python-reminders Documentation

Release 0.1

WisdomWolf

Mar 16, 2018

Contents

python-reminders is intended to be a versatile framework for monitoring resources and generating repeating alerts based on certain conditions. It is very much still a work in progress at this point and should be considered pre-alpha. This includes the state of the documentation as well as the working status of the project.

Contents:

CHAPTER 1

reminders package

1.1 Submodules

1.2 reminders.alerters module

class reminders.alerters.**Alerter**(*reminder*, *message*, *notifiers=None*, *repeat_interval={}*, *max_repeat=0*, *alert_on_activate=True*, **args*, ***kwargs*)

Bases: object

Base Alert object to handle reminder notifications.

__init__(*reminder*, *message*, *notifiers=None*, *repeat_interval={}*, *max_repeat=0*, *alert_on_activate=True*, **args*, ***kwargs*)

Create Alerter object.

Parameters

- **reminder** (Reminder) – Reminder instance to associate this alert with.
- **message** (*str*) – Message to be sent by notifier(s). note: This was added as part of POC. Likely to be removed in future.
- **notifiers** – ¯_()_/¯
- **repeat_args** (*dict*) – Arguments to set repeat interval
- **max_repeat** (*int*) – number of times alert should repeat.
- **alert_on_activate** (*bool*) – When True alert will be emitted as soon as activated rather than waiting for first scheduled job to trigger.

activate()

Activate alerts

alert()

Send alert

deactivate()
Deactivate all existing alerts.

class reminders.alerters.**HTTPAlerter**(*request_kwargs*, *json_params=True*, **args*, ***kwargs*)
Bases: *reminders.alerters.Alerter*

Alerts via POST to HTTP REST interface

__init__(*request_kwargs*, *json_params=True*, **args*, ***kwargs*)
Create HTTPAlerter object

Parameters

- **request_kwargs** (*dict*) – Dictionary containing keyword arguments to be passed to requests.post()
- **json_params** (*bool*) – Indicates if request_kwargs['data'] should be transmitted as JSON string.

alert()
Emit Alert

class reminders.alerters.**LogAlerter**(**args*, ***kwargs*)
Bases: *reminders.alerters.Alerter*

Alerter for outputting to logger

__init__(**args*, ***kwargs*)
Create Alerter object.

Parameters

- **reminder** (Reminder) – Reminder instance to associate this alert with.
- **message** (*str*) – Message to be sent by notifier(s). note: This was added as part of POC. Likely to be removed in future.
- **notifiers** – ¯_()_/¯
- **repeat_args** (*dict*) – Arguments to set repeat interval
- **max_repeat** (*int*) – number of times alert should repeat.
- **alert_on_activate** (*bool*) – When True alert will be emitted as soon as activated rather than waiting for first scheduled job to trigger.

alert()
Emit alert to log

1.3 reminders.main module

1.4 reminders.reminder module

class reminders.reminder.**Reminder**(*condition*, *daemon=None*, *watcher=None*, *alerter=None*)
Bases: object

Base Reminder object to handle watch and notification for a single reminder.

__init__(*condition*, *daemon=None*, *watcher=None*, *alerter=None*)
Create Reminder object.

Parameters

- **condition** (*str*) – An expression to indicate that an alert should be sent. Should evaluate to True or False only.
- **daemon** (ReminderDaemon) – A ReminderDaemon instance where jobs will be scheduled.
- **watcher** (Watcher) – A Watcher instance to handle resource monitoring.
- **alerter** (Alerter) – An Alerter instance to handle sending notifications for Reminder.

activate()
TBD - May be unnecessary at this level.

alerter_type_map = {'log': <class 'reminders.alerters.LogAlerter'>}

check()
Runs self.test_condition() and sends Alert if True.

deactivate()
TBD - May be unnecessary at this level.

eval()
Evaluate self.expression

Returns True if alert should be started

Return type bool

now
Shortcut for expression evaluation against current time

status

test_condition()
Deprecated since version 0.1: Use *eval()* instead.

watcher_type_map = {'http': <class 'reminders.watchers.HTTPWatcher'>, 'mqtt': <class

class reminders.reminder.**ReminderDaemon**(*blocking=True*, *timezone='UTC'*, *config_path='.'*, *logger_level=None*, **args*, ***kwargs*)
Bases: object

Parent Daemon to keep track of scheduled jobs and watch for config file changes.

__init__(*blocking=True*, *timezone='UTC'*, *config_path='.'*, *logger_level=None*, **args*, ***kwargs*)
Create ReminderDaemon object.

