Notifier Documentation

Release 1.0

OpenStack

October 06, 2015

Contents

CHAPTER 1

Notifier

In memory pub/sub [1]

- Free software: Apache license

1.1 API

1.1.1 Classes

class notifier.**Notifier**(*logger=None*)
A notification (pub/sub *like*) helper class.

It is intended to be used to subscribe to notifications of events occurring as well as allow a entity to post said notifications to any associated subscribers without having either entity care about how this notification occurs.

ANY = '*'
Kleene star constant that is used to recieve all notifications

RESERVED_KEYS = ('details',)
Keys that can *not* be used in callbacks arguments

can_be_registered(*event_type*)
Checks if the event can be registered/subscribed to.

Returns whether the event_type can be registered/subscribed to.

Return type boolean

can_trigger_notification(*event_type*)
Checks if the event can trigger a notification.

Parameters **event_type** – event that needs to be verified

Returns whether the event can trigger a notification

Return type boolean

copy()
Clones this notifier (and its bound listeners).

deregister(*event_type*, *callback*, *details_filter=None*)
Remove a single listener bound to event `event_type`.

Parameters **event_type** – deregister listener bound to event_type

[1] See: publish–subscribe pattern

Returns if the callback was deregistered

Return type boolean

deregister_event(*event_type*)
Remove a group of listeners bound to event `event_type`.

Parameters **event_type** – deregister listeners bound to event_type

Returns how many callbacks were deregistered

Return type int

is_registered(*event_type*, *callback*, *details_filter=None*)
Check if a callback is registered.

Returns if the callback is registered

Return type boolean

listeners_iter()
Return an iterator over the mapping of event => listeners bound.

The listener list(s) returned should **not** be mutated.

NOTE(harlowja): Each listener in the yielded (event, listeners) tuple is an instance of the *Listener* type, which itself wraps a provided callback (and its details filter callback, if any).

notify(*event_type*, *details*)
Notify about an event occurrence.

All callbacks registered to receive notifications about given event type will be called. If the provided event type can not be used to emit notifications (this is checked via the *can_be_registered()* method) then it will silently be dropped (notification failures are not allowed to cause or raise exceptions).

Parameters

- **event_type** – event type that occurred
- **details** (*dictionary*) – additional event details *dictionary* passed to callback keyword argument with the same name

register(*event_type*, *callback*, *args=None*, *kwargs=None*, *details_filter=None*)
Register a callback to be called when event of a given type occurs.

Callback will be called with provided `args` and `kwargs` and when event type occurs (or on any event if `event_type` equals to *ANY*). It will also get additional keyword argument, `details`, that will hold event details provided to the *notify()* method (if a details filter callback is provided then the target callback will *only* be triggered if the details filter callback returns a truthy value).

Parameters

- **event_type** – event type input
- **callback** – function callback to be registered.
- **args** (*list*) – non-keyworded arguments
- **kwargs** (*dictionary*) – key-value pair arguments

Returns the listener that was registered

Return type *Listener*

reset()
Forget all previously registered callbacks.

class notifier.**RestrictedNotifier**(*watchable_events*, *allow_any=True*, *logger=None*)
A notification class that restricts events registered/triggered.

NOTE(harlowja): This class unlike *Notifier* restricts and disallows registering callbacks for event types that are not declared when constructing the notifier.

can_be_registered(*event_type*)
Checks if the event can be registered/subscribed to.

Parameters **event_type** – event that needs to be verified

Returns whether the event can be registered/subscribed to

Return type boolean

events_iter()
Returns iterator of events that can be registered/subscribed to.

NOTE(harlowja): does not include back the ANY event type as that meta-type is not a specific event but is a capture-all that does not imply the same meaning as specific event types.

class notifier.**Listener**(*callback*, *args=None*, *kwargs=None*, *details_filter=None*)
Immutable helper that represents a notification listener/target.

args
Tuple of positional arguments to use in future calls.

callback
Callback (can not be none) to call with event + details.

details_filter
Callback (may be none) to call to discard events + details.

is_equivalent(*callback*, *details_filter=None*)
Check if the callback is same

Parameters

- **callback** – callback used for comparison
- **details_filter** – callback used for comparison

Returns false if not the same callback, otherwise true

Return type boolean

kwargs
Frozen dictionary of keyword arguments to use in future calls.

1.1.2 Functions

notifier.**register_deregister**(**args*, ***kwds*)
Context manager that registers a callback, then deregisters on exit.

NOTE(harlowja): if the callback is none, then this registers nothing, which is different from the behavior of the register method which will *not* accept none as it is not callable...

1.1.3 Constants

Notifier.**ANY = '*'**
Kleene star constant that is used to recieve all notifications

CHAPTER 2

Release Notes

2.1 CHANGES

- Get docs building correctly
- Fix numbering
- Tweak this file to be more useful
- Allow for providing a customized logger to be used on debug/warn calls
- Ensure ‘register’ and ‘is_registered’ obtains the thread lock
- Have ‘register’ return the listener registered
- Update docstring for ‘can_be_registered’ method
- Get tox env tests working
- Move and expose ANY constant
- Use a frozendict to ensure immutable listener keyword arguments
- Denote that the listeners returned from iteration should not be mutated
- Fix newlines and imports
- Thread safe usage and tests
- Initial shift from taskflow -> notifier
- Initial commit

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

A

C

D

E

I

K

L

N

R