Modular Inputs – If You Build, They Will Come

Scott Haskell

Client Architect, 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 the 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

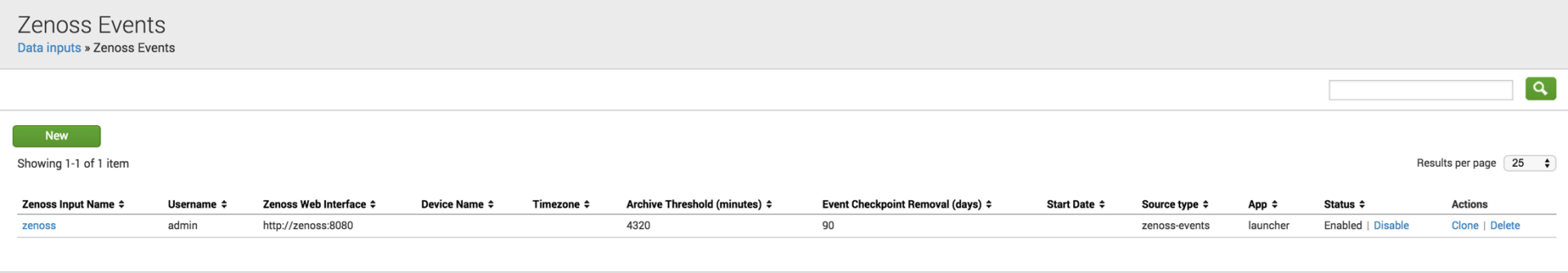
Agenda

- Building a modular input using the SDK by example - TA-zenoss

Modular Inputs 101

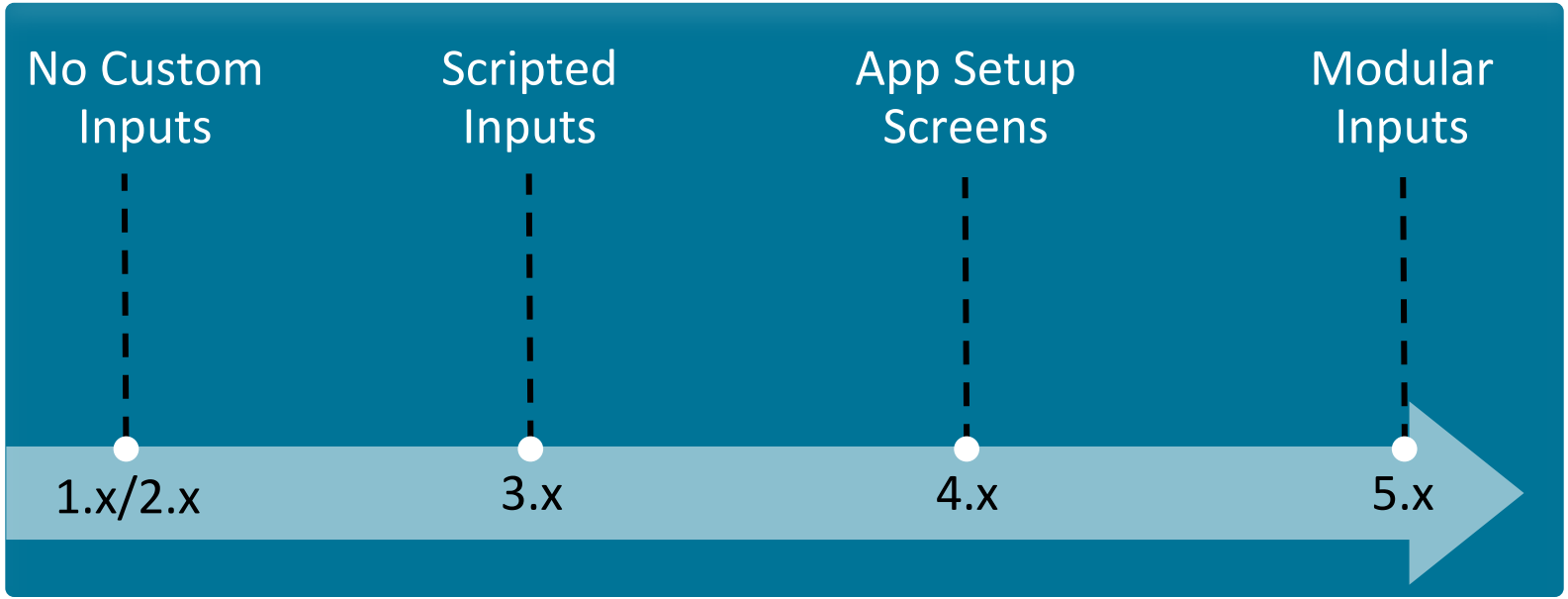
splunk>

What Is A Modular Input?



