Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

VOLTTRON: Development Primer

BRANDON CARPENTER
JEREME HAACK
Pacific Northwest National Laboratory

DOE Building Technologies Office: Technical Meeting on Software Framework for Transactive Energy
July 23-24, 2014

Installing the Platform

- Clone project from Git: https://github.com/VOLTTRON/volttron.git
- Run: ./bootstrap
  - Can take some time
- Launch platform
- Build and deploy ListenerAgent
  - volttron/scripts/build-agent.sh ListenerAgent
  - chmod +x Agents/listeneragent-0.1-py2.7.egg
  - bin/volttron-ctrl install-executable Agents/listeneragent-0.1-py2.7.egg
  - bin/volttron-ctrl load-agent Agents/ListenerAgent/listeneragent.launch.json
  - bin/volttron-ctrl start-agent listeneragent.launch.json
  - To see that it's running: bin/volttron-ctrl list-agents
  - Publishes a heartbeat message
  - Subscribes to all messages

https://github.com/VOLTTRON/volttron/wiki/BuildingTheProject

Drivers

```
Point Name,PNNL Point Name,Units,Units Details,Modbus Register,Writable,Point Address,Notes
CO2Sensor,ReturnAirCO2,PPM,0.00-2000.00,>f,FALSE,1001,CO2 Reading 0.00-2000.0 ppm
FanSpeed,SupplyFanSpeed,%,0.00 to 100.00,>f,FALSE,1003,Fan speed from drive
Cool1Spd,CoolSupplyFanSpeed1,%,0.00 to 100.00 (75 default),>f,TRUE,1005,Fan speed on cool 1 call
Cool2Spd,CoolSupplyFanSpeed2,%,0.00 to 100.00 (90 default),>f,TRUE,1007,Fan speed on Cool2 Call
DaTemp,DischargeAirTemperature,F,(-)39.99 to 248.00,>f,FALSE,1009,Discharge air reading
CO2Stpt,ReturnAirCO2Stpt,PPM,1000.00 (default),>f,TRUE,1011,Setpoint to enable demand control ventilatio
ESMEconMin,ESMDamperMinPosition,%,0.00 to 100.00 (5 default),>f,TRUE,1013,Minimum damper poistion during
FanPower,SupplyFanPower, kW,0.00 to 100.00,>f,FALSE,1015,Fan power from drive
Heat1Spd,HeatSupplyFanSpeed1,%,0.00 to 100.00 (75 default),>f,TRUE,1017,Fan speed on heat 1 Call
```

- Modbus and BACnet drivers use a CSV file to specify points and names
- Initial BACnet driver can be created using a configuration grabbing utility
- No additional code needed for devices using these protocols

https://github.com/VOLTTRON/volttron/wiki/BacnetDriver
https://github.com/VOLTTRON/volttron/wiki/ModbusDriver

sMAP Driver

```
[report 0]
ReportDeliveryLocation = http://                                                  /backend/add/q

[/]
type = Collection
Metadata/SourceName = Jereme's Test Transactional Network
uuid =

[/datalogger]
type = volttron.drivers.data_logger.DataLogger
interval = 1

[/LBNL]
type = Collection
Metadata/Location/Campus = LBNL

[/LBNL/Building46]
type = Collection
Metadata/Location/Building = Building 46

[/LBNL/Building46/RTU1]
type = volttron.drivers.modbus.Modbus
ip_address =
port=502
slave_id = 8
interval = 60
register_config = /home/kyle/workspace/rtunetwork/volttron/drivers/catalyst372.csv
```

- sMAP Driver to push collected data to historian
- Specifies topics data will be published to on Message Bus

https://github.com/VOLTTRON/volttron/wiki/BacnetDriver
https://github.com/VOLTTRON/volttron/wiki/ModbusDriver

