

---

# **AgentZero Documentation**

*Release 0.4.3*

**Gabriel Falcão**

**Nov 11, 2018**



<b>1 Features:</b>	<b>3</b>
<b>2 Table of Contents:</b>	<b>5</b>
2.1 SocketManager . . . . .	5
2.1.1 create . . . . .	6
2.1.2 get_by_name . . . . .	6
2.1.3 get_or_create . . . . .	6
2.1.4 register_socket . . . . .	7
2.1.5 bind . . . . .	7
2.1.6 ensure_and_bind . . . . .	7
2.1.7 bind_to_random_port . . . . .	8
2.1.8 connect . . . . .	8
2.1.9 ensure_and_connect . . . . .	9
2.1.10 engage . . . . .	9
2.1.11 send_safe . . . . .	10
2.1.12 recv_safe . . . . .	10
2.1.13 recv_event_safe . . . . .	11
2.1.14 subscribe . . . . .	11
2.1.15 set_socket_option . . . . .	12
2.1.16 set_topic . . . . .	13
2.1.17 publish_safe . . . . .	13
2.1.18 ready . . . . .	14
2.1.19 wait_until_ready . . . . .	14
2.1.20 get_log_handler . . . . .	14
2.1.21 get_logger . . . . .	15
2.1.22 close . . . . .	15
2.1.23 Utility Functions . . . . .	16
2.1.24 get_free_tcp_port . . . . .	16
2.1.25 get_default_bind_address . . . . .	16
2.1.26 get_public_ip_address . . . . .	16
2.1.27 extract_hostname_from_tcp_address . . . . .	16
2.1.28 resolve_hostname . . . . .	16
2.1.29 fix_zeromq_tcp_address . . . . .	16
2.1.30 get_public_zmq_address . . . . .	16
2.1.31 seconds_since . . . . .	16
2.1.32 datetime_from_seconds . . . . .	16

2.1.33	serialized_exception	16
2.1.34	Exceptions	17
2.1.35	AgentZeroSocketError	17
2.1.36	SocketAlreadyExists	17
2.1.37	SocketNotFound	17
2.1.38	SocketBindError	17
2.1.39	SocketConnectError	17
2.2	Contributor Code of Conduct	17
2.3	Building a distributed worker pipeline	18
2.3.1	Coding the pipeline entity	19
2.3.2	Coding the step entity	20

AgentZero lets you create, connect, bind, and modify zeromq sockets in runtime with ease.

It works great with gevent, making it possible to create network applications with simple code that performs complex operations.



# CHAPTER 1

---

## Features:

---

- Create labeled sockets, every ZMQ socket in AgentZero has a name.
- seamlessly poll across connected/bound sockets
- seamlessly subscribe to events, which return the boxed type: `Event`
- easily publish events
- bind sockets to random ports automatically
- bind to hostnames, with automatic DNS resolution
- ability to wait until a socket has received data
- builtin python log handler that publishes logs in a ZMQ PUB socket





---

## Table of Contents:

---

### 2.1 SocketManager

```
class agentzero.SocketManager (zmq, context, serialization_backend=None,  
                               polling_timeout=10000, timeout=10)
```

High-level abstraction for zeromq's non-blocking api.

This component provides utility methods to create, retrieve, connect and bind sockets by name.

It can wait for a socket to become available in either receiving data, sending data or both at the same time.

#### Parameters

- **zmq** – a reference to the zmq module (either from `import zmq` or `import zmq.green as zmq`)
- **context** – the context where the sockets will be created
- **serialization\_backend** – an instance of a valid `agentzero.serializers.BaseSerializer`. This is completely optional safe for the cases where you utilize the methods `send_safe` and `recv_safe` when communicating to other nodes.
- **polling\_timeout** – a **float** - how long to wait for the socket to become available, in milliseconds
- **timeout** – default value passed to `engage ()`

---

**Note:** An extra useful feature that comes with using a `SocketManager` is that you can use a `SocketManager` to create an application that dynamically connects to new nodes based on scaling instructions coming from other nodes