- Splunk Enterprise app or add-on that extends the Splunk Enterprise framework to define a custom input capability
- Treated as Splunk native input - **Settings > Data > Data Inputs**

Custom Data Input Timeline



Modular Inputs vs. Scripted Inputs

splunk>

Scripted Inputs vs. Modular Inputs

Capability	Modular Inputs	Scripted Inputs
UI Configuration & Validation	✔	
Permissioning	✔	
Checkpointing	✔	
Flexible Data Acquisition	✔	✔
SDK Support	✔	
Multi-Platform Support	✔	
Run As Splunk User	System User Only	✔
Custom REST Endpoints	✔	
Native Logging	✔	

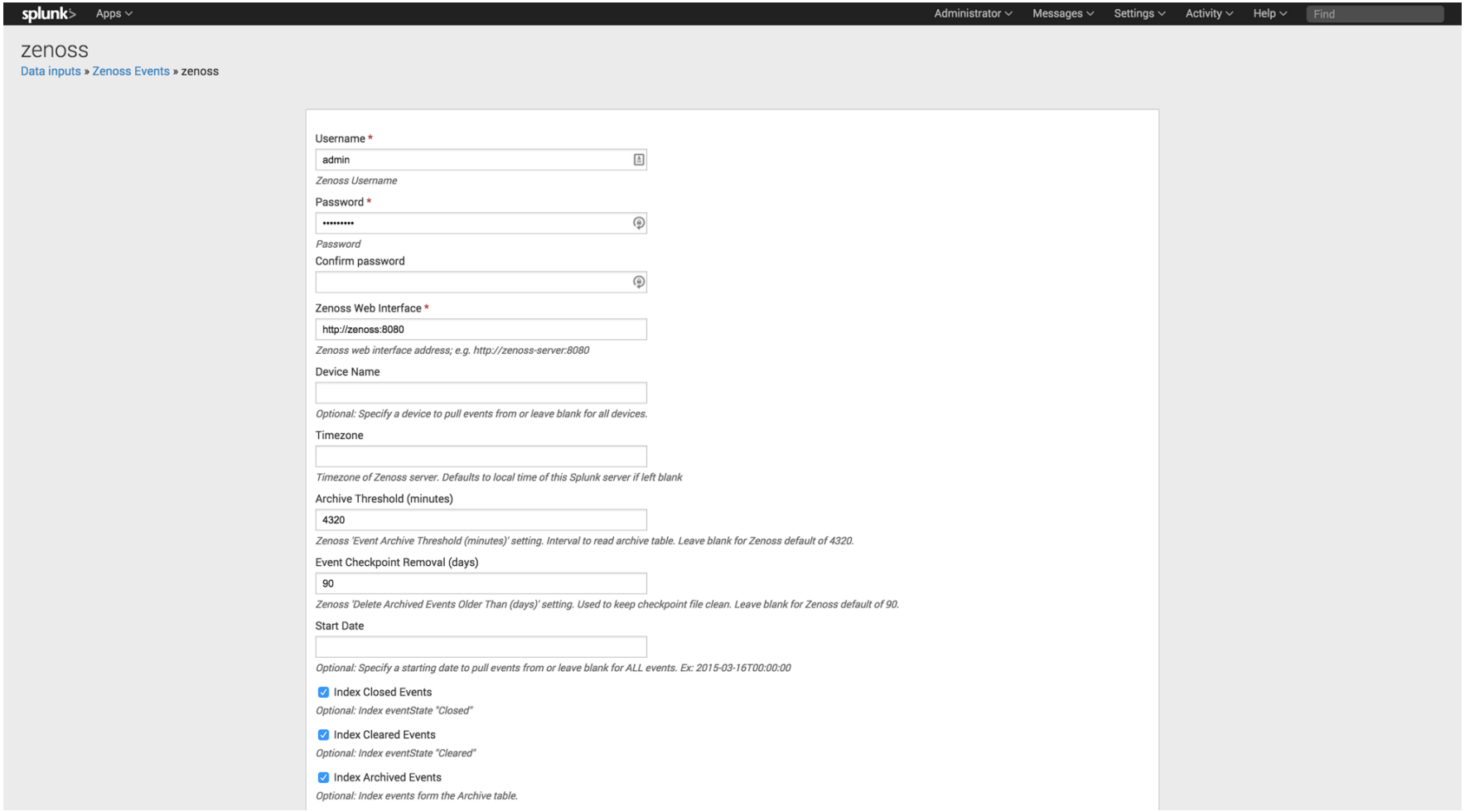
This or…

script://./bin/zenoss_wrapper.sh -u admin -p password -a http://zenoss:8080 -z America/Los_Angeles -t 4320 -r 90 -s 2015-03-16T00:00:00 -index-closed-events 1 -index-cleared-events 1 -index-archived-events 1 -index-suppressed-events 1 -index-repeat-events 1]

sourcetype = zenoss-events

interval = 60

index = zenoss


splunk> Apps
Administrator Messages Settings Activity Help Find
zenoss
Data inputs » Zenoss Events » zenoss
Username *
admin
Zenoss Username
Password *
Password
Confirm password
Zenoss Web Interface *
http://zenoss:8080
Zenoss web interface address; e.g. http://zenoss-server:8080
Device Name
Optional: Specify a device to pull events from or leave blank for all devices.
Timezone
Timezone of Zenoss server. Defaults to local time of this Splunk server if left blank
Archive Threshold (minutes)
4320
Zenoss 'Event Archive Threshold (minutes)' setting. Interval to read archive table. Leave blank for Zenoss default of 4320.
Event Checkpoint Removal (days)
90
Zenoss 'Delete Archived Events Older Than (days)' setting. Used to keep checkpoint file clean. Leave blank for Zenoss default of 90.
Start Date
Optional: Specify a starting date to pull events from or leave blank for ALL events. Ex: 2015-03-16T00:00:00
Index Closed Events
Optional: Index eventState "Closed"
Index Cleared Events
Optional: Index eventState "Cleared"
Index Archived Events
Optional: Index events form the Archive table.

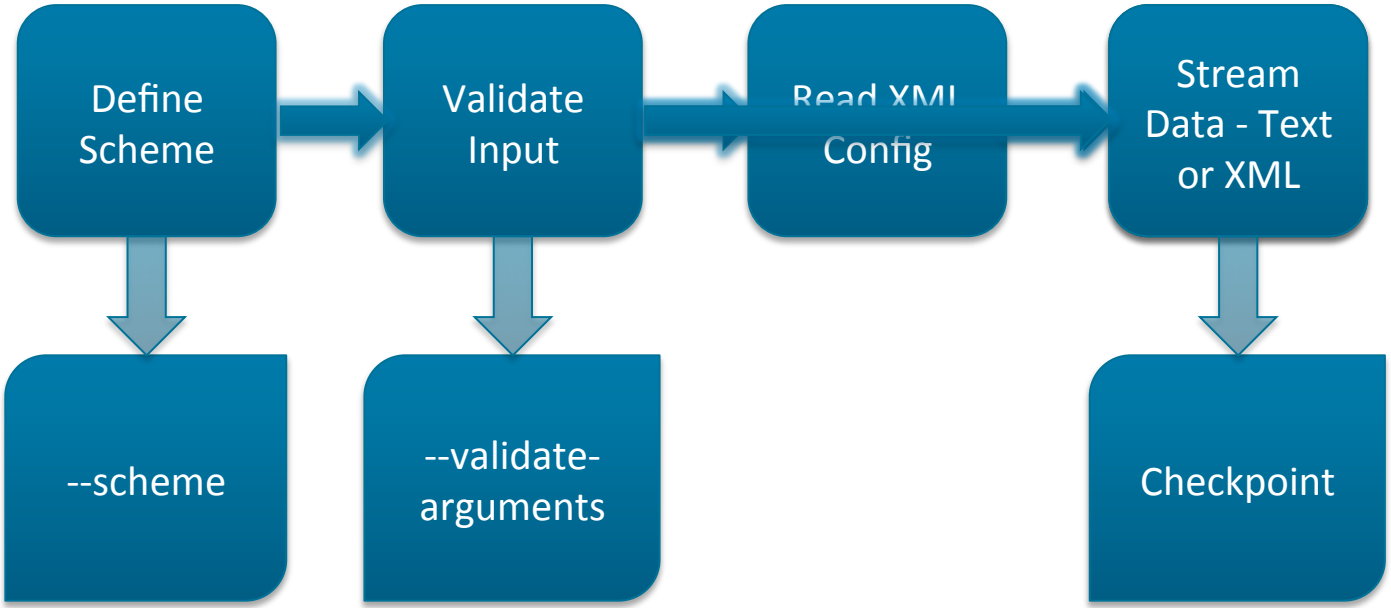
Why I Chose A Modular Input

- Programmatic collection from API via HTTP
- Needed ability to keep state and filter (checkpoint)
- Quick and Flexible configuration and (de-)activation with permissions
- Consumable in easy to understand

SDK Implementation

splunk>

Pseudo-code Without SDK



Step 0 - Create A Spec File

zenoss_events //<name>]

$SPLUNK_HOME/etc/apps/TA-zenoss/bin/zenoss_events.py

password = <value>
zenoss_server = <value>
device = <value>
start_date = <value>
index_closed = <value>
index_cleared = <value>
index_archived = <value>
archive_threshold = <value>
index_suppressed = <value>
index_repeats= <value>
checkpoint_delete_threshold = <value>
tzone = <value>

Skeleton Code

```
from splunklib.modularinput import *

class ZenossModInput(Script):

  def get_scheme(self):

  def validate_input(self, validation_definition):

  def stream_events(self, inputs, ew):

if __name__ == '__main__':
   sys.exit(ZenossModInput().run(sys.argv))
```

Define Introspection

```
def get_scheme(self):
    scheme = Scheme("Zenoss Events")
    scheme.description = "Modular input to pull events from Zenoss API”
    scheme.streaming_mode = “XML”
    scheme.use_external_validation = True
    scheme.use_single_instance = False

    username = Argument("username")
    username.data_type = Argument.data_type_string
    username.required_on_edit = True
    username.required_on_create = True
    scheme.add_argument(username)
return scheme
```

Step 2 - Implement Routines To Validate Configuration

```
def validate_input(self, validation_definition):
    tz =  validation_definition.parameters.get("tzone")
    interval = validation_definition.parameters.get("interval")

    # Validate timezone exists in pytz database
    if tz is not None and tz not in pytz.all_timezones:
        raise ValueError("Invalid timezone”)

    if int(interval) < 1:
        raise ValueError("Interval value must be a non-zero positive integer”)
```

Step 3 – Stream Data as XML

```
def stream_events(self, inputs, ew):
    # All script logic goes here
    for e in events['events']:
        event = Event(data = json.dumps(e))
        ew.write_event(event)
```

Name	Type	Description
data	string	event's text
stanza	string	name of input event should be sent to
time	float	time in seconds + 3 decimal places for milliseconds
host	string	event's host
index	string	index name to write event
source	string	source of event
sourcetype	string	sourcetype of event
done	boolean	complete event or event fragment?
unbroken	boolean	completely encapsulated in this event object?

Checkpointing

splunk>

Do I Need To Checkpoint?

Yes	No
Overlap in data for each script run?	Snapshot in time (ps, lsof)?
Avoid data duplication?	Unique values each script run?
Filter API calls - device, time?	Must record every record returned?

What Does Splunk Provide

Location

$SPLUNK_HOME/var/lib/splunk/modinputs/zenoss_events

Access To Location

```
def stream_events(self, inputs, ew):
    checkpoint_dir    inputs.metadata.get(  checkpoint_dir")
```

Everything Else - Checkpoint Implementation Is On You

- No SDK convenience methods for accessing KV Store
- Use service object from SDK and GET/POST to REST API
-

