SPADE PubSub Documentation

Release 0.1.3

Javi Palanca

Jul 05, 2021

CONTENTS:

CHAPTER

ONE

SPADE PUBSUB

SPADE Plugin for PubSub support.

- Free software: MIT license
- Documentation: https://spade-pubsub.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install SPADE PubSub, run this command in your terminal:

```
$ pip install spade_pubsub
```

This is the preferred method to install SPADE PubSub, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for SPADE PubSub can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/javipalanca/spade_pubsub
```

Or download the tarball:

```
$ curl -OJL https://github.com/javipalanca/spade_pubsub/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

The PubSub plugin gives agents the ability to create nodes, subscribe to nodes and publish items to nodes following the XEP 0060 (https://xmpp.org/extensions/xep-0060.html).

To use SPADE PubSub in a project:

```
from spade_pubsub import PubSubMixin
from spade import Agent

class YourAgent(PubSubMixin, Agent):
    # Your code here...
```

Warning: Mixins MUST be always placed before the Agent class in the inheritance order.

3.1 Working with nodes

An agent can create a node where other agents may subscribe in order to receive notifications when new items are published in such node.

Warning: Due to limitations of the XMPP Publish-Subscribe standard, agents MUST be registered in the XMPP server with creation privileges in order to create nodes. (e.g. In Prosody include them in the admin list (*admins = {}*).

Tip: Note that you MUST substitute PUBSUB_JID with the address of the pubsub component that your XMPP server uses (e.g. "pubsub.localhost)

To create and delete a node in a PubSub server:

```
await self.agent.pubsub.create(PUBSUB_JID, "Name of the node")
await self.agent.pubsub.delete(PUBSUB_JID, "Name of the node")
```

To get all nodes from a PubSub server:

```
list_of_nodes = await self.agent.pubsub.get_nodes(PUBSUB_JID)
```

To purge all items from a node:

```
await self.agent.pubsub.purge(PUBSUB_JID, "Name of the node")
```

3.2 Publishing Items to a node

Once a node is created, you can publish information to that node. That information will be received by all subscriptors of the node.

Publish an item:

```
await self.agent.pubsub.publish(PUBSUB_JID, "Name of the node", "Payload of the item")
```

Get all published items from a node:

```
items = await self.agent.pubsub.get_items(PUBSUB_JID, "Name of the node")
```

Subscribe and unsubscribe:

```
await self.agent.pubsub.subscribe(PUBSUB_JID, "Name of the node")
await self.agent.pubsub.unsubscribe(PUBSUB_JID, "Name of the node")
```

Register callback function to receive published items:

```
def my_callback(self, jid, node, item, message=None):
    # Your code here

# register callback
self.agent.pubsub.set_on_item_published(my_callback)
```

Get all subscriptions of a node:

```
list_of_subs = await self.agent.pubsub.get_node_subscriptions(PUBSUB_JID, "Name of the␣
↪node")
```

CHAPTER

FOUR

API DOCUMENTATION

4.1 spade_pubsub package

4.1.1 Submodules

4.1.2 spade_pubsub.pubsub module

class spade_pubsub.pubsub.**Payload**(**args*, ***kwargs*)
Bases: aioxmpp.xso.model.XSO

ATTR_MAP = {}

CHILD_MAP = {}

CHILD_PROPS = OrderedSet([])

COLLECTOR_PROPERTY = None

DECLARE_NS = {None: 'spade.pubsub'}

TAG = ('spade.pubsub', 'payload')

TEXT_PROPERTY

data

class spade_pubsub.pubsub.**PubSubMixin**
Bases: object

This mixin provides PubSub support to SPADE agents. It must be used as superclass of a spade.Agent subclass.

class PubSubComponent(*client*)
Bases: object

async create(*target_jid: str*, *target_node: Optional[str] = None*)
Create a new node at a service.
Args: target_jid (str): Address of the PubSub service. target_node (str or None): Name of the PubSub node to create

async delete(*target_jid: str*, *target_node: Optional[str]*, *redirect_uri: Optional[str] = None*)
Delete an existing node.
Args: target_jid (str): Address of the PubSub service. target_node (str or None): Name of the PubSub node to delete. redirect_uri (str or None): A URI to send to subscribers to indicate a replacement for the deleted node.

async get_items(*target_jid: str*, *target_node: Optional[str]*)
Request all items at a service or collection node.
Args: target_jid (str): Addressof the PubSub service. target_node (str): Name of the PubSub node.

async get_node_subscriptions(*target_jid: str*, *target_node: Optional[str]*) → List[str]
Return the subscriptions of other jids with a node.
Args: target_jid (str): Address of the PubSub service. target_node (str): Name of the node to query

async get_nodes(*target_jid: str*, *target_node: Optional[str] = None*)
Request all nodes at a service or collection node.
Args: target_jid (str): Address of the PubSub service. target_node (str or None): Name of the collection node to query

async notify(*target_jid: str*, *target_node: str*)
Notify all subscribers of a node without publishing an item. "Publish" to the node at jid without any item. This merely fans out a notification.
Args: target_jid (str): Address of the PubSub service. target_node (str): Name of the PubSub node to send a notify from.

async publish(*target_jid: str*, *target_node: str*, *payload: str*, *item_id: Optional[str] = None*)
Publish an item to a node.
Args: target_jid (str): Address of the PubSub service. target_node (str): Name of the PubSub node to publish to. payload (str): Payload to publish. item_id (str or None): Item ID to use for the item.

async purge(*target_jid: str*, *target_node: Optional[str]*)
Delete all items from a node.
Args: target_jid (str): JID of the PubSub service target_node (str): Name of the PubSub node

async retract(*target_jid: str*, *target_node: str*, *item_id: str*, *notify=False*)
Retract a previously published item from a node.
Args: target_jid (str): Address of the PubSub service. target_node (str): Name of the PubSub node to send a notify from. item_id (str): The ID of the item to retract. notify (bool): Flag indicating whether subscribers shall be notified about the retraction.

set_on_item_published(*callback*)

set_on_item_retracted(*callback*)

async subscribe(*target_jid: str*, *target_node: Optional[str] = None*, *subscription_jid: Optional[str] = None*, *config=None*)
Subscribe to a node.
Args: target_jid (str): Address of the PubSub service. target_node (str): Name of the PubSub node to subscribe to. subscription_jid (str): The address to subscribe to the service. config (Data): Optional configuration of the subscription

async unsubscribe(*target_jid: str*, *target_node: Optional[str] = None*, *subscription_jid: Optional[str] = None*, *subid=None*)
Unsubscribe from a node.
Args: target_jid (str): Address of the PubSub service. target_node (str): Name of the PubSub node to unsubscribe from. subscription_jid (str): The address to subscribe from the service. subid (str): Unique ID of the subscription to remove.

4.1.3 Module contents

Top-level package for SPADE PubSub.

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/javipalanca/spade_pubsub/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

SPADE PubSub could always use more documentation, whether as part of the official SPADE PubSub docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/javipalanca/spade_pubsub/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *spade_pubsub* for local development.

1. Fork the *spade_pubsub* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/spade_pubsub.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv spade_pubsub
$ cd spade_pubsub/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 spade_pubsub tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/javipalanca/spade_pubsub/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_spade_pubsub
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Javi Palanca

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

7.1 0.1.3 (2021-06-29)

- Minor bug fixed.

7.2 0.1.2 (2021-06-25)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX