panoramisk Documentation

Release 1.5.dev0

Gael Pasgrimaud

Sep 09, 2021

Contents

Panoramisk is a library based on python's AsyncIO to play with Asterisk's manager.

It uses the TCP manager server to listen to events and send actions.

For basic usage, you have some examples in examples/ folder.

You can find some help on IRC: irc://irc.freenode.net/panoramisk (www)

Source code is available at https://github.com/gawel/panoramisk/

Check the full documentation

I've spent hours writing this software, with love. Please consider tiping if you like it:

BTC: 1PruQAwByDndFZ7vTeJhyWefAghaZx9RZg

ETH: 0xb6418036d8E06c60C4D91c17d72Df6e1e5b15CE6

LTC: LY6CdZcDbxnBX9GFBJ45TqVj8NykBBqsmT

CHAPTER 1

Installation

Install, upgrade and uninstall panoramisk with these commands:

```
$ pip install panoramisk
$ pip install --upgrade panoramisk
$ pip uninstall panoramisk
```

CHAPTER 2

Detailed Documentation

2.1 panoramisk - AMI

An API to communicate with Asterisk's AMI

2.1.1 Configure Asterisk

In `/etc/asterisk/manager.conf`, add:

```
[username]
secret=password
deny=0.0.0.0/0.0.0.0
permit=127.0.0.1/255.255.255.255
read = all
write = all
```

Launch:

```
$ rasterisk -x 'manager reload'
```

2.1.2 API

class panoramisk.**Manager**(***config*)

Main object:

```
>>> manager = Manager(
...     host='127.0.0.1',
...     port=5038,
...     ssl=False,
...     encoding='utf8')
```

close()
Close the connection

connect(*run_forever=False*, *on_startup=None*, *on_shutdown=None*)
connect to the server

register_event(*pattern*, *callback=None*)
register an event. See `Message`:

```
>>> def callback(manager, event):
...     print(manager, event)
>>> manager = Manager()
>>> manager.register_event('Meetme*', callback)
<function callback at 0x...>
```

You can also use the manager as a decorator:

```
>>> manager = Manager()
>>> @manager.register_event('Meetme*')
... def callback(manager, event):
...     print(manager, event)
```

run_forever(*on_startup*, *on_shutdown*)
Start loop forever

send_action(*action*, *as_list=None*, ***kwargs*)
Send an `Action` to the server:

Parameters

- **action** (*Action or dict or Command*) – an Action or dict with action name and parameters to send
- **as_list** (*boolean*) – If True, the action will retrieve all responses

Returns an Action that will receive the response(s)

Return type panoramisk.actions.Action

Example

To retrieve answer:

```
manager = Manager()
resp = await manager.send_action({'Action': 'Status'})
```

Or with an async for:

```
manager = Manager()
async for resp in  manager.send_action({'Action': 'Status'}):
    print(resp)
```

See https://wiki.asterisk.org/wiki/display/AST/AMI+Actions for more information on actions

send_agi_command(*channel*, *command*, *as_list=False*)
Send a `Command` to the server:

Parameters

- **channel** (*String*) – Channel name where to launch command. Ex: 'SIP/000000-00000a53'
- **command** (*String*) – command to launch. Ex: 'GET VARIABLE async_agi_server'

- **as_list** (*boolean*) – If True, the action Future will retrieve all responses

Returns a Future that will receive the response

Return type asyncio.Future

Example

```
manager = Manager()
resp = manager.send_agi_command('SIP/000000-00000a53',
                                'GET VARIABLE async_agi_server')
```

Return a response `Message`. See https://wiki.asterisk.org/wiki/display/AST/Asterisk+11+ManagerAction_AGI

send_command(*command*, *as_list=False*)

Send a `Command` to the server:

```
manager = Manager()
resp = await manager.send_command('http show status')
```

Return a response `Message`. See https://wiki.asterisk.org/wiki/display/AST/ManagerAction_Command

2.2 panoramisk.fast_agi - Fast AGI

An API to create Fast AGI applications

2.2.1 API

class panoramisk.fast_agi.**Application**(*default_encoding='utf-8'*, *loop=None*)