Logging

splunk>

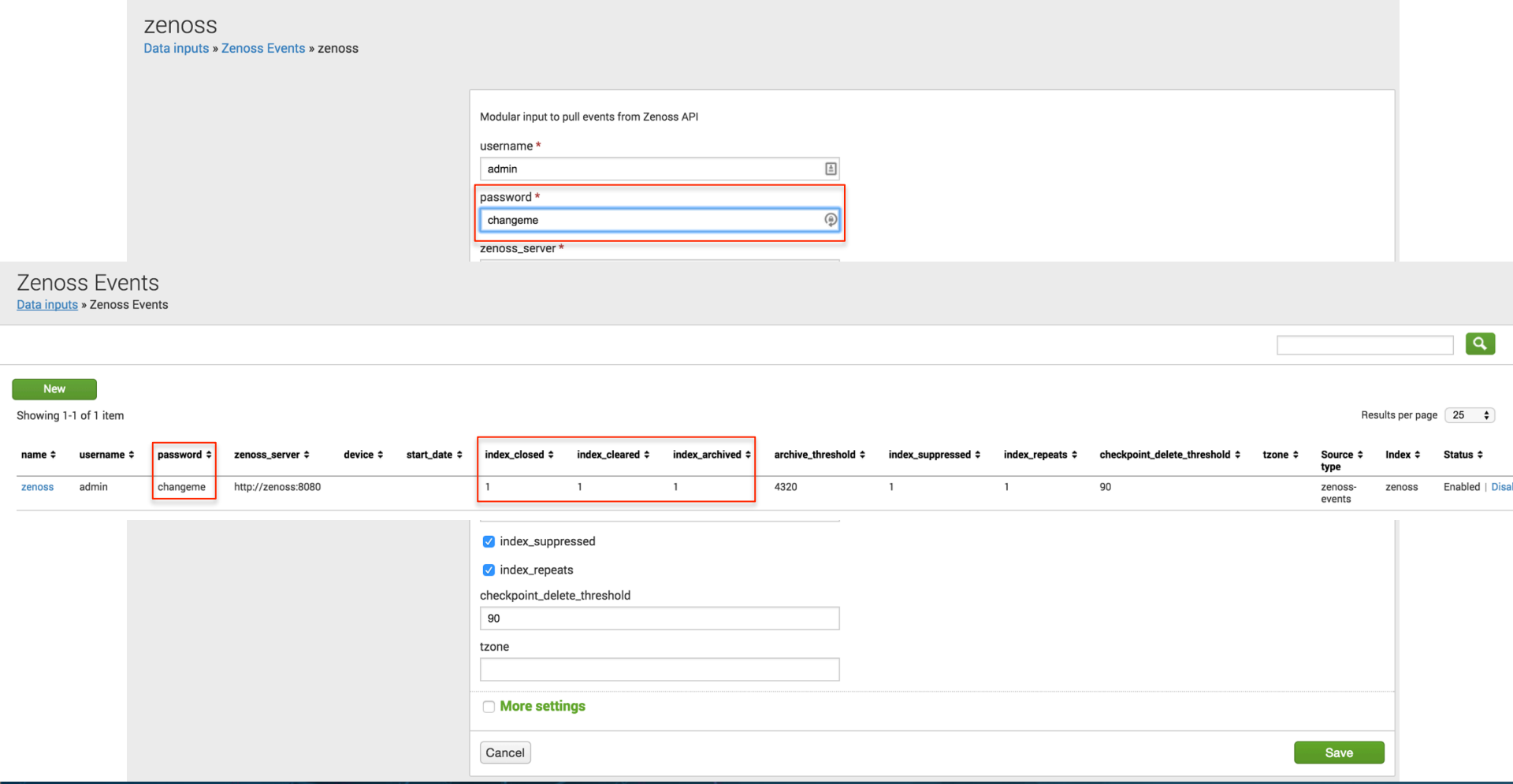
message = “Oh geez … awful”

