



.conf2015

# Creating and Using Custom Alert Actions

Nicholas Filippi

Product Management, Splunk

Siegfried Puchbauer

Client Engineering, Splunk



splunk®

# Disclaimer

During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC. The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not, be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

# Who We Are



Nicholas Filippi

Product Management  
**splunk>**



Siegfried Puchbauer

Client Engineering  
**splunk>**

# Agenda

- Alert Actions Framework Overview
- Demo
- Live Coding – Build alert action from scratch

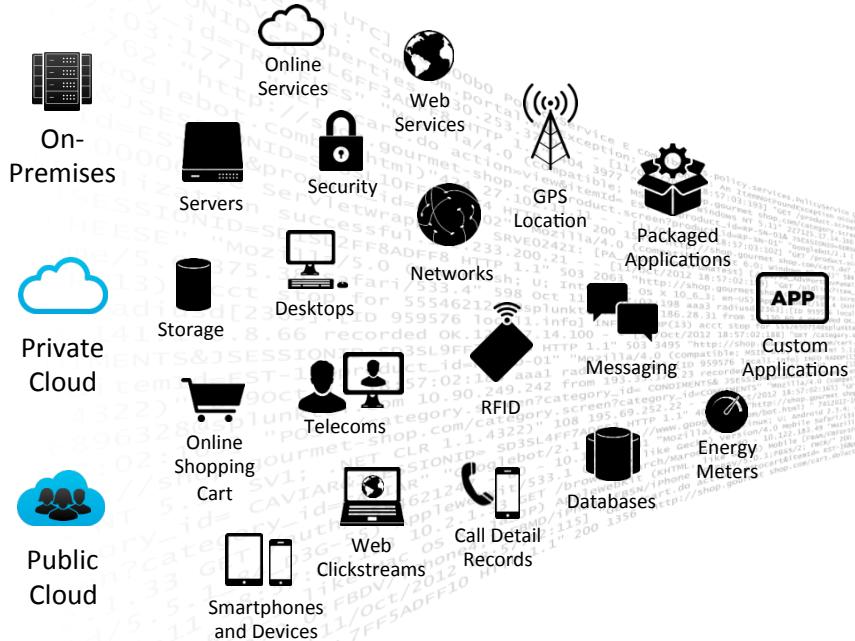
**splunk®**

# Make machine data accessible, usable and valuable to everyone.

# Turning Machine Data Into Business Value

Index untapped data: any source, type, volume

Ask any question



splunk>

Application Delivery

IT Operations

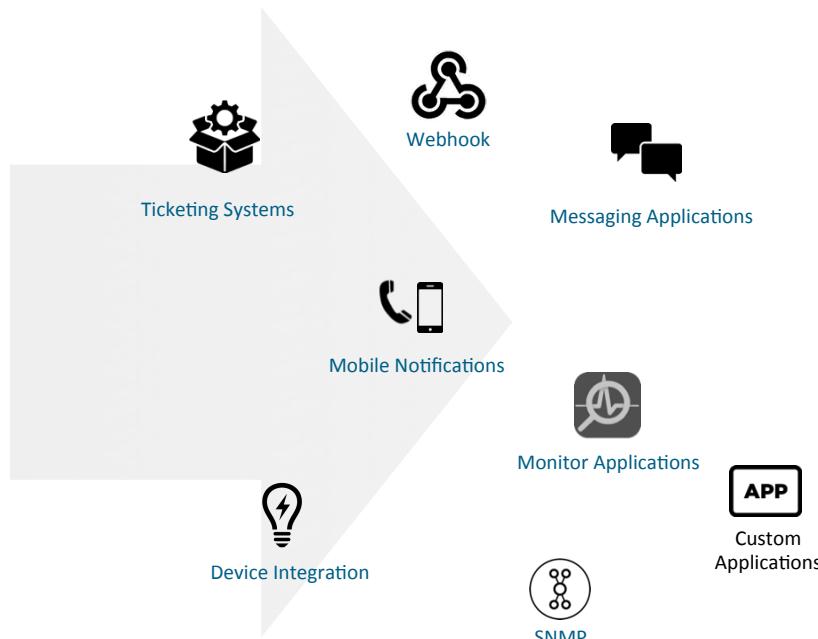
Security, Compliance  
and Fraud

Business Analytics

Industrial Data and  
the Internet of Things

# Turning Machine Data Into Business Value

Integrate with other applications to automate workflows and improve efficiencies



Application Delivery

IT Operations

Security, Compliance  
and Fraud

Business Analytics

Industrial Data and  
the Internet of Things

# Custom Alert Actions

*Use Splunk Alerts to trigger & automate workflows*

- Allows packaged integration with third-party applications
- Simple admin/user configuration
- Developers can build, package, and publish alert action extensions for native integration to Splunk
- Growing list of integrations available

+ Add Actions ▾

 HipChat	Send HipChat room notifications
 Run a script	Invoke a custom script
 Send email	Send an email notification to specified recipients
 ServiceNow	Create a ticket in ServiceNow
 Slack	Send a message to a Slack channel
 Webhook	Generic HTTP POST to a specified URL

# Alert Action Examples

- Notification Services
  - Send message to IM clients (HipChat, Slack)
  - Send SMS