---

**Warning:** Always use the same context per each thread. If you are using `gevent`, please using a single instance for your whole main process, across all greenlets that you manage.

```
import zmq
from agentzero.core import SocketManager
from agentzero.serializers import JSON

context = zmq.Context()

sockets = SocketManager(zmq, context, serialization_backend=JSON())
sockets.ensure_and_connect(
    "requester",
    zmq.REQ,
    'tcp://192.168.2.42:5051',
    zmq.POLLIN | zmq.POLLOUT
)
```

## 2.1.1 create

`SocketManager.create` (*name*, *socket\_type*)

Creates a named socket by type. Can raise a `SocketAlreadyExists`.

Returns the socket itself

### Parameters

- **name** – the socket name
- **socket\_type** – a valid socket type (i.e: `zmq.REQ`, `zmq.PUB`, `zmq.PAIR`, ...)

## 2.1.2 get\_by\_name

`SocketManager.get_by_name` (*name*)

Returns an existing socket by name. It can raise a `SocketNotFound` exception.

Returns the socket

**Parameters** **name** – the socket name

## 2.1.3 get\_or\_create

`SocketManager.get_or_create` (*name*, *socket\_type*, *polling\_mechanism*)

ensure that a socket exists and is registered with a given `polling_mechanism` (`POLLIN`, `POLLOUT` or both)

returns the socket itself.

### Parameters

- **name** – the socket name
- **socket\_type** – a valid socket type (i.e: `zmq.REQ`, `zmq.PUB`, `zmq.PAIR`, ...)
- **polling\_mechanism** – one of (`zmq.POLLIN`, `zmq.POLLOUT`, `zmq.POLLIN | zmq.POLLOUT`)

```
import zmq
from agentzero.core import SocketManager
from agentzero.serializers import JSON

context = zmq.Context()
```

```
sockets = SocketManager(zmq, context, serialization_backend=JSON())
sockets.get_or_create(
    "requester",
    zmq.REQ,
    zmq.POLLIN | zmq.POLLOUT
)
```

## 2.1.4 register\_socket

`SocketManager.register_socket` (*socket, polling\_mechanism*)  
 registers a socket with a given *polling\_mechanism* (POLLIN, POLLOUT or both)  
 returns the socket itself.

### Parameters

- **socket** – the socket instance
- **polling\_mechanism** – one of (zmq.POLLIN, zmq.POLLOUT, zmq.POLLIN | zmq.POLLOUT)

## 2.1.5 bind

`SocketManager.bind` (*socket\_name, address, polling\_mechanism*)  
 binds a socket to an address and automatically registers it with the given *polling\_mechanism*.  
 returns the socket itself.

### Parameters

- **socket\_name** – the socket name
- **address** – a valid zeromq address (i.e: inproc://whatevs)
- **polling\_mechanism** – zmq.POLLIN, zmq.POLLOUT or zmq.POLLIN | zmq.POLLOUT

### Example:

```
>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.create('pipe-in', zmq.PULL)
>>> sockets.bind('pipe-in', 'tcp://*:6000', zmq.POLLIN)
```

## 2.1.6 ensure\_and\_bind

`SocketManager.ensure_and_bind` (*socket\_name, socket\_type, address, polling\_mechanism*)  
 Ensure that a socket exists, that is *binded* to the given address and that is registered with the given *polling\_mechanism*.

---

**Tip:** This method is a handy replacement for calling `get_or_create()`, `bind()` and then `engage()`.

---

returns the socket itself.