Parameters

- **blocking** (*boolean*) – Determines if Scheduler should be BlockingScheduler or BackgroundScheduler.
- **timzone** (*str*) – Timezone for the scheduler to use when scheduling jobs.
- **config_path** (*str*) – Path to configuration files.
- **logger_level** (*int*) – Level to set logger to.

add_reminder(*reminder_config*)
Create new reminder and add to daemon.

Parameters **reminder_config** (*dict*) – Dictionary configuration for creating Reminder. Typically loaded from YAML file.

load_yaml(*path*)
Read and process yaml config.

Parameters **path** (*str*) – The path of yaml config to load.

on_created(*event*)
Callback for on_created events to be associated with watchdog EventHandler.

Parameters **event** – Event object representing the file system event.

Event type watchdog.events.FileSystemEvent

on_deleted(*event*)
Callback for on_deleted events to be associated with watchdog EventHandler.

Parameters **event** – Event object representing the file system event.

Event type watchdog.events.FileSystemEvent

remove_reminder(*reminder*)
Remove reminder from Daemon.

Parameters **reminder** (Reminder) – The Reminder to be removed.

start()
Start the observer and scheduler associated with daemon.

update(*reminder*)
Update Daemon with new Reminder object. Operates by either appending new reminder or replacing existing reminder.

Parameters **reminder** (Reminder) – Reminder to be added or updated.

1.5 reminders.watchers module

class reminders.watchers.**HTTPWatcher**(*request_kwargs*, *json_expression*, **args*, ***kwargs*)
Bases: *reminders.watchers.Watcher*

Watcher object for monitoring HTTP(S) REST Resource.

__init__(*request_kwargs*, *json_expression*, **args*, ***kwargs*)
Create HTTPWatcher object. note:

Assumes response is JSON. May require separate classes for JSON/XML/Others in future.

Parameters

- **request_kwargs** (*dict*) – Dictionary containing keyword arguments to be passed to requests.get()
- **json_expression** (*str*) – JMESPath expression to be used to retrieve status from results JSON object.

update()
Return resource status for Reminder to evaluate.

class reminders.watchers.**MQTTWatcher**(*hostname*, *port=1883*, *tls=False*, *topic_kwargs=None*, *username=None*, *password=None*, **args*, ***kwargs*)
Bases: *reminders.watchers.Watcher*

Watcher object for monitoring MQTT Resource.

__init__(*hostname*, *port=1883*, *tls=False*, *topic_kwargs=None*, *username=None*, *password=None*, **args*, ***kwargs*)
Create MQTTWatcher object.

Parameters

- **hostname** (*str*) – url for MQTT client to connect to.
- **port** (*int*) – port to be used for MQTT connection.
- **tls** (*bool*) – Use SSL/TLS for secure connection.
- **topic_kwargs** (*dict*) –

  Dictionary containing:

  – topic to monitor

  – condition to start Alerter

  – condition to cancel Alerter

Note: May be replaced with just topic as *str* in future.

Parameters

- **username** (*str*) – Username for MQTT client authentication.
- **password** (*str*) – Password for MQTT client authentication.

listener_callback(*userdata*, *msg*)

Callback to set status when msg received on monitored topic.

Parameters

- **client** – Required by callback signature.
- **userdata** – Required by callback signature.
- **msg** – Message received on topic that generated this callback.

update()

Return status for Reminder evaluation.

class reminders.watchers.**NullWatcher**(**args*, ***kwargs*)

Bases: *reminders.watchers.Watcher*

Empty watcher for timed reminders

__init__(**args*, ***kwargs*)

Create Watcher object.

Parameters

- **reminder** (Reminder) – Reminder instance to associate watcher with.
- **schedules** (*dict*) – Initial job schedules for watcher to use. *Possibly going to be removed from base class*

update()

REQUIRED Return status from monitored resource. Up to concrete class to determine implementation.

class reminders.watchers.**Watcher**(*reminder*, *schedules*, **args*, ***kwargs*)

Bases: object

Base Watcher object for resource monitoring

__init__(*reminder*, *schedules*, **args*, ***kwargs*)

Create Watcher object.

Parameters

- **reminder** (Reminder) – Reminder instance to associate watcher with.
- **schedules** (*dict*) – Initial job schedules for watcher to use. *Possibly going to be removed from base class*

update()

REQUIRED Return status from monitored resource. Up to concrete class to determine implementation.

1.6 Module contents

CHAPTER 2

reminders

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index

Symbols

A

C

D

E

H

L

M

N

O

R

S

T

U

W