

---

# **panoramisk Documentation**

***Release 1.4***

**Gael Pasgrimaud**

**Sep 09, 2021**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Detailed Documentation</b>	<b>5</b>
<b>3</b>	<b>Who use Panoramisk on production ?</b>	<b>9</b>
<b>4</b>	<b>CHANGES</b>	<b>11</b>
<b>5</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>





*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 tipping 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
```





## 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 Future will retrieve all responses

**Returns** a Future that will receive the response

**Return type** `asyncio.Future`

#### Example

To retrieve answer in a coroutine:

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

With a callback:

```
manager = Manager()
future = manager.send_action({'Action': 'Status'})
future.add_done_callback(handle_status_response)
```

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](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 = manager.send_command('http show status')
```

Return a response Message. See [https://wiki.asterisk.org/wiki/display/AST/ManagerAction\\_Command](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 FastAPI 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 FastAPI 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 containing 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*)

---

### Who use Panoramisk on production ?

---

For now, mainly Eyepea and ALLOcloud.

You shouldn't know these 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.



### 4.1 1.4 (2021-08-05)

- py38 support
- no longer use *yield from* syntax.
- 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 FastAPI support improvement

### 4.2 1.3 (2018-09-21)

- py37 support

### 4.3 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

### 4.4 1.1 (2016-12-29)

- We no longer loosing calls on reconnection

- End of support for python<3.4
- Better test coverage

## 4.5 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)

## 4.6 0.6 (2014-11-16)

- Avoid failure when factory is not already set

## 4.7 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)

## 4.8 0.4 (2014-05-30)

- Compat with the latest trollius

## 4.9 0.3 (2014-01-10)

- Don't send commands twice



## 4.10 0.2 (2014-01-09)

- Initial release



## CHAPTER 5

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### p

- `panoramisk`, [5](#)
- `panoramisk.fast_agi`, [7](#)
- `panoramisk.testing`, [8](#)



## A

`add_route()` (*panoramisk.fast\_agi.Application method*), 7  
`Application` (*class in panoramisk.fast\_agi*), 7

## C

`close()` (*panoramisk.Manager method*), 5  
`connect()` (*panoramisk.Manager method*), 6

## D

`del_route()` (*panoramisk.fast\_agi.Application method*), 7

## H

`handler()` (*panoramisk.fast\_agi.Application method*), 8

## M

`Manager` (*class in panoramisk*), 5  
`Manager` (*class in panoramisk.testing*), 8

## P

`panoramisk` (*module*), 5  
`panoramisk.fast_agi` (*module*), 7  
`panoramisk.testing` (*module*), 8

## R

`register_event()` (*panoramisk.Manager method*), 6  
`run_forever()` (*panoramisk.Manager method*), 6

## S

`send_action()` (*panoramisk.Manager method*), 6  
`send_agi_command()` (*panoramisk.Manager method*), 6  
`send_command()` (*panoramisk.Manager method*), 7