#### Parameters

- **socket\_name** – the socket name
- **socket\_type** – a valid socket type (i.e: `zmq.REQ`, `zmq.PUB`, `zmq.PAIR`, ...)
- **address** – a valid zeromq address (i.e: `inproc://whatevs`)
- **polling\_mechanism** – `zmq.POLLIN`, `zmq.POLLOUT` or `zmq.POLLIN | zmq.POLLOUT`

### 2.1.7 bind\_to\_random\_port

`SocketManager.bind_to_random_port` (*socket\_name*, *polling\_mechanism*, *local\_address=u'tcp://0.0.0.0'*)

binds the socket to a random port

returns a 2-item tuple with the socket instance and the address string

#### Parameters

- **socket\_name** – the socket name
- **polling\_mechanism** – `zmq.POLLIN`, `zmq.POLLOUT` or `zmq.POLLIN | zmq.POLLOUT`

#### Example:

```
>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.create('api-server', zmq.REP)
>>> _, address = sockets.bind_to_random_port(
...     'api-server',
...     zmq.POLLIN | zmq.POLLOUT,
...     local_address='tcp://192.168.10.24
... )
>>> address
'tcp://192.168.10.24:61432'
```

### 2.1.8 connect

`SocketManager.connect` (*socket\_name*, *address*, *polling\_mechanism*)

connects a socket to an address and automatically registers it with the given polling mechanism.

returns the socket itself.

#### Parameters

- **socket\_name** – the socket name
- **address** – a valid zeromq address (i.e: `inproc://whatevs`)
- **polling\_mechanism** – `zmq.POLLIN`, `zmq.POLLOUT` or `zmq.POLLIN | zmq.POLLOUT`

#### Example:

```

>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.ensure_and_connect(
...     socket_name='logs',
...     zmq.PUB,
...     'tcp://192.168.10.24:6000',
...     zmq.POLLOUT
... )
>>> sockets.publish_safe('logs', 'output', 'some data')

```

## 2.1.9 ensure\_and\_connect

`SocketManager.ensure_and_connect` (*socket\_name, socket\_type, address, polling\_mechanism*)

Ensure that a socket exists, that is *connected* to the given address and that is registered with the given polling mechanism.

---

**Tip:** This method is a handy replacement for calling `get_or_create()`, `connect()` and then `engage()`.

---

returns the socket itself.

### Parameters

- **socket\_name** – the socket name
- **socket\_type** – a valid socket type (i.e: `zmq.REQ`, `zmq.PUB`, `zmq.PAIR`, ...)
- **address** – a valid zeromq address (i.e: `inproc://whatevs`)
- **polling\_mechanism** – `zmq.POLLIN`, `zmq.POLLOUT` or `zmq.POLLIN | zmq.POLLOUT`

### Example:

```

>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.ensure_and_connect(
...     socket_name='logs',
...     zmq.REQ,
...     'tcp://192.168.10.24:7000',
...     zmq.POLLIN | zmq.POLLOUT
... )

```

## 2.1.10 engage

`SocketManager.engage` (*timeout=None*)

polls all registered sockets with the given timeout in milliseconds

returns a dictionary with the sockets that are ready to be used in their respective state (`zmq.POLLIN` or `zmq.POLLOUT`)

**Parameters** `timeout` – how long should it poll until a socket becomes available. defaults to `agentzero.core.DEFAULT_POLLING_TIMEOUT`

### 2.1.11 `send_safe`

`SocketManager.send_safe(name, data, *args, **kw)`

serializes the data with the configured `serialization_backend`, waits for the socket to become available, then sends it over through the provided socket name.

returns `True` if the message was sent, or `False` if the socket never became available.

---

**Note:** you can safely use this function without waiting for a socket to become ready, as it already does it for you.

---

raises `SocketNotFound` when the socket name is wrong.

#### Parameters

- **name** – the name of the socket where data should be sent through
- **data** – the data to be serialized then sent
- **\*args** – args to be passed to `wait_until_ready`
- **\*\*kw** – kwargs to be passed to `wait_until_ready`

### 2.1.12 `recv_safe`

`SocketManager.recv_safe(name, *args, **kw)`

waits for the socket to become available then receives data through it and deserializes the result using the configured `serialization_backend` before returning.

