Pattoo OPC UA Agents Documentation

Peter Harrison

Aug 01, 2020

Introduction:

`pattoo` agents collect IoT data for a centralized `pattoo` server.

Visit the Pattoo Agents GitHub site to see the code.

CHAPTER 1

Introduction

General information about the project, including the the prerequisite steps to get it operational on your system.

1.1 About Pattoo

`pattoo` allows you to use your web browser to chart your organization's constantly changing data.

It was inspired by the need to collect and visualize data from various DevOps, network, industrial PLC controllers, electro-mechanical and enterprise systems on a single web dashboard.

This data is collected by `pattoo` agents. There are standard agents for:

- Linux
- SNMP
- Modbus TCP
- Bacnet/IP
- OPC UA

With programming skill, you can create your own custom agents if needed.

1.1.1 Operational Overview

`pattoo` has a number of inter-related components. You can see how they all work together on the pattoo web page.

1.1.2 The Palisadoes Foundation

`pattoo` is based on the original `infoset` code created by the Palisadoes Foundation as part of its annual Calico Challenge program. Calico provides paid summer internships for Jamaican university students to work on selected open source projects. They are mentored by software professionals and receive stipends based on the completion of predefined milestones. Calico was started in 2015.

1.2 Basic Installation

This section covers some key steps to get you started.

1.2.1 Prerequisites

There are some software components that need to be installed prior to starting.

1. Install the prerequisite packages for the `easysnmp` python pip package. Instructions can be found here.
2. `pattoo` only runs on Python 3.6 or higher

Let's install the software.

1.2.2 Installation

Follow these steps.

1. Install `git` on your system.
2. Select and create the parent directory in which you want to install `pattoo-agent-opcua`.

```
$ mkdir -p /installation/parent/directory
$ cd /installation/parent/directory
```

3. Clone the repository to the parent directory using the `git clone` command. You can also choose to downloading and unzip the file in the parent directory. The repository can be found at: https://github.com/PalisadoesFoundation/pattoo-agent-opcua

```
$ cd /installation/parent/directory
$ git clone https://github.com/PalisadoesFoundation/pattoo-agent-opcua.git
```

4. Enter the `/installation/parent/directory/pattoo-agent-opcua` directory with the `pattoo-agent-opcua` files.
5. Install the required packages using the `pip_requirements` document in the `pattoo-agent-opcua` root directory

```
$ pip3 install --user --requirement pip_requirements.txt
```

6. Use the *Configuration Guide* to create a working configuration.
7. Follow the configuration steps for each daemon as explained in the *Agent Documentation*.

1.2.3 Configuring systemd Daemons

You can also setup all the `pattoo-agent-opcua` agents as system daemons by executing the `setup/systemd/bin/install_systemd.py` script.

You have to specify a `--config_dir` defining the configuration file directory.

Note The daemons are not enabled or started by default. You will have to do this separately using the `systemctl` command after running the script.

```
$ sudo setup/systemd/bin/install_systemd.py --config_dir ~/GitHub/pattoo-agent-opcua/
↪etc

SUCCESS! You are now able to start/stop and enable/disable the following systemd␣
↪services:

pattoo_agent_os_spoked.service
pattoo_agent_snmpd.service
pattoo_agent_os_autonomousd.service
pattoo_agent_os_hubd.service

$
```

1.3 Configuration Guide

After installation, you will need to create a configuration file in a directory dedicated to `pattoo`.

1.3.1 Setting the Configuration Directory Location

You must first set the location of the configuration directory by using the `PATTOO_CONFIGDIR` environmental variable. Here is how to do this from the Linux command line:

```
$ export PATTOO_CONFIGDIR=/path/to/configuration/directory
```

`pattoo` applications will read the configuration files located in this directory when `PATTOO_CONFIGDIR` is set.

You can automatically set this variable each time you log in by adding these lines to your `~/.bash_profile` file.

```
export PATTOO_CONFIGDIR=/path/to/configuration/directory
```

Make sure that files in this directory are readable by the user that will be running `pattoo` agent daemons or scripts.

1.3.2 Configuration Options

There are two ways to configure `pattoo`. These are the:

1. Quick Method
2. Expert Method

Quick Method

Use the quick method if you are new to `pattoo`.