- Incident Remediation/Ticketing
  - Automate the creation of tickets (ServiceNow, Jira)
- IT Monitoring
  - Send incident/alert into monitoring tools (xMatters, BigPanda)
- Security
  - Take action or send events to firewalls, devices, management consoles
- Internet-of-Things
  - Trigger device-level actions (change lights, sounds an alarm, send action to device)
- Custom Action
  - Trigger any organization-specific action (restart application, integrate with homegrown service, and more)

Eco-system Partners

(x) matters®

service**now**

 octoblu  
Now a part of Citrix

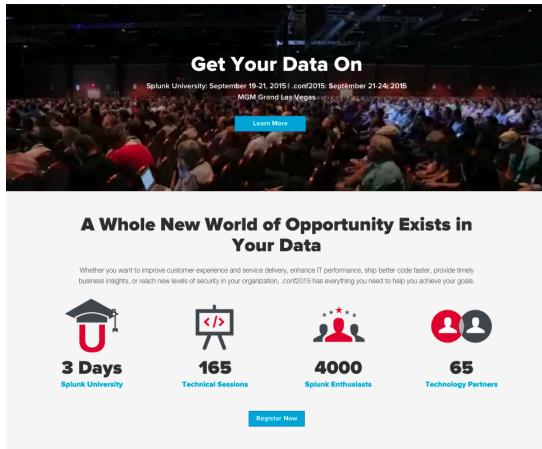
 bigpanda

 twilio

 splunk>

# Real-World Example

## Data Source



Monitoring mobile accelerometer data during keynote presentation

## Search & Trigger

Query:

```
index=main sourcetype=load  
| stats avg(load) as avg_load  
| search avg_load > 89
```

10

## Alert Action



Trigger an air cannon to fire Splunk ponies into the crowd

# Alert Action Framework

<input checked="" type="checkbox"/>	<b>Packaging</b>	Package and distribute custom alert actions within apps
<input checked="" type="checkbox"/>	<b>UI Integration</b>	Build UI to support user input parameters within the alert workflow
<input checked="" type="checkbox"/>	<b>Management</b>	Admin-level management to enable/disable, view usage stats, and more
<input checked="" type="checkbox"/>	<b>Permissions</b>	Access controls (configure role-level restrictions/permissions to access these alert actions)
<input checked="" type="checkbox"/>	<b>Logging</b>	Logs both Splunkd process level as well as script-level to internal index
<input checked="" type="checkbox"/>	<b>Input Validation</b>	Perform user input validation on save of a given alert
<input checked="" type="checkbox"/>	<b>Dynamic Parameter Support (w/ token substitution)</b>	Define parameters to be passed on script invocation; available token substitution for 1 <sup>st</sup> result and search/job/server endpoints