---

**Note:** you can safely use this function without waiting for a socket to become ready, as it already does it for you.

---

raises `SocketNotFound` when the socket name is wrong.

returns the deserialized data, or `None` if the socket never became available

#### Parameters

- **name** – the name of the socket where data will pad through
- **\*args** – args to be passed to `wait_until_ready`
- **\*\*kw** – kwargs to be passed to `wait_until_ready`

#### Example:

```
>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.ensure_and_bind('pipe-in', zmq.PULL, 'tcp://*:6000', zmq.POLLIN)
>>> sockets.recv_safe('pipe-in')
{
    "pipeline": "video-download",
```

```

"instructions": {
  "url": "https://www.youtube.com/watch?v=FPZ6mVsv4EI"
}

```

### 2.1.13 recv\_event\_safe

SocketManager.**recv\_event\_safe** (*name*, *topic=False*, *\*args*, *\*\*kw*)

waits for the socket to become available then receives multipart data assuming that it's a pub/sub event, thus it parses the topic and the serialized data, then it deserializes the result using the configured `serialization_backend` before returning.

---

**Note:** you can safely use this function without waiting for a socket to become ready, as it already does it for you.

---

raises `SocketNotFound` when the socket name is wrong.

returns the deserialized data, or `None` if the socket never became available

#### Parameters

- **name** – the name of the socket where data will pad through
- **\*args** – args to be passed to `wait_until_ready`
- **\*\*kw** – kwargs to be passed to `wait_until_ready`

#### Example:

```

>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.ensure_and_bind('events', zmq.SUB, 'tcp://*:6000', zmq.POLLIN)
>>>
>>> # subscribe only to topics beginning with "logs"
>>> sockets.set_topic('events', 'logs')
>>> event = sockets.recv_event_safe('events')
>>> event.topic, event.data
'logs:2016-06-20', {'stdout': 'hello world'}

```

### 2.1.14 subscribe

SocketManager.**subscribe** (*name*, *topic=None*, *keep\_polling=None*, *\*args*, *\*\*kw*)

waits for the socket to become available then receives data through it and deserializes the result using the configured `serialization_backend` before returning.

---

**Note:** you can safely use this function without waiting for a socket to become ready, as it already does it for you.

---

raises `SocketNotFound` when the socket name is wrong.

returns an `:py:class:`~agentzero.core.Event``, or `None` if the socket never became available

**Parameters**

- **name** – *str* - the name of the socket where data will pad through
- **topic** – *str* - the name of the socket where data will pad through
- **keep\_polling** – (*optional*) - a callable that must return *bool*
- **\*args** – args to be passed to `wait_until_ready`
- **\*\*kw** – kwargs to be passed to `wait_until_ready`

---

**Tip:** pass a function to the **keep\_polling** to control the finality of the loop

---

**Example:**

```
>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.ensure_and_bind('logs', zmq.SUB, 'tcp://*:6000', zmq.POLLIN)
>>> for topic, data in sockets.subscribe('logs', 'output'):
...     print topic, '==>', data
...
output:0 ==> some data
output:1 ==> more data
...
```

### 2.1.15 set\_socket\_option

`SocketManager.set_socket_option` (*name, option, value*)  
calls `zmq.setsockopt` on the given socket.

**Parameters**

- **name** – the name of the socket where data will pad through
- **option** – the option from the `zmq` module
- **value** –

Here are some examples of options:

- `zmq.HWM`: Set high water mark
- `zmq.AFFINITY`: Set I/O thread affinity
- `zmq.IDENTITY`: Set socket identity
- `zmq.SUBSCRIBE`: Establish message filter
- `zmq.UNSUBSCRIBE`: Remove message filter
- `zmq.SNDBUF`: Set kernel transmit buffer size
- `zmq.RCVBUF`: Set kernel receive buffer size
- `zmq.LINGER`: Set linger period for socket shutdown
- `zmq.BACKLOG`: Set maximum length of the queue of outstanding connections
- for the full list go to <http://api.zeromq.org/4-0:zmq-setsockopt>