Run the `setup/configure.py` script. It will prompt you for all configuration parameters. The defaults should be sufficient in most cases.

Here's the command to run:

```
setup/configure.py
```

Next Steps:

1. Run the installation script next as outlined in the *Basic Installation* guide.

2. You will now need to configure each agent individually. See the *Agent Documentation* file for details on how to configure each type of agent.

Expert Method

This section goes into configuration parameters in great detail.

Setting the Configuration Directory Location

You must first set the location of the configuration directory by using the `PATTOO_CONFIGDIR` environmental variable. Here is how to do this from the Linux command line:

```
$ export PATTOO_CONFIGDIR=/path/to/configuration/directory
```

`pattoo` applications will read the configuration files located in this directory when `PATTOO_CONFIGDIR` is set.

You can automatically set this variable each time you log in by adding these lines to your `~/.bash_profile` file.

```
export PATTOO_CONFIGDIR=/path/to/configuration/directory
```

Make sure that files in this directory are readable by the user that will be running `pattoo` agent daemons or scripts.

Copy the Template to Your Configuration Directory

You can create your first `pattoo.yaml` configuration file by copying the template file in the `examples/etc` directory to the `PATTOO_CONFIGDIR` location.

NOTE: If a `/path/to/configuration/directory/pattoo.yaml` file already exists in the directory then skip this step and edit the file according to the steps in following sections.

```
$ cp examples/etc/pattoo.yaml.template \
  /path/to/configuration/directory/pattoo.yaml
```

The next step is to edit the contents of `pattoo.yaml`

Edit Your Configuration

Take some time to read up on `YAML` formatted files if you are not familiar with them. A background knowledge is always helpful.

The `pattoo.yaml` file created from the template will have sections that you will need to edit with custom values. Don't worry, these sections are easily identifiable as they all start with `PATTOO_`

NOTE: The indentations in the YAML configuration are important. Make sure indentations line up. Dashes '-' indicate one item in a list of items (if applicable).

```
pattoo:

    log_level: debug
    log_directory: PATTOO_LOG_DIRECTORY
    cache_directory: PATTOO_CACHE_DIRECTORY
    daemon_directory: PATTOO_DAEMON_DIRECTORY
    system_daemon_directory: PATTOO_SYSTEM_DAEMON_DIRECTORY
```

(continues on next page)

(continued from previous page)

```
    language: en

pattoo_agent_api:

    ip_address: 192.168.1.100
    ip_bind_port: 20201
```

Configuration Explanation

This table outlines the purpose of each configuration parameter

Section	Config Options	Description
pattoo		This section defines the locations of key directories for both operation and troubleshooting
	log_directory	Path to logging directory. Make sure the username running the daemons have RW access to files there.
	log_level	Default level of logging. debug is best for troubleshooting.
	cache_directory	Directory of unsuccessful data posts to pattoo
	daemon_directory	Directory used to store daemon related data that needs to be maintained between reboots
	system_daemon_directory	Directory used to store daemon related data that should be deleted between reboots. This should only be configured if you are running pattoo daemons as systemd daemons. The systemd daemon installation procedure automatically adjusts this configuration. This parameter defaults to the daemon_directory value if it is not configured.
	language	Language spoken by the human users of pattoo. Defaults to en (English)
pattoo_agent_api		This section provides information needed by pattoo agent clients when contacting the pattoo server
	ip_address	IP address of remote pattoo server
	ip_bind_port	Port of remote pattoo server accepting agent data. Default 20201.

Agent Configuration

You will now need to configure each agent individually. See the *Agent Documentation* file for details on how to configure each type of agent.

1.4 Configuring systemd Daemons

You can also setup all the pattoo related daemons located in this GitHub repository as system daemons by executing the setup/systemd/bin/install_systemd.py script.