# Alert Action Framework

✓	<b>Multi-Platform Compatibility</b>	Package multiple scripts to run based on platform type (Windows, Linux)
✓	<b>Secured Storage of Credentials</b>	Encrypt sensitive credentials on disk, with access methods to read/write from your alert script
✓	<b>Ad-Hoc Invocation</b>	Invoke directly from query string or workflow actions for testing or targeted use cases
✓	<b>Setup/Configuration</b>	Leverage the app setup.xml to persist global-level script parameters. These parameters can also be layered with invocation-specific parameters
✓	<b>Runtime Arguments (Language Agnostic)</b>	Specify runtime parameters (ex., java interpreter and argument flags)

# Demo!





# .conf2015

## Alert Action: From Scratch

splunk®

# Resources

*Available information relevant for developers and admins*

- Splunk Docs
  - <http://docs.splunk.com/Documentation/Splunk/6.3.0/>  
[AdvancedDev/ModAlertsIntro](#)
  - <http://docs.splunk.com/Documentation/Splunk/6.3.0/>  
[Alert/Setupalertactions#Webhooks](#)
- Splunkbase
  - Partner Apps
    - xMatters, BigPanda, Twilio, ServiceNow, Octoblu
  - Internal
    - Hipchat, Slack, Hue Bulbs
- Developer Guidance
  - Updated chapter on alert actions
  - Reference implementation – Atlassian Jira Integration

The screenshot displays the Splunkbase homepage, which is a platform for finding and sharing Splunk apps and add-ons. The top navigation bar includes links for 'splunk>docs', 'splunkbase', 'CATORIES', 'TECHNOLOGIES', and 'FOR DEVELOPERS'. The main header reads 'Welcome to Splunk Documentation' with a sub-header 'Extend the power of Splunk'. Below this, there's a search bar and a 'SEARCH' button. The page features several cards showcasing different Splunk apps:

- Splunk DB Connect 2**: A card for the Splunk DB Connect 2 app, which integrates structured data from databases into Splunk. It includes a 'Splunk CERTIFIED' badge.
- Kepware Explorer**: A card for the Kepware Explorer app, which integrates data from various industrial protocols into Splunk. It also has a 'Splunk CERTIFIED' badge.
- Splunk App for Unix and Linux**: A card for the Splunk App for Unix and Linux, which provides operational visibility into large-scale Unix and Linux environments. It includes a 'Splunk INTELLIGENCE' badge.
- Splunk App Certification Program**: A card for the Splunk App Certification Program, which helps organizations validate the security and performance of their apps. It includes a 'Splunk INTELLIGENCE' badge.

At the bottom, there are sections for 'BROWSE BY CATEGORY', 'NEWEST', 'View All', 'POPULAR', and 'View All' again. The 'POPULAR' section lists several apps with their names and download counts:

App	Downloads
Kepware Explorer	367
Splunk App for Unix and Linux	337
Gileman Visibility App For Splunk	56
Splunk Sa App for Microsoft	746

# Wrap-Up

- Use Splunk Alerts to Automate Workflows & Integrate with External Applications
  - Notification/Messaging Services, Incident Remediation (Ticketing) Solutions, IT Monitoring Tools, Security Solutions, Internet-of-Things Devices, and Custom Applications
- Use Custom Alert Actions via the “Add Actions” scheduled alert menu
- Manage Alert Actions via “Settings > Alert actions” page
  - Find more alert actions to install via “Browse more”
- Leverage Docs/Examples/Developer-Guidance to Develop a Custom Alert Action

.conf2015

# THANK YOU

**splunk®**



# .conf2015

# Developer Guide



splunk®

# Build a New Alert Action

## Steps

1. Alert action definition (alert\_actions.conf)
2. Alert action script (in <app>/bin)
3. UI for configuring the alert action (HTML)
4. Spec for custom parameters
5. Optionally:
  - Icon
  - Validation rules
  - App setup



# Example Alert Action



- Atlassian Hipchat (Group Chat)
- API v2 – Send Room Notifications
  - [https://www.hipchat.com/docs/apiv2/method/send\\_room\\_notification](https://www.hipchat.com/docs/apiv2/method/send_room_notification)
- Custom Alert Action – Includes:
  - App setup page to specify Hipchat server URL & API Token
  - For each alert, user should be able to configure
    - Room
    - Message
    - Format (plaintext, html)
    - Message color
    - Notify users?