**Example:**

```
>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.create('pipe-in', zmq.PULL)
>>>
>>> # block after 10 messages are queued
>>> sockets.set_socket_option('pipe-in', zmq.HWM, 10)
```

## 2.1.16 set\_topic

`SocketManager.set_topic(name, topic)`

shortcut to `SocketManager.set_socket_option()` with `(name, zmq.SUBSCRIBE, topic)`

**Parameters**

- **name** – the name of the socket where data will pad through
- **topic** – the option from the `zmq` module

**Example:**

```
>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.ensure_and_bind('events', zmq.SUB, 'tcp://*:6000', zmq.POLLIN)
>>>
>>> # subscribe only to topics beginning with "logs"
>>> sockets.set_topic('events', 'logs')
>>> event = sockets.recv_event_safe('events')
>>> event.topic, event.data
'logs:2016-06-20', {'stdout': 'hello world'}
```

## 2.1.17 publish\_safe

`SocketManager.publish_safe(name, topic, data)`

serializes the data with the configured `serialization_backend`, waits for the socket to become available, then sends it to the given topic through `socket.send_multipart`.

returns `True` if the message was sent, or `False` if the socket never became available.

---

**Note:** you can safely use this function without waiting for a socket to become ready, as it already does it for you.

---

raises `SocketNotFound` when the socket name is wrong.

**Parameters**

- **name** – the name of the socket where data should be sent through
- **topic** – the name of the topic
- **data** – the data to be serialized then sent

### 2.1.18 ready

`SocketManager.ready` (*name*, *polling\_mechanism*, *timeout=None*)

Polls all sockets and checks if the socket with the given name is ready for either `zmq.POLLIN` or `zmq.POLLOUT`.

returns the socket if available, or `None`

#### Parameters

- **socket\_name** – the socket name
- **polling\_mechanism** – either `zmq.POLLIN` or `zmq.POLLOUT`
- **timeout** – the polling timeout in milliseconds that will be passed to `zmq.Poller().poll()` (optional, defaults to `core.DEFAULT_POLLING_TIMEOUT`)

### 2.1.19 wait\_until\_ready

`SocketManager.wait_until_ready` (*name*, *polling\_mechanism*, *timeout=None*, *polling\_timeout=None*)

Briefly waits until the socket is ready to be used, yields to other greenlets until the socket becomes available.

returns the socket if available within the given timeout, or `None`

#### Parameters

- **socket\_name** – the socket name
- **polling\_mechanism** – either `zmq.POLLIN` or `zmq.POLLOUT`
- **timeout** – the timeout in seconds (accepts float) in which it should wait for the socket to become available (optional, defaults to `core.DEFAULT_TIMEOUT_IN_SECONDS`)
- **polling\_timeout** – the polling timeout in milliseconds that will be passed to `zmq.Poller().poll()` (optional, defaults to `core.DEFAULT_POLLING_TIMEOUT`)

### 2.1.20 get\_log\_handler

`SocketManager.get_log_handler` (*socket\_name*, *topic\_name=u'logs'*)

returns an instance of `ZMQPubHandler` attached to a previously-created socket.

#### Parameters

- **socket\_name** – the name of the socket, previously created with `SocketManager.create()`
- **topic\_name** – the name of the topic in which the logs will be PUBLISHED

#### Example:

```
import logging
import zmq.green as zmq
from agentzero.core import SocketManager

context = zmq.Context()
sockets = SocketManager(zmq, context)

handler = sockets.get_log_handler('logs', topic_name='app_logs')
logger = logging.getLogger()
logger.addHandler(handler)
```

```
logger.info("Server is up!")
```

## 2.1.21 get\_logger

`SocketManager.get_logger` (*socket\_name*, *topic\_name=u'logs'*, *logger\_name=None*)  
returns an instance of `Logger` that contains a `ZMQPubHandler` attached to.

