Agent Development

JEREME HAACK

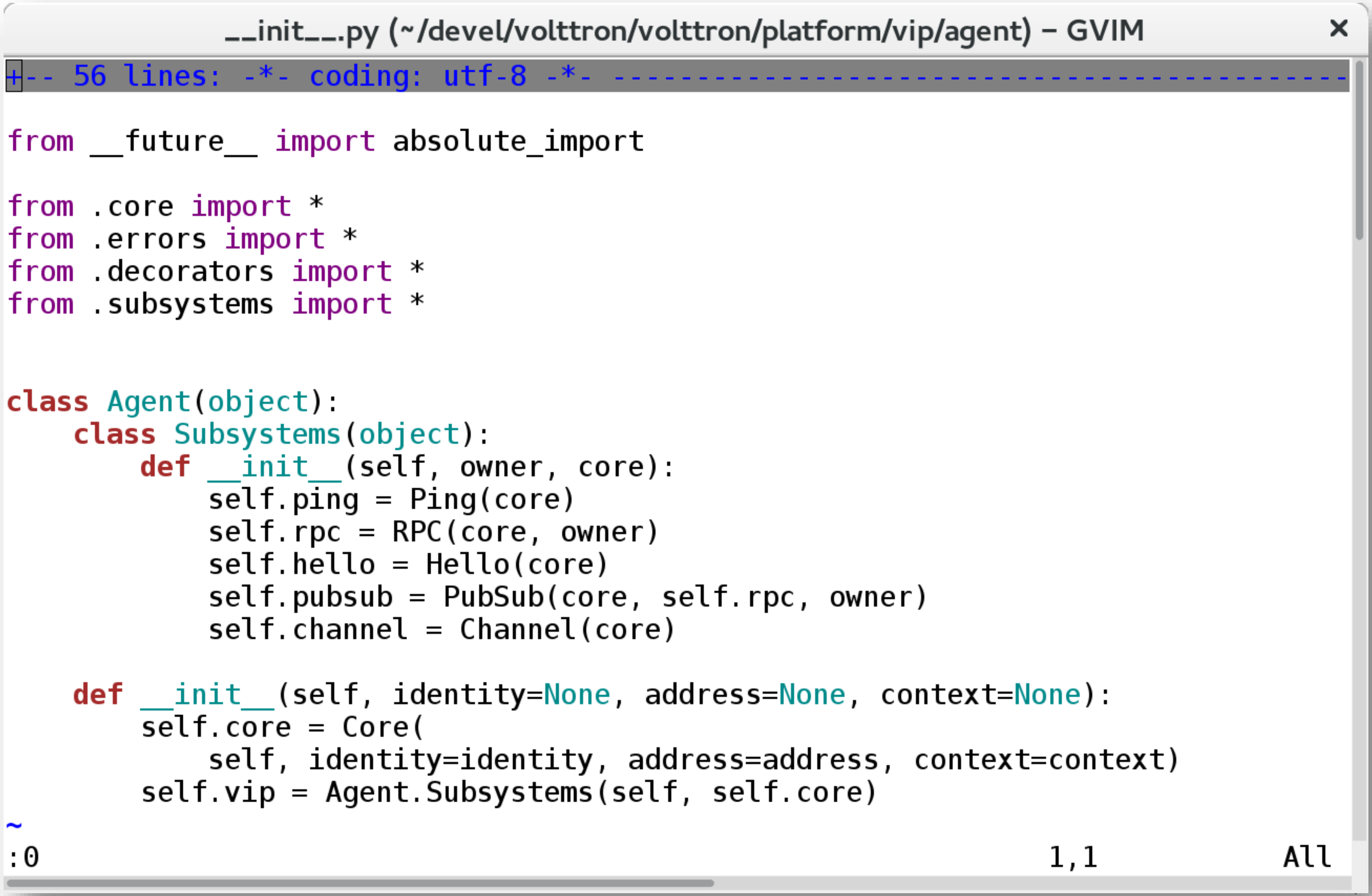
Pacific Northwest National Laboratory

VOLTTRON^{TM} 2016

PNNL-SA-120049

Base Agent Definition

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*



__init__.py (~/devel/volttron/volttron/platform/vip/agent) - GVIM

```
+-- 56 lines: -*- coding: utf-8 -*- ------------------------------------------

from __future__ import absolute_import

from .core import *
from .errors import *
from .decorators import *
from .subsystems import *


class Agent(object):
    class Subsystems(object):
        def __init__(self, owner, core):
            self.ping = Ping(core)
            self.rpc = RPC(core, owner)
            self.hello = Hello(core)
            self.pubsub = PubSub(core, self.rpc, owner)
            self.channel = Channel(core)

    def __init__(self, identity=None, address=None, context=None):
        self.core = Core(
            self, identity=identity, address=address, context=context)
        self.vip = Agent.Subsystems(self, self.core)
~
:0                                                              1,1           All
```

Agent Core

- Main event loop handler and VIP message dispatcher
- Namespace: *Agent.core*
- Methods:
  - *register(name, handler, error_handler)*
    - Register a subsystem handler
  - *run(running_event=None)*
    - Connects VIP socket and starts processing of VIP messages
  - *stop(timeout=None)*
    - Stop the agent (can be called from any context)
  - *send(func, *args, **kwargs)* and *send_async(func, *args, **kwargs)*
    - Send functions from any thread to execute
  - *spawn(func, *args, **kwargs)* and *spawn_in_thread(func, *args, **kwargs)*
    - Spawn function in new greenlet or thread

Agent Core (continued)

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*

- Decorators:
  - *periodic(period, args=None, kwargs=None, wait=0)*
    - Execute a method on a regular interval
  - *schedule(deadline, *args, **kwargs)*
    - Execute a method at a specific time
  - receiver(signal)
    - Register a method as a callback for the named signal
- Signals:
  - *onsetup* – used for instantiation and configuration
    - VIP messaging is not running
    - All receivers run serially
  - *onstart* – used to spawn tasks as VIP loop starts
  - *onstop* – signaled just before VIP loop stops
  - *onfinish* – signaled after VIP loop stops
    - Used for teardown and cleanup

VIP Subsystem: pubsub

- Platform pub/sub service
  - Global service allows for discovery and platform-level messaging
- Message format:
  - Topic
    - UTF-8 encoded string
    - /-separated components
  - Headers
    - JSON serialized dictionary (mapping)
  - Body
    - Zero or more ZeroMQ frames
- Improvements with 3.0
  - Source attribution (not spoofable)
  - Unlimited per-agent buses
  - Decentralized

VIP Subsystem: RPC

- Remote procedure calls via JSON-RPC 2.0
  - Specification at http://www.jsonrpc.org/specification
  - Safe, expressive, simple, well-supported, etc.
  - Supports one-way notifications
- Extended to support simultaneous use of list (*args) and keyword (**kwargs) arguments
- Export agent methods with export() decorator
- Calls handled asynchronously (spawned in own greenlet)
- Calling remote procedure returns AsyncResult
  - Wait for results
  - Set callback to handle results
- Discover exported methods with inspect()
  - Also used to query parameters, return value, documentation, etc.

Other VIP Subsystems

- error
  - Protocol for communicating routing errors
    - EHOSTUNREACH: no route to peer (peer not connected)
    - EAGAIN: temporary failure because of full buffers
- hello
  - Get version and identity (router and peer) information from router
- ping
  - Send ping requests to any agent
- query
  - Query router for properties (e.g. TCP addresses)
- channel
  - Tunnel ZeroMQ frames between agents

VIP Compatible with 2.x Agents

- Compatibility layer
  - Relays 2.x pub/sub messages via VIP
  - Completely modular
  - Can be easily removed
- 2.x agents work without modification
- 2.x legacy support will be removed in subsequent release

gevent for Cooperative Multitasking

- According to gevent.org:

gevent is a coroutine-based Python networking library that uses greenlet to provide a high-level synchronous API on top of the libev event loop.

Features include:

- Fast event loop based on libev (epoll on Linux, kqueue on FreeBSD).
- Lightweight execution units based on greenlet.
- API that re-uses concepts from the Python standard library (for example there are Events and Queues).
- Cooperative sockets with SSL support
- DNS queries performed through threadpool or c-ares.
- Monkey patching utility to get 3rd party modules to become cooperative

Tips for Using gevent

- gevent is cooperative
  - Greenlet's own the thread until explicitly relinquished or blocking operation
  - Must use gevent-aware routines when blocking
  - Locking is not necessary
  - Use *gevent.sleep(0)* to yield thread
- Use caution when using gevent across threads
  - Agent core has methods to assist
    - *send()*, *send_async()*, and *spawn_in_thread()*
  - This use case is rare
  - Can monkey-patch threading module
- When reading files use gevent.fileobject.FileObject proxy
- Use zmq.green in place of zmq
  - *from zmq import green as zmq*
- Provides socket, ssl, select, etc. modules

Publish/Subscribe

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*

- Decorator
  - *@PubSub.subscribe('pubsub', ‘devices/campus/building/device/point‘)*
- Callback
  - *self.vip.pubsub.subscribe(peer='pubsub',prefix=” devices/campus/building/device/point‘”,callback=callback_method)*

Status and Alerts

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by **Battelle** *Since 1965*

- Agent can send an alert when off-normal event occurs
  - *self.vip.health.send_alert(“Short name”, “Status message”)*
- Platform agent receives alerts
- Actions can be triggered by alerts (email admin)
  - *EmailerAgent*

Agent Creation Walkthrough

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

http://volttron.readthedocs.io/en/develop/devguides/agent_development/Agent-Development.html

Agent Lifecycle

- Build
- Install
- Enable
- Start
- Stop
- Remove

VOLTTRON^{TM} Resources

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle *Since 1965*

- GitHub
  - https://github.com/VOLTTRON/volttron.git
- Email: volttron@pnnl.gov
- Bi-weekly office hours, email to be added
  - http://bgintegration.pnnl.gov/volttronofficehours.asp