The screenshot shows the 'HipChat Alerts' configuration page in the Splunk web interface. The 'Server' section contains fields for 'Server Base URL' (set to https://hipchat.splunk.com/v2/) and 'API Token' (set to a long string of characters). A 'Save' button is visible at the bottom right.

The screenshot shows the 'Trigger Actions' configuration page. Under 'When triggered', there is a section for 'HipChat' with 'Room' set to 'Failed Login Attempts'. The 'Message' section contains a template message: 'Failed login attempts to wimpy:15000; user="\$result.user\$" and reason="\$result.reason\$" and client\_ip=\$result.clientip\$'. Below this, 'Message Format' is set to 'Plain Text', 'Message Color' is 'Red', and 'Notify users in the room' is checked. An 'Auth Token' field is also present. To the right, there are instructions for naming the HipChat room and sending the message, along with a 'More' link.

# Registration

default/alert\_actions.conf

```
[hipchat]  
  
is_custom = 1  
  
label = HipChat  
description = Send HipChat room notifications  
icon_path = hipchat_alert_icon.png  
  
payload_format = json  
  
disabled = 0  
  
param.base_url = https://api.hipchat.com/v2/  
param.auth_token =
```



Trigger Actions

+ Add Actions ▾

- HipChat Send HipChat room notifications
- List in Triggered Alerts Make available in triggered alerts
- Run a script Invoke a custom script
- Send Email Send an email notification to specified recipients
- ServiceNow Create a ticket in ServiceNow
- Webhook Generic HTTP POST to a specified URL

# Alert Script or Binary

- <app>/bin/<name>
  - **bin/hipchat.py**
- Carries out the actual alert action logic
- Receives XML or JSON payload from Splunk when invoked
- Supposed to be short-lived
  - Terminate as soon as action has been executed
- Python, shell scripts or binaries
- Multi-platform/architecture support

# Execute a Custom Command

- Execution command/arguments
  - Override default filename/path
  - Specify custom interpreter
  - Pass args

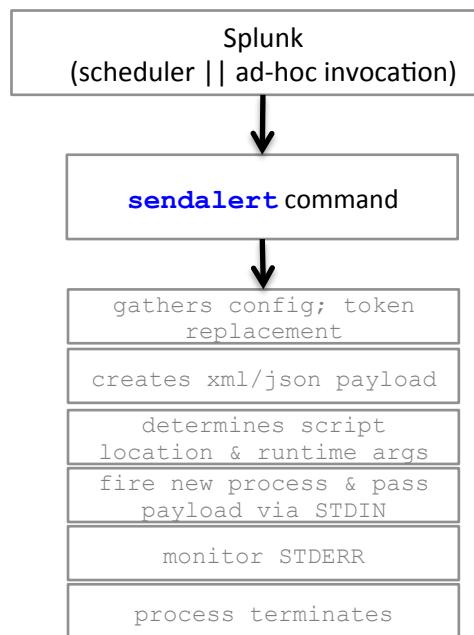
## **default/alert\_actions.conf**

```
[myaction]
...
alert.execute.cmd = java.path
alert.execute.cmd.arg.0 = -jar
alert.execute.cmd.arg.1 = $SPLUNK_HOME/etc/apps/myapp/bin/my.jar
alert.execute.cmd.arg.2 = --execute
```

# Multi-Platform Compatibility

- Determines the alert script or binary to execute using the below precedence
  - \$SPLUNK\_HOME/etc/apps/<app\_name>/<arch>/bin
    - (arch : linux\_x86\_64, darwin\_x86\_64, windows\_x86\_64, windows\_x86)
  - \$SPLUNK\_HOME/etc/apps/<app\_name>/bin

