eventify Documentation

Release 0.5.26

Matthew Harris

May 03, 2019

CONTENTS:

CHAPTER

ONE

EVENTIFY PACKAGE

1.1 Subpackages

1.1.1 eventify.drivers package

Submodules

eventify.drivers.crossbar module

Crossbar Driver Module

class eventify.drivers.crossbar.**Component**(*config=None*)

Bases: eventify.drivers.base.BaseComponent, autobahn.asyncio.wamp.ApplicationSession

Handle subscribing to topics

emit_event(*event*)

Publish an event back to crossbar :param event: Event object

log = <Logger eventify.drivers.crossbar (WARNING)>

lookup_session(*topic_name*)

Attempts to find the session id for a given topic

http://crossbar.io/docs/Subscription-Meta-Events-and-Procedures/

onClose(*wasClean*)

Disconnect when connection to message broker is lost

onDisconnect()

Event fired when transport is lost

onJoin(*details*)

Implements autobahn.wamp.interfaces.ISession.onJoin()

onLeave(*reason=None*, *message=None*)

Parameters

- **reason** –
- **message** –

onUserError(*fail*, *message*)

Handle user errors

show_sessions()
Returns an object with a lists of the session IDs for all sessions currently attached to the realm

http://crossbar.io/docs/Session-Metaevents-and-Procedures/

total_sessions()
Returns the number of sessions currently attached to the realm.

http://crossbar.io/docs/Session-Metaevents-and-Procedures/

class eventify.drivers.crossbar.**Service**(*driver='crossbar'*, *config_file='config.json'*, *handlers=None*)
Bases: *eventify.Eventify*

Create crossbar service

check_transport_host()
Check if crossbar port is open on transport host

reconnect()
Handle reconnect logic if connection to crossbar is lost

setup_runner()
Setup instance of runner var

start(*start_loop=True*)
Start a producer/consumer service

Module contents

1.1.2 eventify.event package

Module contents

Event Module

class eventify.event.**Event**(*event*, ***kwargs*)
Bases: object

Event object

eventify.event.**replay_events**()
Replay events from a given timestamp or event_id :param timestamp: Human readable datetime :param event_id: UUID of a given event

1.1.3 eventify.exceptions package

Module contents

Eventify exceptions

exception eventify.exceptions.**EventifyConfigError**(*message*)
Bases: Exception

Configuration related errors

exception eventify.exceptions.**EventifyHandlerInitializationFailed**(*message*)
Bases: Exception

Error initialization of handler. It means that we can not continue working so service should be stopped with non zero exit code

exception eventify.exceptions.**EventifyInitError**(*message*)
Bases: Exception

Initialization errors

exception eventify.exceptions.**EventifyPersistanceConfigError**(*message*)
Bases: Exception

Errors with persistance configuration

exception eventify.exceptions.**EventifySanityError**(*message*)
Bases: Exception

Errors with security and sanity

1.1.4 eventify.persist package

Submodules

eventify.persist.constants module

Eventify Persistance Configuration

Module contents

Persist Helper Module

eventify.persist.**persist_event**(*topic*, *event*, *pool*)
Track event to prevent duplication of work and potential loss of event :param topic: The event topic :param event: The event object

1.1.5 eventify.tracking package

Submodules

eventify.tracking.constants module

Eventify Tracking Constants

class eventify.tracking.constants.**EventState**
Bases: object

Event States

completed = 'COMPLETED'

inflight = 'IN-FLIGHT'

started = 'STARTED'

Module contents

Event Tracking Module

eventify.tracking.**track_event**(*event*, *state*, *service_name*)
Store state of events in memory :param event: Event object :param state: EventState object :param service_name: Name of service name

1.2 Submodules

1.3 eventify.base_handler module

Abstract Base Class for Handler

class eventify.base_handler.**BaseHandler**
Bases: object

Base event handler

publish_topic = None

session = None

set_session(*session*)
Setup session for publishing

subscribe_topic = None

1.4 eventify.service module

Service Module

class eventify.service.**Service**(*driver='crossbar'*, *config_file='config.json'*, *handlers=None*)
Bases: *eventify.drivers.crossbar.Service*

Crossbar Service

eventify.service.**event_tracker**(*func*)
Event tracking handler

1.5 Module contents

Eventify! A simple module for implementing event driven systems

class eventify.**Eventify**(*driver='crossbar'*, *config_file='config.json'*, *handlers=None*)
Bases: object

Base Class for eventify

static check_event_loop()
Check if event loop is closed and create a new event loop

config_sanity_check()
Base configuration sanity checks

load_config
Load configuration for the service

Parameters **config_file** – Configuration file path

set_missing_defaults()
Ensure that minimal configuration is setup and set defaults for missing values

CHAPTER

TWO

SUMMARY

A simple python module for building event driven systems.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

T