The script requires you to specify the following parameters. Make sure you have a username and group created for running your pattoo services.

```
usage: install_systemd.py [-h] -f CONFIG_DIR -u USERNAME -g GROUP

optional arguments:
  -h, --help            show this help message and exit
  -f CONFIG_DIR, --config_dir CONFIG_DIR
                        Directory where the pattoo configuration files will be located
  -u USERNAME, --username USERNAME
```

(continues on next page)

(continued from previous page)

```
                        Username that will run the daemon
  -g GROUP, --group GROUP
                        User group to which username belongs
```

Note The daemons are not enabled or started by default. You will have to do this separately using the `systemctl` command after running the script.

```
$ sudo setup/systemd/bin/install_systemd.py --user pattoo --group pattoo --config_dir␣
↪/etc/pattoo

SUCCESS! You are now able to start/stop and enable/disable the following systemd␣
↪services:

pattoo_api_agentd.service
pattoo_apid.service
pattoo_ingesterd.service

$
```

1.5 Backup and Restoration

Always take precautions. Backup your data as you'll never know when you'll need to restore it.

1.5.1 Backup

It is strongly advised that you backup your agents to protect you in the event of catastrophe.

The following directories need to be saved periodically.

1. The `PATTOO_CONFIGDIR` directory which contains your configuration
2. The `daemon_directory` location defined in your configuration. This area stores important authentication information.
3. The `pattoo-agent-opcua` directory which contains your source code.

We'll discuss data restoration next.

1.5.2 Restoration

It's important to follow these steps in this order when restoring `pattoo-agent-opcua` after a disaster.

1. FIRST make sure all the `pattoo` agents are stopped.
2. SECOND restore the contents of the `daemon_directory` location defined in your configuration. This area stores important authentication information.
3. Restore the `PATTOO_CONFIGDIR` directory which contains your configuration
4. Restore `pattoo-agent-opcua` directory which contains your source code.

You should now be able to restart your agents without issue.

1.6 Periodic Jobs

You will need to configure some jobs to improve `pattoo` performance and troubleshooting.

1.6.1 Logrotate Configuration

The default `pattoo` debug logging mode can quickly create large logging files. The logrotate utility can automatically compress and archive them.

1. Copy the the `examples/logrotate.d/pattoo` file to the `/etc/logrotate.d` directory.
2. Edit the file path accordingly.

Read up on the logrotate utility if you are not familiar with it. The documentation is easy to follow.

CHAPTER 2

Agent Setup

How to get the daemons running to collect data.

2.1 Agent Documentation

`pattoo` comes with a number of standard agents, but you can also create your own custom agents to meet your needs. Both approaches are described here.

2.1.1 `pattoo` Standard Agents

Here is a description of currently supported `pattoo` agents.

Agent	Description	Documenatation
`pattoo_agent_opcuad`	Python3 based daemon that polls remote `ip_devices` for OPC UA data.	Documentation can be found here. *Pattoo OPC UA Agents*

2.1.2 Creating Custom Agents

Please visit the Pattoo Shared documentation site to see how it is done.

2.2 Pattoo OPC UA Agents

`pattoo_agent_opcuad` polls `Analog Value` data from OPC UA enabled systems and reports it to the `pattoo` server.

2.2.1 Installation

These steps outline what needs to be done to get `pattoo_agent_opcuad` working.

1. Follow the installation steps in the *Basic Installation* file.
2. Configure the `pattoo.yaml` configuration file following the steps in *Configuration Guide*. This file tells `pattoo_agent_opcuad`, and all other agents, how to communicate with the `pattoo` server.
3. Create a `pattoo_agent_opcuad.yaml` configuration file. Details on how to do this follow.
4. Start the desired daemons as explained in sections to follow. You may want to make these `systemd` daemons, if so follow the steps in the *Basic Installation* file.

2.2.2 Setting the Configuration Directory Location

`pattoo_agent_opcuad` is a standard `pattoo` agent and needs its configuration directory defined by using the `PATTOO_CONFIGDIR` environmental variable. Here is how to do this from the Linux command line:

```
$ export PATTOO_CONFIGDIR=/path/to/configuration/directory
```

`pattoo_agent_opcuad` client will read its own `pattoo_agent_opcuad.yaml` configuration file located this directory when `PATTOO_CONFIGDIR` is set.

You can automatically set this variable each time you log in by adding these lines to your `~/.bash_profile` file.

```
export PATTOO_CONFIGDIR=/path/to/configuration/directory
```

Make sure that files in this directory are readable by the user that will be running standard `pattoo` agent daemons or scripts.