ListenerAgent Walkthough

- ListenerAgent – Simple agent which publishes a heartbeat and listens to all messages. Useful for testing install and listening to agents being developed.
  - BaseAgent
    - Init
    - Setup
    - Loop
    - Etc.
  - PublishMixin
  - Topic Matching
    - Matching.py

```python
class ListenerAgent(PublishMixin, BaseAgent):
    '''Listens to everything and publishes a heartbeat according to the
    heartbeat period specified in the settings module.
    '''

    def __init__(self, config_path, **kwargs):
        super(ListenerAgent, self).__init__(**kwargs)
        self.config = utils.load_config(config_path)

    def setup(self):
        # Demonstrate accessing a value from the config file
        _log.info(self.config['message'])
        self._agent_id = self.config['agentid']
        # Always call the base class setup()
        super(ListenerAgent, self).setup()

    @matching.match_all
    def on_match(self, topic, headers, message, match):
        '''Use match_all to receive all messages and print them out.'''
        _log.debug("Topic: {topic}, Headers: {headers}, "
                   "Message: {message}".format(
                   topic=topic, headers=headers, message=message))

    # Demonstrate periodic decorator and settings access
    @periodic(settings.HEARTBEAT_PERIOD)
    def publish_heartbeat(self):
        '''Send heartbeat message every HEARTBEAT_PERIOD seconds.

        HEARTBEAT_PERIOD is set and can be adjusted in the settings module.
        '''
        now = datetime.utcnow().isoformat(' ') + 'Z'
        headers = {
            'AgentID': self._agent_id,
            headers_mod.CONTENT_TYPE: headers_mod.CONTENT_TYPE.PLAIN_TEXT,
            headers_mod.DATE: now,
        }
        self.publish('heartbeat/listeneragent', headers, now)


def main(argv=sys.argv):
    '''Main method called by the eggsecutable.'''
    try:
        utils.default_main(ListenerAgent,
                           description='Example VOLTTRON Lite™ heartbeat agent',
                           argv=argv)
    except Exception as e:
        _log.exception('unhandled exception')


if __name__ == '__main__':
    # Entry point for script
    try:
        sys.exit(main())
    except KeyboardInterrupt:
        pass
```

https://github.com/VOLTTRON/volttron/wiki/ListenerAgent

Scheduler

```python
def publish_schedule(self):

    headers = {
                'AgentID': self._agent_id,
                'type': 'NEW_SCHEDULE',
                'requesterID': self._agent_id, #The name of the requesting agent.
                'taskID': self._agent_id + "-TASK", #The desired task ID for this task.
                                #It must be unique among all other scheduled tasks.
                'priority': 'LOW', #The desired task priority, must be 'HIGH', 'LOW', or 'LOW_PREEMPT'
            }


    msg = [
            ["campus/building/device1", #First time slot.
             "2014-1-31 12:27:00",     #Start of time slot.
             "2014-1-31 12:29:00"],     #End of time slot.
            ["campus/building/device1", #Second time slot.
             "2014-1-31 11:26:00",     #Start of time slot.
             "2014-1-31 11:30:00"],     #End of time slot.
            ["campus/building/device2", #Third time slot.
             "2014-1-31 12:30:00",     #Start of time slot.
             "2014-1-31 12:32:00"],     #End of time slot.
            #etc...
        ]
    self.publish_json(topics.ACTUATOR_SCHEDULE_REQUEST, headers, msg)
```

- Steps for using Scheduler
  - Publish schedule request
  - Get success/error
  - If success, wait for schedule announce and then take action via Actuator
- Priorties
  - High, Low, Low_Preempt

https://github.com/VOLTTRON/volttron/wiki/ActuatorAgent

Actuator

```
def command_equip(self, point_name, value, timeout=None):
    _log.debug('set_point({}, {}, {})'.format(point_name, value, timeout))
    headers = {
            'Content-Type': 'text/plain',
            'requesterID': agent_id,
    }
    self.publish(topics.ACTUATOR_SET(point=point_name, **rtu_path),
                 headers, str(value))
    try:
        return self.value_queue.wait(timeout)
    except green.Timeout:
        return True
```

- Listen for schedule announce
- Send command
- Listen for error
- Check that value changed

https://github.com/VOLTTRON/volttron/wiki/ActuatorAgent

Weather


```
{
    "agent": {
        "exec": "weatheragent-.1-py2.7.egg --config \"%c\" --sub \"%s\" --pub \"%p\""
    },
    "agentid": "Weather1",
    "poll_time": 600,
    "minute_threshold" : 5,
    "daily_threshold" : 200,
    "zip" : "99352"
}
```

```
weather/location/display_location/elevation, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['121.00000000']
weather/location/local_tz_long, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['America/Los_Angeles']
weather/location/observation_location/all, Headers: Headers({u'From': u'Weather1', u'Content-Type': [u'application/json']}), Message: ['"{\\"cit
weather/location/observation_location/city, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['Cottonwood, Ric
weather/location/observation_location/full, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['Cottonwood, Ric
weather/location/observation_location/elevation, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['407 ft']
weather/location/observation_location/country, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['US']
weather/location/observation_location/longitude, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['-119.30519
weather/location/observation_location/state, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['Washington']
weather/location/observation_location/country_iso3166, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['US']
weather/location/observation_location/latitude, Headers: Headers({u'Content-Type': u'text/plain', u'From': u'Weather1'}), Message: ['46.286606']
```


