G Node Factory

Jessica Millar

Dec 16, 2022

CONTENTS

The GNodeFactory is an actor in a larger Transactive Energy Management (TEM) system. Within that system, it has the authority for creating and updating GNodes. Among other things, it has the authority for creating and updating `TerminalAssets`, which represent the devices capable of transacting on electricity markets within the TEM.

This repo has been developed through the generous funding of a grant provided by the Algorand Foundation. For more context, please read our Algorand Milestone 1 writeup and Milestone 2 deck. For design specifications for the repo, go here. For a very short description of what GNodes are and why we need a factory for them, skip to Background below.

CHAPTER

ONE

LOCAL DEMO SETUP

PREP

1. Clone this repo

   - Using python 3.10.* or greater, create virtual env inside this repo

   ```
   python -m venv venv
   source venv/bin/activate
   pip install -e .
   ```

2. In sister directories, clone and make virtual envs for these two repos:

   - https://github.com/thegridelectric/gridworks-marketmaker (MarketMaker GNode repo)
   - https://github.com/thegridelectric/gridworks-atn-spaceheat (AtomicTNode GNode repo)

   For now, each of these needs a separate virtual env.

3. Start the Algorand sandbox in a **sibling directory**

   a. Clone Algorand Sandbox into sibling directory

   ```
   - YourDemoFolder
     |
     -- g-node-factory
     -- sandbox
   ```

   b. Start the Algorand sandbox. From the `YourDemoFolder/sandbox` directory

   ```
   ./sandbox up dev
   ```

4. Install docker

RUNNING A SIMULATION OF 4 TERMINAL ASSETS

Note: if your machine is x86, substitute `docker-demo-arm.yml` for `docker-demo-x86.yml` in the instructions below. If you are not sure, try one. If rabbit fails to load try the other.

1. In a terminal for `g-node-factory`, start the dockerized APIS:

   ```
   ./1prep.sh
   ```

   - Check at:
     - http://localhost:8001/docs - Api for the dockerized TaValidator
     - http://localhost:7997/get-time/ - Api for the dockerized TaValidator

2. In that same terminal, start the final gnf API (not in docker yet):

   ```
   ./2prep.sh
   ```

3. In a terminal for `gridworks-marketmaker`:

   ```
   python demo.py
   ```

   - Check that `d1.isone.ver.keene-Fxxx` shows up in rabbitmq passwd/username: smqPublic

4. In a new terminal window for `g-node-factory` repo:

   ```
   python demo.py
   ```

CHAPTER TWO

TESTING

pytest -v

CHAPTER

THREE

CONFIGURATION AND SECRETS

The repo uses dotenv and `.env` files. Look at `src/gnf/config` for default values. These are overwritten with values from a git ignored top-level `.env` file. All dev examples are intended to run without needing to create a `.env` file.

CHAPTER

FOUR

CODE DERIVATION TOOLS

The primary derivation tool used for this is ssot.me, developed by EJ Alexandra of An Abstract Level LLC. All of the xslt code in `CodeGeneration` uses this tool.

The `ssotme` cli and its upstream `ssotme` service pull data from our private airtable down into an odxml file and a json file, and then references local `.xslt` scripts in order to derive code. The `.xslt` allows for two toggles - one where files are always overwritten, and one where the derivation tools will leave files alone once any hand-written code is added. Mostly that toggle is set to `always overwrite` since we are working with immutable schemata. Note that the `ssotme cli` requires an internet connection to work, since it needs to access the upstream `ssotme` service.

If you want to add enums or schema, you will need access to the `ssotme cli` and the airtable. Contact Jessica for this.

CHAPTER FIVE

BACKGROUND

What are GNodes and why do we need a factory for them?

The GNodeFactory stands at the boundary between the physical world and the world of code, maintaining a high fidelity connection between the physical components of real-world electric grids and code objects (GNodes) representing them.

The goal of GNodeFactory is to support transactive devices, especially transactive loads, in taking on the mantle of balancing the electric grid in a renewable future. This requires establishing a link of trust between the the physical reality of a transactive device, and the GNode acting as its online representation. The GNodeFactory does this by issuing NFTs that certify the gps location, metering, and device type of the transactive device prior to activating the corresponding GNode.