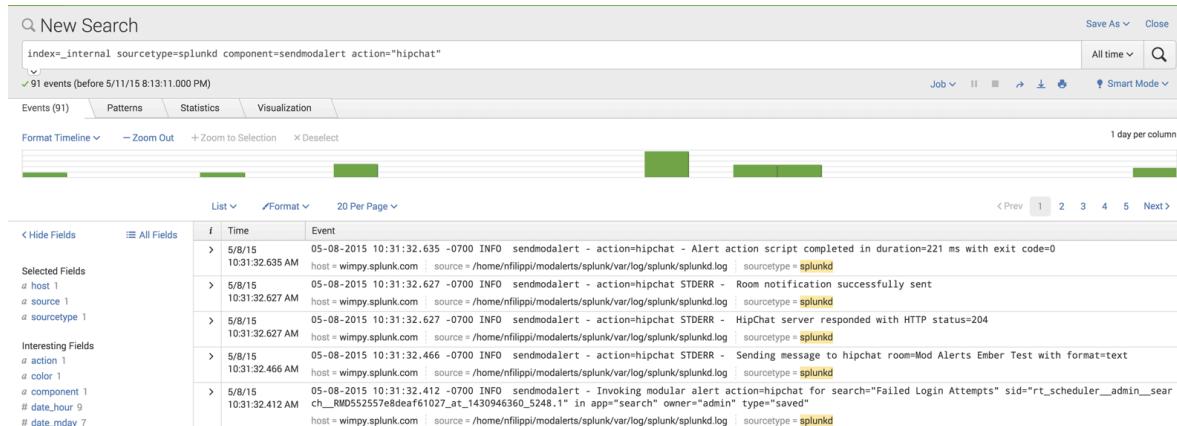
# Script Invocation (Example)



```
{  
    "server_host": "localhost:8089",  
    "server_uri": "https://localhost:8089",  
    "session_key": "1234512345",  
    "results_file": "/opt/splunk/var/run/splunk/12938718293123.121/results.csv.gz",  
    "results_link": "http://splunk.server.local:8000/en-US/app/search?sid=12341234.123",  
    "sid": "12341234.123",  
    "search_name": "My Saved Search",  
    "owner": "admin",  
    "app": "search",  
  
    "configuration": {  
        "room": "DevOps",  
        "message": "Failed login attempts to wimpy:15000; user='Jeff' and reason='user-initiated'  
and client_ip='10.14.0.186'",  
        "message_format": "plain",  
        "base_url": "https://api.hipchat.com/v2/",  
        "auth_token": "KJHJKHJGJHLKNJBLJBBL"  
    },  
    "result": {  
        "sourcetype": "splunkd_access",  
        "count": "24",  
        "user": "Jeff",  
        "client_ip": "10.14.0.186"  
    }  
}
```

# Logging

- Messages printed to STDERR are logged to splunkd.log
- Direct access to alert action logs linked from “Alert actions” manager page (via “[view log events](#)”)
  - Includes both (a) splunkd logs for alert action, and (b) alert-specific log
- Search Query
  - `index=_internal sourcetype=splunkd component=sendmodalert action=<action_name>`



# UI Integration

## default/data/ui/alerts/hipchat.html

```
<!-- Defines UI to be rendered in the alert workflow (w/ data binding to savedsearch.conf params) -->
```

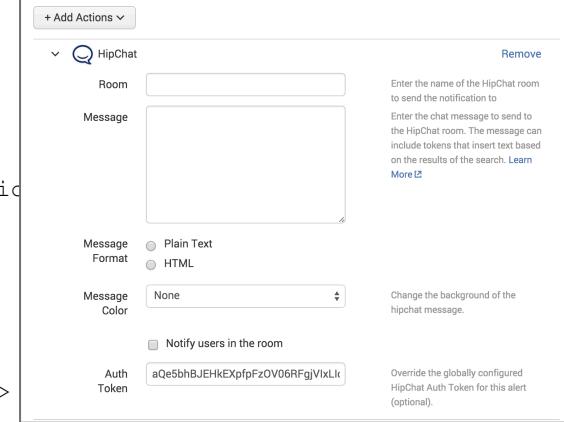
```
<form class="form-horizontal form-complex">
    <div class="control-group">
        <label class="control-label" for="hipchat_room">Room</label>

        <div class="controls">
            <input type="text" name="action.hipchat.param.room" id="hipchat_room" />
            <span class="help-block">Enter the name of the HipChat room to send the notification.</span>
        </div>
    </div>
    <div class="control-group">
        <label class="control-label" for="hipchat_message">Message</label>

        <div class="controls">
            <textarea name="action.hipchat.param.message" id="hipchat_message"></textarea>
            <span class="help-block">
                Enter the chat message to send to the HipChat room.
                The message can include tokens that insert text based on the results of the search.
                <a href="{{SPLUNKWEB_URL_PREFIX}}/help?location=learnmore.alert.action.tokens"
target=_blank
title="Splunk help">Learn More <i class="icon-external"></i></a>
            </span>
        </div>
    </div>
    ...

