VOLTTRON
FEATURE HIGHLIGHTS

September 25, 2019

CHANDRIKA SIVARAMAKRISHNAN
Software Engineer
In this talk

- Highlight some underutilized VOLTTRON features (fan favorites)
- Get feedback about user interests
- Demonstrate the features during afternoon working session
Configuration Store

• Provides storage for agent configurations outside of the agent package
• Alternative to packaging a configuration file with the agent
• Dynamically change configuration without having to restart the agent

Example Use cases:
• Master driver – add or remove a device, update scrapping interval
• Make a historian read only
• Useful to copy configuration of multiple agents across distribution
• You can add the configuration store in a git repository

Add and edit to configuration store using:
  vctl config commands
Configuration Store

• Add and edit to configuration store using volttron-ctl config commands
  ▪ vctl config store <agent vip identity> <configuration name> <infile> --<filetype>
  ▪ vctl config edit <agent vip identity> <configuration name>
  ▪ vctl config list
  ▪ vctl config list <agent vip identity>
  ▪ vctl config get <agent vip identity> <configuration name>
  ▪ vctl config delete <agent vip identity> <configuration name>

• The Platform Configuration Store informs agents of changes to their configuration store

• Agent code should subscribe to configuration store updates and respond to changes accordingly

• Each agent by default has access only to its own configuration store
Protected pubsub topics

• Platform owner can limit who can publish to a given topic
• Protects subscribers on that platform from receiving messages from unauthorized publishers
• To setup:
  ▪ Add topic name to $VOLTTRON_HOME/protected_topics.json
    
    ```json
    { "write-protect": [ {"topic": "foo", "capabilities": ["can_publish_to_foo"]} ] }
    { "write-protect": [ {"topic": "/foo/\.*\", "capabilities": ["can_publish_to_foo"]} ] }
    ```

  ▪ Add the capability “can_publish_to_foo” to the authorized agent, say AgentA, using
    ```shell
    vctl auth add --user_id AgentA --credentials J1GTSqMrf5t3GDYFKMEOkMxkRorrXN-mJ_M_yRv1dBk --capabilities can_publish_to_foo
    ```

  ▪ Now AgentA can publish to topic ”foo”
    ```python
    self.vip.pubsub.publish('pubsub', 'foo', message='Here is a message')
    ```
Protected RPC methods

Protected RPC methods

• Agent’s exposed RPC methods can also be protected using capabilities
  ```python
  @RPC.allow('CAP_SET_TEMP')
  def set_temperature(temp):
  ```

• Can require more than one permission to allow access
  ```python
  @RPC.allow(['CAP_SET_TEMP', 'CAP_FOO_BAR'])
  or
  @RPC.allow('CAP_SET_TEMP')
  @RPC.allow('CAP_FOO_BAR')
  ```

• Platform owner can grant access to a specific agent to access set_temperature by adding the capability/capabilities to the agent’s auth entry
  ```bash
  vctl auth add
  vctl auth update
  ```
Protected RPC methods

• Can also restrict call to methods with only certain runtime parameters
• Example: Configuration Store

```python
@RPC.export
@RPC.allow('edit_config_store')
def manage_store(self, identity, config_name, raw_contents, config_type="raw"):
```

• An agent's (say in the below example agent1) default capability is:

```json
"capabilities": {
    "edit_config_store": { "identity": "agent1" }
}
```

```python
agent1.vip.rpc.call(CONFIGURATION_STORE, 'manage_store',
    "agent1", "config", json_config, config_type="json").get() ✓

agent1.vip.rpc.call(CONFIGURATION_STORE, 'manage_store',
    "agent2", "config", json_config, config_type="json").get() ✗
```
Tagging Service

- Add semantic tags to topics, enabling queries by tags instead of topic name
- Not relying on everyone following a naming convention
- VOLLTRON will use tags from Project Haystack.
- Tags are imported and grouped by categories
- Tag individual components of a topic such as campus, building, device, point etc.
- Add tag to multiple topics based on topic pattern
- Query topics based on tags.
- Example:
  - Give me all the Air Handling Units in Building 1
  - Give me all outdoor air temperatures
  - Give me all power measurements
Agent Template

• Creates the directory structure and a well documented code template to do common tasks in VOLTTRON.

• vpkg init <agent directory name> <agent package name>
  TestAgent/
      ├── setup.py
      │   └── config
      │       └── tester
      │           └── agent.py
      │                   └── __init__.py

• Code auto generated for
  ▪ setup.py
  ▪ Agent initialization
  ▪ Managing subscriptions and RPC calls
  ▪ Agent lifecycle events
  ▪ Configuration store support.
Agent Examples

Useful collection of example agents under volttron/examples

- ListenerAgent
- Standalone Agent
- CSVHistorian

- Integration with other code/applications:
  - ✓ Matlab
  - ✓ C Agent
  - ✓ FNCS
Debugging Agents

- You can start VOLTTRON process and agents within PyCharm IDE in debug mode
- More efficient to debug with break points than adding print statements
- Can also debug pytests from within PyCharm
Running VOLTTRON within PyCharm

- Enable Gevent compatibility
- Import checked out VOLTTRON code
- Set project interpreter as `<volttronsrc>/env/bin/python`
- Launch python code `<volttronsrc>/env/bin/volttron` with working directory = `<volttronsrc>`
Running agent within python

- Run the scripts/pycharm-launch.py and provide the agent's agent.py file as the parameter
- Set environment variables:
  - AGENT_VIP_IDENTITY
  - AGENT_CONFIG
Agent priority

• Agents can be started on VOLTTRON startup
• Agents can start in specific order by assigning priority

• Useful for
  ▪ Dependent agents
  ▪ Example - start historians before master driver

• Set using:
  volttron-ctl enable <AGENT_UUID> <PRIORITY>
  Integer between 0 – 100
  Default : 50
Agent grouping with tags

Tagging Agents:

```
python scripts/install-agent.py -s examples/ListenerAgent --tag listener
volttron-ctl tag <AGENT_UUID> <TAG>
```

Uses: Group one or more agents with a single tag or tag pattern

- Can be used instead of agent uuid in VOLTTRON commands (using --tag)
- Can start or stop multiple agents with single command
  ```
  vctl start --tag historians
  ```
- Can start or stop using tagname_wildcard (--tag historians_*)
  ```
  vctl start --tag historian_*
  ```
Links to documentation

- https://volttron.readthedocs.io/en/develop/devguides/supporting/examples/MatLabAgent.html
- https://project-haystack.org/tag
Thank you