### Parameters

- **socket\_name** – the name of the socket, previously created with `create()`
- **topic\_name** – (optional) the name of the topic in which the logs will be PUBLISHED, defaults to “logs”
- **logger\_name** – (optional) defaults to the given socket name

### Example:

```
import logging
import zmq.green as zmq
from agentzero.core import SocketManager

context = zmq.Context()
sockets = SocketManager(zmq, context)

logger = sockets.get_logger('logs', topic_name='logs', logger_name=__name__)
logger.info("Server is up!")
```

## 2.1.22 close

`SocketManager.close` (*socket\_name*)  
closes a socket if it exists

### Parameters

- **socket\_name** – the socket name
- **address** – a valid zeromq address (i.e: `inproc://whatevs`)
- **polling\_mechanism** – `zmq.POLLIN`, `zmq.POLLOUT` or `zmq.POLLIN | zmq.POLLOUT`

### Example:

```
>>> import zmq
>>> from agentzero.core import SocketManager
>>>
>>> sockets = SocketManager()
>>> sockets.create('logs', zmq.SUB)
>>> sockets.bind('logs', 'tcp://*:6000', zmq.POLLIN)
>>> sockets.close('logs')
```

## 2.1.23 Utility Functions

### 2.1.24 `get_free_tcp_port`

`agentzero.util.get_free_tcp_port()`  
returns a TCP port that can be used for listen in the host.

### 2.1.25 `get_default_bind_address`

`agentzero.util.get_default_bind_address()`

### 2.1.26 `get_public_ip_address`

`agentzero.util.get_public_ip_address(hostname=None)`

### 2.1.27 `extract_hostname_from_tcp_address`

`agentzero.util.extract_hostname_from_tcp_address(address)`

### 2.1.28 `resolve_hostname`

`agentzero.util.resolve_hostname(hostname)`

### 2.1.29 `fix_zeromq_tcp_address`

`agentzero.util.fix_zeromq_tcp_address(address)`

### 2.1.30 `get_public_zmq_address`

`agentzero.util.get_public_zmq_address()`

### 2.1.31 `seconds_since`

`agentzero.util.seconds_since(timestamp)`

### 2.1.32 `datetime_from_seconds`

`agentzero.util.datetime_from_seconds(timestamp)`

### 2.1.33 `serialized_exception`

`agentzero.util.serialized_exception(e)`

## 2.1.34 Exceptions

### 2.1.35 AgentZeroSocketError

**exception** `agentzero.errors.AgentZeroSocketError`  
Base exception class for errors originated in `SocketManager`

### 2.1.36 SocketAlreadyExists

**exception** `agentzero.errors.SocketAlreadyExists` (*manager, socket\_name*)  
raised by `SocketManager` when trying to create a named socket that already exists

```
>>> from agentzero.core import zmq
>>> from agentzero.core import SocketManager
>>> sockets = SocketManager()
>>> sockets.create('foo', zmq.REP)
>>> sockets.create('foo', zmq.REP)
Traceback (most recent call last):
...
SocketAlreadyExists: SocketManager(sockets=['foo']) already has a socket named
↳ 'foo'.
```

### 2.1.37 SocketNotFound

**exception** `agentzero.errors.SocketNotFound` (*manager, socket\_name*)  
raised by `SocketManager` when trying to retrieve an unexisting socket

```
>>> from agentzero.core import zmq
>>> from agentzero.core import SocketManager
>>> sockets = SocketManager()
>>> sockets.get_by_name('some-name', zmq.PUB)
Traceback (most recent call last):
...
SocketNotFound: SocketManager(sockets=[]) has no sockets named 'some-name'.
```

### 2.1.38 SocketBindError

**exception** `agentzero.errors.SocketBindError`  
raised by `SocketManager` when a `bind()` operation fails.

### 2.1.39 SocketConnectError

**exception** `agentzero.errors.SocketConnectError`  
raised by `SocketManager` when a `connect()` operation fails.

## 2.2 Contributor Code of Conduct

As contributors and maintainers of these projects, and in the interest of fostering an open and welcoming community, we pledge to respect all people who contribute through reporting issues, posting feature requests, updating documentation, submitting pull requests or patches, and other activities.