```

*Note: Currently, we only support static html (filter out any javascript). Markup should follow the bootstrap syntax (<http://getbootstrap.com/2.3.2/base-css.html#forms>)*



# Spec Files

## README/alert\_actions.conf.spec

```
# Defines parameters that to be added to  
the alert_actions.conf specification  
  
[hipchat]  
  
param.base_url = <string>  
* HipChat API base URL - adjust if you're  
using your own server on premise  
  
param.auth_token = <string>  
* HipChat OAuth2 token  
* see https://www.hipchat.com/docs/apiv2/auth
```

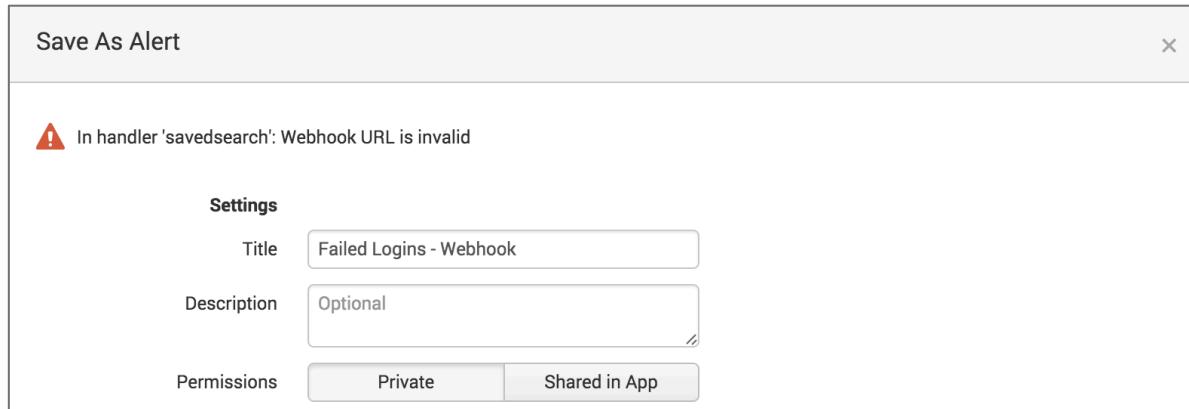
## README/savedsearches.conf.spec

```
# Defines parameters that to be added to the  
savedsearches.conf specification  
# HipChat alert settings  
  
action.hipchat = [0|1]  
* Enable hipchat notification  
action.hipchat.param.room = <string>  
* Name of the room to send the notification to  
action.hipchat.param.message = <string>  
* The message to send to the hipchat room.  
* * *
```

# Input Validation

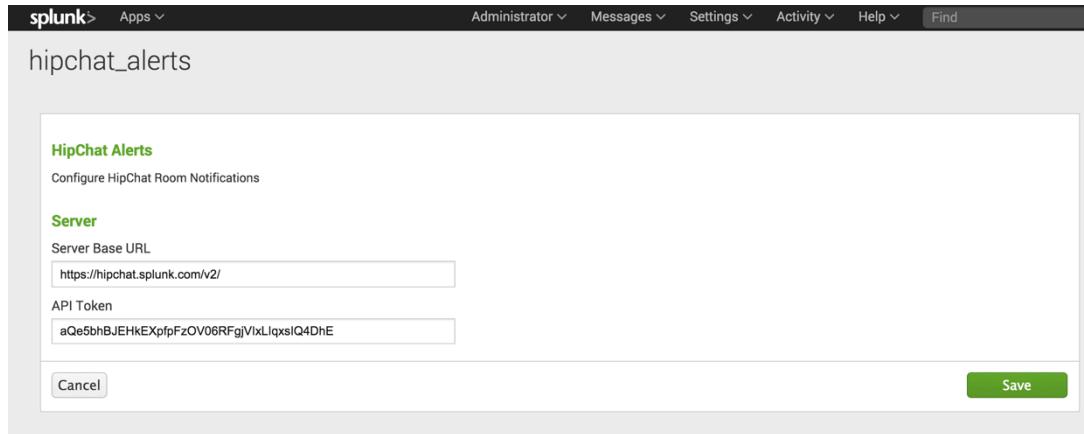
Available via restmap.conf

```
[validation:savedsearch]
action.webhook.param.url = validate( match('action.webhook.param.url',
    "^https?://[^\\s]+$"), "Webhook URL is invalid")
```