ew.log(“ERROR”, me…

Severity
DEBUG
INFO
WARN
ERROR
FATAL

UI Customization

splunk>

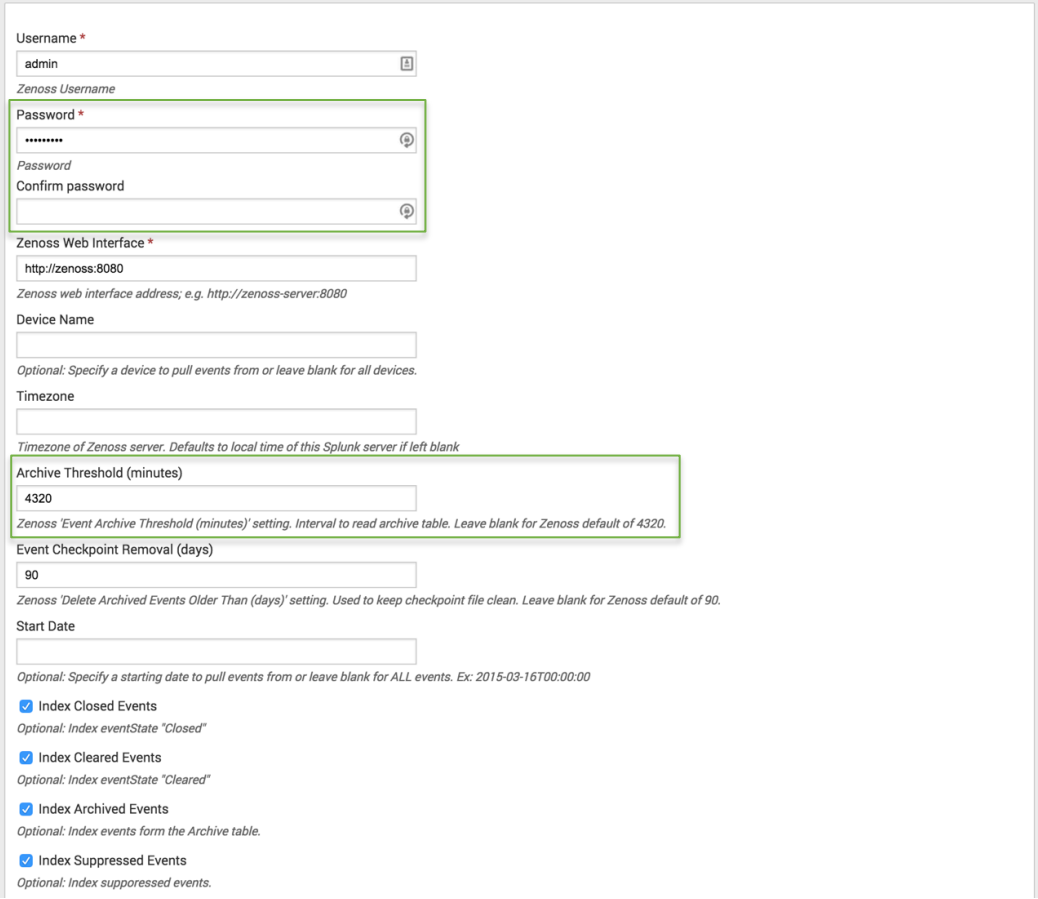

zenoss
Data inputs » Zenoss Events » zenoss
Modular input to pull events from Zenoss API
username *
admin
password *
changeme
zenoss_server *
Zenoss Events
Data inputs » Zenoss Events
New
Showing 1-1 of 1 item
Results per page 25
name username password zenoss_server device start_date index_closed index_cleared index_archived archive_threshold index_suppressed index_repeats checkpoint_delete_threshold tzone Source type Index Status
zenoss admin changeme http://zenoss:8080 1 1 1 4320 1 1 90 zenoss-events zenoss Enabled | Disal
index_suppressed
index_repeats
checkpoint_delete_threshold
90
tzone
More settings
Cancel
Save

Manager XML

- Override Splunk default config page
- Customize with example text & assign default values
- Control fields displayed during create, update, and list
- ***$SPLUNK_HOME/etc/apps/TA-zenoss/default/data/ui/manager/zenoss_events.xml***

zenoss

Data inputs » Zenoss Events » zenoss



Password Management

splunk>

Strategies

REST storage/passwords endpoint

+ Passwords masked on entry
+ Encrypted with easy clear text access
+ Hash stored in local/passwords.conf
- No search Head Cluster Support

Manager XML

+ Passwords masked on entry
+ Search Head Cluster Support
- Clear text in inputs.conf - file/directory l

Do It Yourself

+ Flexibility
- Additional development time

Testing Your Script

splunk>

Testing Your Script

splunk cmd splunkd print-modinput-config *scheme stanza*

```
$SPLUNK_HOME/bin/splunk cmd splunkd print-modinput-config \
zenoss_events \
zenoss_events://zenoss \
| $SPLUNK_HOME/bin/splunk cmd python zenoss_events.py
```

Key Takeaways

splunk>

- The SDK makes implementing modular inputs EASY
- Elevate your scripted input to modular input and share on splunkbase
- Get the code and play with it

  https://github.com/sghaskell/TA-zenoss
  https://splunkbase.splunk.com/app/2766/

THANK YOU

splunk>