- Obtain WeatherUnderground key
- Edit settings.py to use that key
- Agent publishes all weather data from the site for a given zipcode
- Serves as an example for a proxy agent

https://github.com/VOLTTRON/volttron/wiki/WeatherAgent

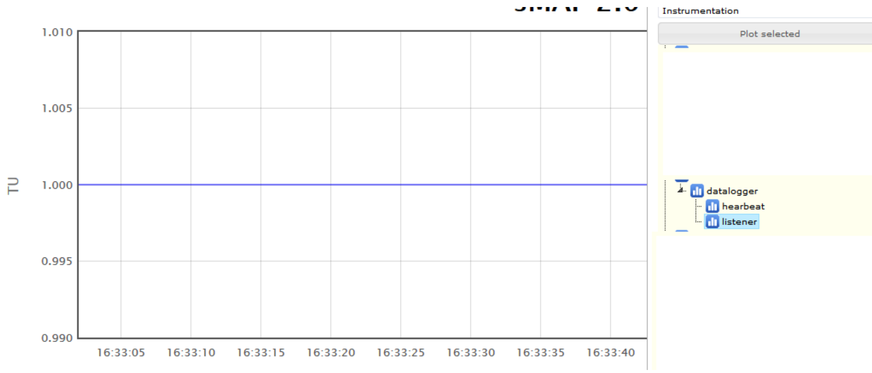
Logging

```python
headers = {}
headers[headers_mod.FROM] = self._agent_id
headers[headers_mod.CONTENT_TYPE] = headers_mod.CONTENT_TYPE.JSON

mytime = int(time.time())

content = {
    "listener": {
        "Readings": [[mytime, 1.0]],
        "Units": "TU",
        "data_type": "double"
    },
    "hearbeat": {
        "Readings": [[mytime, 1.0]],
        "Units": "TU",
        "data_type": "double"
    }
}

self.publish('datalogger/log/', headers, json.dumps(content))
```



- Publish message to the logging topic
    - Topic after /log/ used in point or defaults to /datalogger
- Dynamically creates topics as needed

https://github.com/VOLTTRON/volttron/wiki/Logging

Archiver

- Publish request to archiver topic
  - Point
  - Timerange
  - Note: best to chunk large requests
  - publish('archiver/request/campus1/building1/realcatalyst1/CoolCall1',{},'(now - 1h, now)')
    publish('archiver/request/campus1/building1/realcatalyst1/CoolCall1',{},'(1374192541000.0, 1374193541000.0)')
- Receive answer on response topic in a list of time/value pairs
  - [[1371851254000.0,1.0],[1371851314000.0,1.0],[1371851374000.0,1.0]]

https://github.com/VOLTTRON/volttron/wiki/ArchiverAgent

Multi-Node

```
{
    "agent": {
        "exec": "multibuildingagent-0.1-py2.7.egg --config \"%c\" --sub \"%s\" --pub \"%p\""
    },


    "building-publish-address" : "tcp://0.0.0.0:12201",
    "building-subscribe-address" : "tcp://0.0.0.0:12202",

    "hosts": {"PNNL/hardware1": {"pub": "tcp://xxx.xxx.xxx.xxx:12201",
                                 "sub": "tcp://xxx.xxx.xxx.xxx:12202"},
              "PNNL/hardware2": {"pub": "tcp://yyy.yyy.yyy.yyy:12201",
                                 "sub": "tcp://yyy.yyy.yyy.yyy:12202"},
              "PNNL/hardware3": {"pub": "tcp://zzz.zzz.zzz.zzz:12201",
                                 "sub": "tcp://zzz.zzz.zzz.zzz:12202"},
              "PNNL/hardware4": {"pub": "tcp://aaa.aaa.aaa.aaa:12201",
                                 "sub": "tcp://aaa.aaa.aaa.aaa:12202"}
                        },

     "uuid" : "MultiBuilding"
}
```

- MultiBuildingAgent provides multi-node communication
  - Listens on it's own pub/sub channels and passes messages between platforms
  - Can be encrypted

https://github.com/VOLTTRON/volttron/wiki/MultiBuildingMessaging

VOLTTRON 2.0 Changes

- Optional Features:
  - Resource Management
  - Agent signature validation
- Packing format changed to Wheel
- New configuration file and options
- New command structure

FY15 Plans

- Centralized Management Console
- Enhanced modularization to easily swap out different component implementations
- Supervisory Agent
  - Monitor other platforms
  - Return appliances to a known state in case of abnormal agent condition
- Scalability Study
  - Investigate large scale deployment of platform
- Penetration Testing and Security Enhancements

Resources

- VOLTTRON Resources
  - Wiki: https://github.com/VOLTTRON/volttron/wiki
  - Email: volttron@pnnl.gov
- If you are interested please contact us to be added to:
  - Mailing list: volttron-dev@lyris.pnnl.gov
  - Office hours
    - Twice monthly telecon with VOLTTRON development team