# Setup & Configuration

- Available via app-level setup.xml
  - <http://docs.splunk.com/Documentation/Splunk/6.3.0/AdvancedDev/SetupApp>
  - <http://docs.splunk.com/Documentation/Splunk/6.3.0/AdvancedDev/SetupXML>
- Linked from “Alert actions” manager page



# Secure Storage of Credentials

- Leverages Splunk's storage/passwords endpoint for CRUD operations
  - <http://docs.splunk.com/Documentation/Splunk/6.2.3/RESTREF/RESTaccess#storage.2Fpasswords>
- Note: This can only be used for global-level alert action settings (not per-alert)
- Required Steps:
  - Add entry to app.conf
    - (ex. "[credential::hipchat\_api\_token:]")
  - Update setup.xml to write to storage/passwords endpoint
    - (ex. '<input endpoint="storage/passwords" entity=":hipchat\_api\_token:" field="password">')
  - Update script to get secure credentials from splunkd endpoint

## default/bin/hipchat.py

```
...
def get_api_key(payload):
    url_tpl = '%s/servicesNS/nobody/hipchat_alerts/storage/passwords/%3Ahipchat_api_token%3A?output_mode=json'
    req = urllib2.Request(url_tpl % payload.get('server_uri'), None,
                          {'Authorization': 'Splunk %s' % payload.get('session_key')})
    res = urllib2.urlopen(req)
    body = json.loads(res.read())
    return body['entry'][0]['content']['clear_password']
...

```

# Access Control

- Packaged alerts actions should set to global
  - Via `metadata/default.meta`
- Enable role-level permissions via “Alert actions” manager page

The screenshot shows the Splunk Alert Actions manager. On the left, a list of alert actions is displayed, including 'Atlassian Jira', 'F5 Network Firewall Rule', 'HipChat', 'IBM Tivoli', 'Run a script', 'Send Email', 'ServiceNow', and 'Webhook'. On the right, a modal window titled 'Edit Permissions' is open for the 'hipchat' alert action. The modal has two tabs: 'Owner' and 'App'. Under 'Owner', the 'nobody' role is selected. Under 'App', the 'hipchat.alerts' app is selected. The 'Display For' dropdown is set to 'All apps'. The permissions table lists roles (Everyone, admin, can\_delete, power, splunk-system-role, user) and their access levels (Read, Write). The 'admin' role has 'Write' checked. Below the table, there are four rows of status and usage information:

Status	Usage	Log	Setup
Disabled   Enable	Usage statistics	View log events	
Disabled   Enable	Usage statistics	View log events	
Enabled   Disable	Usage statistics	View log events	Setup HipChat Room Notifications

# Packaging

- App can package more than 1 alert action
- Packaged files support
  - Registration as an alert action
  - Script itself
  - UI Integration
  - Configuring alert/search parameters
  - Permissions (exporting to global)
  - Default param values
  - Etc.

Example folder/file structure of an app shipping a mod alert action

```
hipchat_alerts
├── appserver
│   └── static
│       ├── appIcon.png
│       └── hipchat_icon.png
├── bin
│   └── hipchat.py
└── default
    ├── alert_actions.conf
    ├── app.conf
    ├── restmap.conf
    ├── setup.xml
    └── data
        └── ui
            └── alerts
                └── hipchat.html
├── README
└── metadata
    └── default.meta
```

.conf2015

# THANK YOU

**splunk®**