevents() only once after all events
        # have been added
helper.addevent("hello", sourcetype="sample_sourcetype")
helper.addevent("world", sourcetype="sample_sourcetype")
helper.writeevents(index="summary", host="localhost", source="localhost")
         # The following example gets the events that trigger the alert
events = helper.get_events()
         for event in events:
39
               helper.log_info("event={}".format(event))
40
         # helper.settings is a dict that includes environment configuration
# Example usage: helper.settings["server_uri"]
41
42
         helper.log_info("server_uri={}".format(helper.settings["server_uri"]))
43
          [sample_code_macro:end]
44
45
46
         helper.log_info("Alert action presentation_action started.")
```

# TODO: Implement your alert action logic here

2 # encoding = utf-8

splunk> App: Demo ∨ Configuration Search Configuration Set up your add-on Add-on Settings Ticketing Username \* AzureDiamond Ticketing Password \* ...... Save

# **Trigger Actions** + Add Actions > When triggered Presentation Action Remove **Ticket** Demo Alert Triggered Title \* Assign To

```
2 # encoding = utf-8
4 def process_event(helper, *args, **kwargs):
     # The following example gets the setup parameters and prints them to the log
     ticketing_username = helper.get_global_setting("ticketing_username")
     helper.log_info("ticketing_username={{}}".format(ticketing_username))
     ticketing_password = helper.get_global_setting("ticketing_password")
     helper.log_info("ticketing_password={}".format(ticketing_password))
     # The following example gets the alert action parameters and prints them to the log
     ticket_title = helper.get_param("ticket_title")
     helper.log_info("ticket_title={}".format(ticket_title))
     assign_to = helper.get_param("assign_to")
     helper.log_info("assign_to={}".format(assign_to))
     # The following example adds two sample events ("hello", "world")
     # and writes them to Splunk
     # NOTE: Call helper.writeevents() only once after all events
     # have been added
     helper.addevent("hello", sourcetype="sample_sourcetype")
     helper.addevent("world", sourcetype="sample_sourcetype")
     helper.writeevents(index="summary", host="localhost", source="localhost")
     # The following example gets the events that trigger the alert
     events = helper.get_events()
     for event in events:
         helper.log_info("event={}".format(event))
     # helper.settings is a dict that includes environment configuration
     # Example usage: helper.settings["server_uri"]
     helper.log_info("server_uri={}".format(helper.settings["server_uri"]))
     helper.log_info("Alert action presentation_action started.")
```

# TODO: Implement your alert action logic here

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"
```