This link of trust allows us to redefine demand response.

GNodes come in several flavors (see this enum), and the first flavor to understand is a TerminalAsset.

CHAPTER

SIX

CREDITS

This project was generated from @cjolowicz's Hypermodern Python Cookiecutter template.

6.1 Usage

6.1.1 g-node-factory

G Node Factory.

```
g-node-factory [OPTIONS]
```

Options

--version

Show the version and exit.

6.2 Reference

6.2.1 gnf

G Node Factory.

6.3 Contributor Guide

Thank you for your interest in improving this project. This project is open-source under the MIT license and welcomes contributions in the form of bug reports, feature requests, and pull requests.

Here is a list of important resources for contributors:

- Source Code
- Documentation
- Issue Tracker
- *Code of Conduct*

6.3.1 How to report a bug

Report bugs on the Issue Tracker.

When filing an issue, make sure to answer these questions:

- Which operating system and Python version are you using?
- Which version of this project are you using?
- What did you do?
- What did you expect to see?
- What did you see instead?

The best way to get your bug fixed is to provide a test case, and/or steps to reproduce the issue.

6.3.2 How to request a feature

Request features on the Issue Tracker.

6.3.3 How to set up your development environment

You need Python 3.10+ and the following tools:

- Poetry
- Nox
- nox-poetry

Install the package with development requirements:

```
$ poetry install
```

You can now run an interactive Python session, or the command-line interface:

```
$ poetry run python
$ poetry run g-node-factory
```

6.3.4 How to test the project

Run the full test suite:

```
$ nox
```

List the available Nox sessions:

```
$ nox --list-sessions
```

You can also run a specific Nox session. For example, invoke the unit test suite like this:

```
$ nox --session=tests
```

Unit tests are located in the *tests* directory, and are written using the pytest testing framework.

6.3.5 How to submit changes

Open a pull request to submit changes to this project.

Your pull request needs to meet the following guidelines for acceptance:

- The Nox test suite must pass without errors and warnings.
- Include unit tests. This project maintains 100% code coverage.
- If your changes add functionality, update the documentation accordingly.

Feel free to submit early, though—we can always iterate on this.

To run linting and code formatting checks before committing your change, you can install pre-commit as a Git hook by running the following command:

```
$ nox --session=pre-commit -- install
```

It is recommended to open an issue before starting work on anything. This will allow a chance to talk it over with the owners and validate your approach.

6.4 GNodeFactory Repo Code of Conduct

6.4.1 Basic Truth

All humans are worthy.

6.4.2 Scope

This Code of Conduct applies to moderation of comments, issues and commits within this repository to support its alignment to the above basic truth.

6.4.3 Enforcement Responsibilities

Jessica Millar (jmillar@gridworks-consulting.com) is responsible for clarifying and enforcing this repo's standards of behavior. She has 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, and will communicate reasons for moderation decisions when appropriate.

If you read something in this repo that you want Jessica to consider moderating, please send an email to her at jmillar@gridworks-consulting.com. All complaints will be reviewed and investigated, and Jessica will respect the privacy and security of the reporter of any incident.

6.4.4 What not to add to this repo

Ways to trigger Jessica's moderation enforcement:

- Publish others' private information, such as a physical or email address, without their explicit permission
- Use of sexualized language or imagery, or make sexual advances
- Troll

6.4.5 Suggestions

- Empathize
- Be interested in differing opinions, viewpoints, and experiences
- Give and accept constructive feedback
- Accept responsibility for your mistakes and learn from them
- Recognize everybody makes mistakes, and forgive
- Focus on the highest good for all

6.4.6 Enforcement Escalation

1. Correction

A private, written request from Jessica to change or edit a comment, commit, or issue.

2. Warning

With a warning, Jessica may remove your comments, commits or issues. She may also freeze a conversation.

3. Temporary Ban

A temporary ban from any sort of interaction or public communication within the repository for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

4. Permanent Ban

A permanent ban from any sort of interaction within the repository.

6.4.7 Attribution

This Code of Conduct is loosely adapted from the Contributor Covenant, version 2.1, available at https://www.contributor-covenant.org/version/2/1/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

6.5 License

PYTHON MODULE INDEX

INDEX