VOLTTRON Community Contributions: Drivers, Utilities, Historians

ChargePoint / Kisensum

August 22, 2018

Kisensum

Kisensum is an energy-management software team

- Primary focus: EV fleet management, microgrids
- Founded by creators of Akuacom (now part of Honeywell), which specialized in smart-building energy management
- Creators of OpenADR, an open standard for automated demand response
- One Kisensum user is LA Air Force Base, where the EV fleet provides services in the frequency-regulation market, charging and discharging EV batteries in response to DNP3 3-second demand signals from CA ISO

Kisensum and VOLTTRON

Why has Kisensum contributed to VOLTTRON?

- Working with Sila Kiliccote's group at SLAC, and with funding and encouragement from DOE, Kisensum has striven to encourage widespread adoption of VOLTTRON as an open-source energy management platform

Recent changes at Kisensum

- Kisensum was acquired on June 25 by ChargePoint, the world's largest and most open EV charging network
- The group formerly known as Kisensum is now ChargePoint Energy Solutions

Key Kisensum Contributions to VOLTTRON in 2018

- modbus-tk driver
- InfluxDB historian agent
- OpenADR Agent
- Jupyter notebooks

modbus-tk device driver

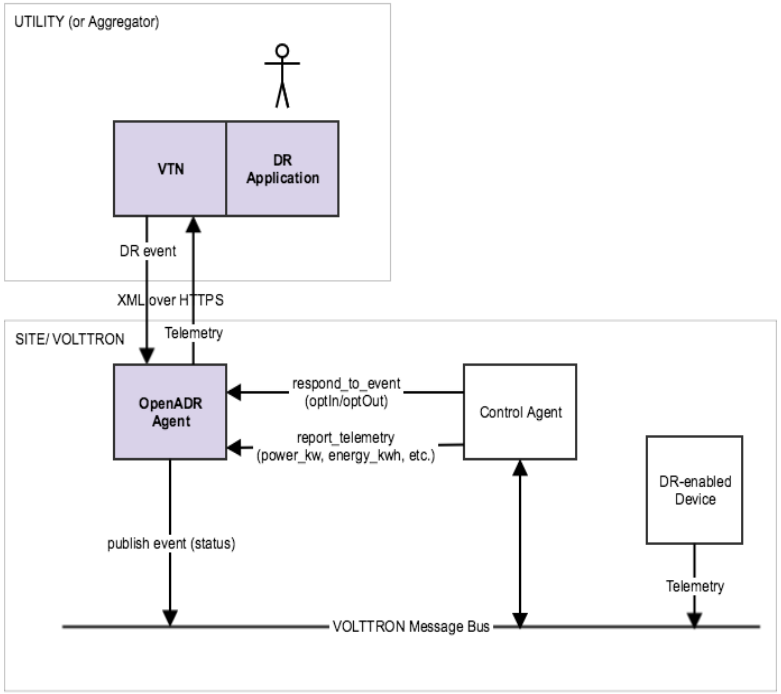
- An alternative to VOLTTRON's original modbus device driver
- Built on the Python modbus-tk library
- Supports Modbus RTU as well as Modbus over TCP/IP
- Optionally uses a map library and config builder to streamline config file maintenance
- Backward-compatible with VOLTTRON modbus driver's .config and .csv file parameter definitions (well, mostly, with obscure exceptions)
- ReadTheDocs: https://volttron.readthedocs.io/en/develop/core_services/drivers/driver_configuration/modbus-tk-driver.html

InfluxDB Historian

- InfluxDB: An open-source time-series database with a fast, scalable engine and high availability
- Often used for DevOps Monitoring (Infrastructure / Application / Cloud Monitoring), IoT Monitoring, and Real-Time Analytics
- This adds a time-series DB to the family of VOLTTRON historians
- ReadTheDocs: https://volttron.readthedocs.io/en/develop/core_services/historians/Influxdb-Historian.html