We are committed to making participation in these projects a harassment-free experience for everyone, regardless of level of experience, gender, gender identity and expression, sexual orientation, disability, personal appearance, body size, race, ethnicity, age, religion, or nationality.

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery
- Personal attacks
- Trolling or insulting/derogatory comments
- Public or private harassment
- Publishing other's private information, such as physical or electronic addresses, without explicit permission
- Other unethical or unprofessional conduct.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct. By adopting this Code of Conduct, project maintainers commit themselves to fairly and consistently applying these principles to every aspect of managing this project. Project maintainers who do not follow or enforce the Code of Conduct may be permanently removed from the project team.

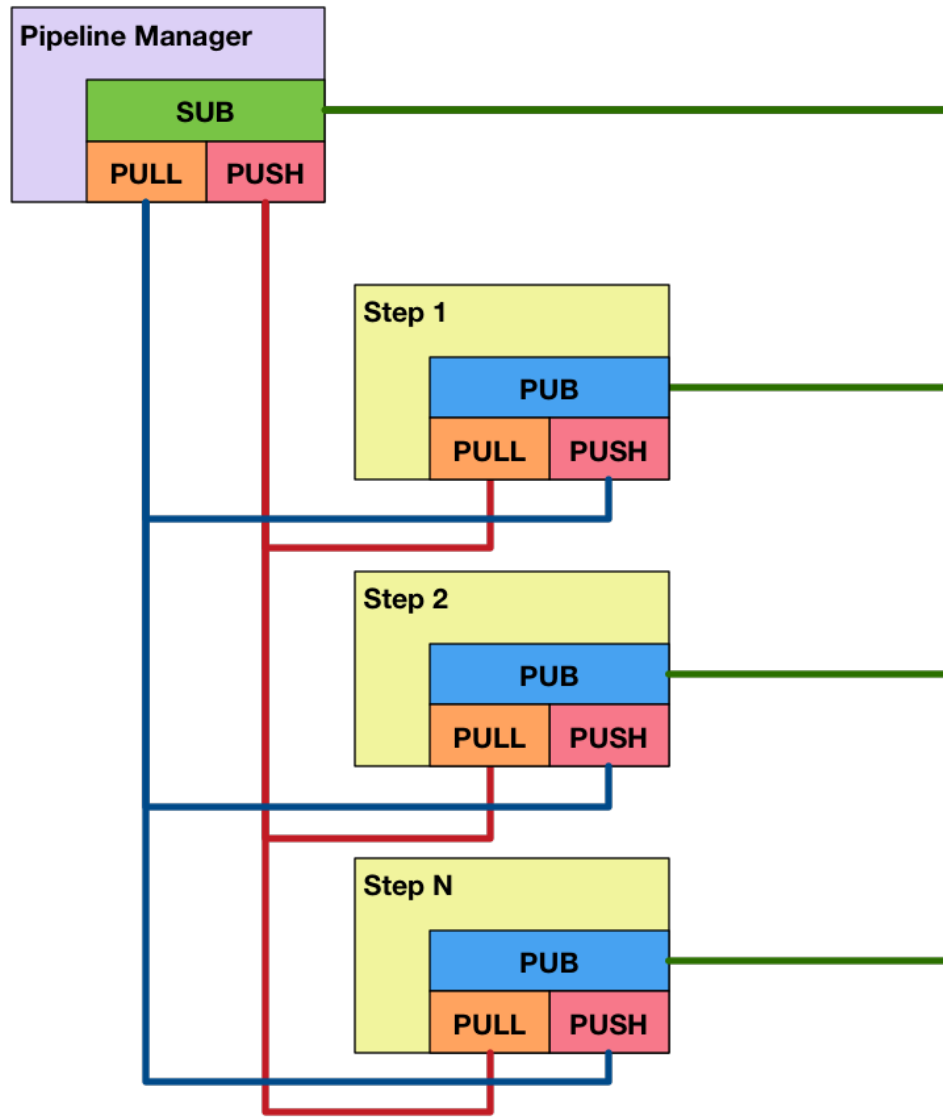
This code of conduct applies both within project spaces and in public spaces when an individual is representing the project or its community.