Main object:

```
>>> fa_app = Application()
```

add_route(*path*, *endpoint*)

Add a route for FastAGI requests:

Parameters

- **path** (*String*) – URI to answer. Ex: 'calls/start'
- **endpoint** (*callable*) – command to launch. Ex: start

Example

```
async def start(request):
    print('Receive a FastAGI request')
    print(['AGI variables:', request.headers])

fa_app = Application()
fa_app.add_route('calls/start', start)
```

del_route(*path*)

Delete a route for FastAGI requests:

Parameters **path** (*String*) – URI to answer. Ex: 'calls/start'

Example

```
async def start(request):
    print('Receive a FastAGI request')
    print(['AGI variables:', request.headers])

fa_app = Application()
fa_app.add_route('calls/start', start)
fa_app.del_route('calls/start')
```

handler (*reader*, *writer*)

AsyncIO coroutine handler to launch socket listening.

Example

```
async def start(request):
    print('Receive a FastAGI request')
    print(['AGI variables:', request.headers])

fa_app = Application()
fa_app.add_route('calls/start', start)
coro = asyncio.start_server(fa_app.handler, '0.0.0.0', 4574)
server = loop.run_until_complete(coro)
```

See https://docs.python.org/3/library/asyncio-stream.html

2.3 panoramisk.testing - Writing tests

```
>>> from panoramisk import testing
>>> manager = testing.Manager(stream=stream)  # stream is a filename contaning an␣
↪Asterisk trace
>>> future = manager.send_action({'Action': 'Ping'})
>>> resp = future.result()
>>> assert 'ping' in resp
>>> assert resp.ping == 'Pong'
```

2.3.1 API

class panoramisk.testing.**Manager**(**config*)

CHAPTER 3

Who use Panoramisk on production ?

For now, mainly Eyepea and ALLOcloud.

You shouldn't know theses companies, however, Eyepea is a provider of several famous European companies and governments organizations. You can check their references on their website:

- http://www.eyepea.eu/customers
- http://www.eyepea.eu/company/news

Moreover, ALLOcloud is a cloud solution for SMEs, it handles several millions of calls by month.

If you also use Panoramisk on production, don't hesitate to open a pull request to add your company's name with some details.

CHAPTER 4

CHANGES

4.1 1.5 (unreleased)

- Action is now both a Future and an async iterator

4.2 1.4 (2021-08-05)

- py38 support
- no longer use *yield from* syntaxe.
- Added custom options for connect method: run_forever, on_startup, on_shutdown, reconnect_timeout
- Make ping interval and ping delay more configurable. Callbacks on_login, on_connect and on_disconnect added for use in metrics
- Avoid double exception logging because log.exception() already logs the stacktrace
- Multiple FastAGI support improvement

4.3 1.3 (2018-09-21)

- py37 support

4.4 1.2 (2018-05-24)

- Fix KeyError problem in call_manager
- as_list is now default to none and is used first in Action.multi
- improve AMI version detection

4.5 1.1 (2016-12-29)

- We no longer loosing calls on reconnection
- End of support for python<3.4
- Better test coverage

4.6 1.0 (2015-08-19)

- Add FastAGI server, implemented with high-level AsyncIO API (Streams)
- Add events parameter for Manager, to disable AMI events reception (Useful for performance reasons)
- Finish AsyncAGI commands support (WIP)

4.7 0.6 (2014-11-16)

- Avoid failure when factory is not already set

4.8 0.5 (2014-11-16)

- AMI command results tracking fixed
- Return AMI command result with multiple events in a Future
- Return AsyncAGI command result in a Future
- Add several examples
- Internal refactoring
- Remove arawman support
- Remove external dependencies
- Add support for multiple responses from Actions (example: QueueStatus)
- Improved performance with Events pattern matching
- Add mocked test wrapper
- Add coroutine support for Events dispatching
- Invert event callback signature to create Manager methods to handle events
- Support of AMI commands
- Support of AsyncAGI commands (Not finished)

4.9 0.4 (2014-05-30)

- Compat with the latest trollius

4.10 0.3 (2014-01-10)

- Don't send commands twice

4.11 0.2 (2014-01-09)

- Initial release

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index