OpenADR VEN Agent

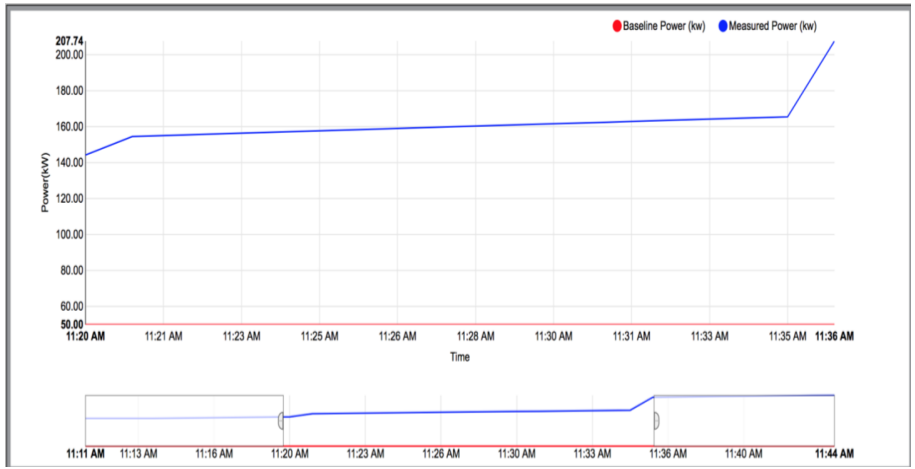
- Alert and respond to demand response events
- OpenADR protocol manages communications between Virtual Top Nodes (VTNs) and Virtual End Nodes (VENs)
- A VOLTTRON agent, OpenADRVenAgent, acts as a VEN, communicating with its VTN using XML over HTTPs
- ReadTheDocs: https://volttron.readthedocs.io/en/develop/core_services/openadr/index.html



OpenADR VTN

Kisensum also contributed source code for the OpenADR interface's other end, a VTN server

- Django 2.0 / Python 3.6
- In volttron/volttron-applications
- Web UI defines and reports on OpenADR events
- VTN "calls an event," indicating time span and load shed level, to request load-shedding based on a pre-committed capacity-bidding agreement
- VEN responds to a called event with an "optIn"
- VEN reports power telemetry



ReferenceApp Agent

- Sample VOLTTRON control agent in volttron-applications
- Manages a set of simulated device drivers
- Responds to OpenADR events by adjusting device behavior (battery charging/discharging)
- Furnishes telemetry to OpenADR VEN agent
- ReadTheDocs: https://volttron.readthedocs.io/en/develop/volttron_applications/Reference-App.html

Jupyter Notebooks

- Jupyter: Open-source web application that creates and shares “notebooks”
- A notebook is displayed in a browser, integrating formatted text with live code
- The Jupyter server executes a notebook’s code on a host; output is displayed and preserved in the notebook
- VOLTTRON notebooks demonstrate how to configure, install and run sets of VOLTTRON agents
- ReadTheDocs: https://volttron.readthedocs.io/en/develop/devguides/supporting/JupyterNotebooks.html

Use notebook_loader to Import Notebooks

```
# Set up local variables vhome and vroot.
# The environment variables VOLTTRON_ROOT and VOLTTRON_HOME should already be defined -- see above.
vroot = %env VOLTTRON_ROOT
approot = vroot + '/applications'
notebooks = vroot + '/examples/JupyterNotebooks'
vhome = %env VOLTTRON_HOME
data_dir = vhome + '/data'
print('VOLTTRON_ROOT={}'.format(vroot))
print('VOLTTRON applications root={}'.format(approot))
print('Jupyter notebooks directory={}'.format(notebooks))
print('VOLTTRON_HOME={}'.format(vhome))
print('VOLTTRON data directory={}'.format(data_dir))

# Enable one Jupyter notebook to import and run the contents of another notebook.
os.chdir(notebooks)
import notebook_loader

# Import notebooks containing utility functions.
import NotebookUtilities
import ConfigureAgents
```

Notebook Utility Methods: sh(), install_agent()

sh()

A "run this shell command" method that wraps subprocess.check_output()

```
import subprocess

def sh(shell_command, shell=True, stderr=None):
    try:
        return_value = subprocess.check_output(shell_command, shell=shell, stderr=stderr)
    except Exception, err:
        print('Shell command failed: {}', shell_command)
        print(err)
        return_value = 'Error'
    return return_value
```

install_agent()

Install a VOLTTRON agent given its directory, id, and config file

```
def install_agent(dir=None, id=None, config=None):
    script_install_command = 'python scripts/install-agent.py -s {0} -i {1} -c {2} -t {3} -f --start'
    sh(script_install_command.format(dir, id, config, id))
    print('Installed and started {} with config {}'.format(id, config))
```

Notebook Example: Install an Agent

Execution: Install and start SimulationClockAgent

SimulatedClockAgent manages a simulation's clock. While a simulation is in progress, it responds to "what time is it?" RPC calls by returning the current simulated time.

```
print('Wait for the message that the agent has been started.')
NotebookUtilities.install_agent(dir=approot+'/kisensum/Simulation/SimulationClockAgent',
                                id='simulationclock',
                                config=approot+'/kisensum/Simulation/SimulationClockAgent/simulationclock.config')
```

Jupyter and VOLTTRON in Docker Containers

As a teaching technique (e.g. for a Hackathon), run VOLTTRON and Jupyter instances in Docker containers on a single host. Each user logs into his/her own container and reads, modifies and executes the notebook's training materials.