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by opening an issue or contacting one or more of the project maintainers.

This Code of Conduct is adapted from the [Contributor Covenant](http://contributor-covenant.org/version/1/2/0/), version 1.2.0 available at <http://contributor-covenant.org/version/1/2/0/>.

## 2.3 Building a distributed worker pipeline

Let's build a worker pipeline where `Steps` that will execute specific job types, and can be scaled individually.



Here is an overview of the socket architecture:

### 2.3.1 Coding the pipeline entity

The main pipeline contains the following sockets:

- a SUB socket where it will `bind()` at the given `bind_address` and subscribe to Step events
- a REP socket where it will respond to pipeline execution `bind()` at the given `reply_address` and reply with a job id for later status querying
- a REP socket where it will respond to pipeline execution `bind()` at the given `reply_address`

```

1 import zmq.green as zmq
2 from agentzero.core import SocketManager
3
4
5 class Pipeline(object):
6     steps = []
7 
```

```
8     def __init__(self):
9         self.context = zmq.Context()
10        self.sockets = SocketManager(zmq, self.context)
11        self.sockets.create("pipe-sub", zmq.SUB)
12        self.sockets.create("pipe-in", zmq.PULL)
13        self.sockets.create("pipe-out", zmq.PUSH)
14        self.children = []
```

### 2.3.2 Coding the step entity

A Step contains a PUB socket where it will send the following events:

- announce its JobType as well as its PUSH/PULL address pair
- announce failed jobs, so that they can be auto-recovered later
- announce succeeded jobs
- announce exceptions and auto-schedule a later retry
- live metrics
- live logging output



**A**

AgentZeroSocketError, 17

**B**

bind() (agentzero.SocketManager method), 7

bind\_to\_random\_port() (agentzero.SocketManager method), 8

**C**

close() (agentzero.SocketManager method), 15

connect() (agentzero.SocketManager method), 8

create() (agentzero.SocketManager method), 6

**D**

datetime\_from\_seconds() (in module agentzero.util), 16

**E**

engage() (agentzero.SocketManager method), 9

ensure\_and\_bind() (agentzero.SocketManager method), 7

ensure\_and\_connect() (agentzero.SocketManager method), 9

extract\_hostname\_from\_tcp\_address() (in module agentzero.util), 16

**F**

fix\_zeromq\_tcp\_address() (in module agentzero.util), 16

**G**

get\_by\_name() (agentzero.SocketManager method), 6

get\_default\_bind\_address() (in module agentzero.util), 16

get\_free\_tcp\_port() (in module agentzero.util), 16

get\_log\_handler() (agentzero.SocketManager method), 14

get\_logger() (agentzero.SocketManager method), 15

get\_or\_create() (agentzero.SocketManager method), 6

get\_public\_ip\_address() (in module agentzero.util), 16

get\_public\_zmq\_address() (in module agentzero.util), 16

**P**

publish\_safe() (agentzero.SocketManager method), 13

**R**

ready() (agentzero.SocketManager method), 14

recv\_event\_safe() (agentzero.SocketManager method), 11

recv\_safe() (agentzero.SocketManager method), 10

register\_socket() (agentzero.SocketManager method), 7

resolve\_hostname() (in module agentzero.util), 16

**S**

seconds\_since() (in module agentzero.util), 16

send\_safe() (agentzero.SocketManager method), 10

serialized\_exception() (in module agentzero.util), 16

set\_socket\_option() (agentzero.SocketManager method), 12

set\_topic() (agentzero.SocketManager method), 13

SocketAlreadyExists, 17

SocketBindError, 17

SocketConnectError, 17

SocketManager (class in agentzero), 5

SocketNotFound, 17

subscribe() (agentzero.SocketManager method), 11

**W**

wait\_until\_ready() (agentzero.SocketManager method), 14