2.2.3 Configuring `pattoo_agent_opcuad.yaml`

Let's get started on configuring `pattoo_agent_opcuad.yaml`. Here is a sample of what should be added. An explanation follows.

NOTE: The indentations in the YAML configuration are important. Make sure indentations line up. Dashes '-' indicate one item in a list of items.

```
polling_interval: 300

polling_groups:

  - group_name: GROUP 1
    ip_target: server-01.opcua.net
    ip_port: 4840
    username: opcua_username
    password: opcua_password
    nodes:
      - address: ns=1;s=[OPCUA_SERVER_1]DischargehAirTemp.PV

  - group_name: GROUP 2
    ip_target: server-02.opcua.net
    ip_port: 4840
    username: opcua_username
    password: opcua_password
```

(continues on next page)

(continued from previous page)

```
    nodes:
      - address: ns=1;s=[OPCUA_SERVER_2]DischargehAirTemp.PV
```

Configuration Explanation

This table outlines the purpose of each configuration parameter

Section	Sub-Section	Description
`polling_interval`		The `pattoo_agent_opcuad` will report to the `pattoo` server every `polling_interval` seconds
`polling_groups`		List of groupings of `ip_devices` that need data from a shared set of OPC UA nodes. Make this the first entry in the configuration sub-section. Make sure it starts with a dash '-' which indicates the beginning of a new grouping.
	`group_name`	Unique name for the set of parameters required to poll an OPC UA `ip_device`
	`ip_device`	The `ip_device` to poll for data
	`ip_port`	The `ip_port` on which the `ip_device` is listening for data
	`username`	The OPC UA `username` to use when querying the `ip_device`
	`password`	The OPC UA `password` to use when querying the `ip_device`
	`nodes`	OPC UA `Analog Value` node to poll for data from for the `ip_devices`. Each `address` must be a OPC UA node. The `multiplier` is the value by which the polled data result must be multiplied. This is useful in converting byte values to bits. The default `multiplier` is 1.

2.2.4 Polling

Use `pattoo_agent_opcuad` to poll your devices. The daemon has a simple command structure below.

You will need a `pattoo_agent_opcuad.yaml` configuration file in the `PATTOO_CONFIGDIR` directory before you start.

```
$ bin/pattoo_agent_opcuad.py --help
usage: pattoo_agent_opcuad.py [-h] [--start] [--stop] [--status] [--restart]
                              [--force]

optional arguments:
  -h, --help  show this help message and exit
  --start     Start the agent daemon.
  --stop      Stop the agent daemon.
  --status    Get daemon daemon status.
  --restart   Restart the agent daemon.
  --force     Stops or restarts the agent daemon ungracefully when used with --stop or
              --restart.
$
```

Use these commands for general operation of the daemon.

Starting

Start the daemon using this command.

```
$ bin/pattoo_agent_opcuad.py --start
```

Stopping

Stop the daemon using this command.

```
$ bin/pattoo_agent_opcuad.py --stop
```

Restarting

Restart the daemon using this command.

```
$ bin/pattoo_agent_opcuad.py --restart
```

Start Polling at Boot

Configuration Guide provides information on how to get the `pattoo_agent_opcuad` daemon to start at boot.

2.2.5 Troubleshooting

Troubleshooting steps can be found in the PattooShared troubleshooting documentation

CHAPTER 3

Miscellaneous Information

Technical background information on the project.

3.1 Troubleshooting Pattoo Agents

Troubleshooting steps can be found in the PattooShared troubleshooting documentation

3.2 JSON Formatting for pattoo-agent-opcua

JSON data formatting can be found in the PattooShared data documentation

3.3 Pattoo Terminology

A complete glossary of terms can be found in the Pattoo Shared glossary documentation

CHAPTER 4

Developers

4.1 How To Contribute

Start contributing today!

4.1.1 Introduction

Below is the workflow for having your contribution accepted into the `pattoo-agent-opcua` repository.

1. Create an Issue or comment on an existing issue to discuss the feature
2. If the feature is approved, assign the issue to yourself
3. Fork the project
4. Clone the fork to your local machine
5. Add the original project as a remote (git remote add upstream https://github.com/PalisadoesFoundation/pattoo-agent-opcua, check with: git remote -v)
6. Create a topic branch for your change (git checkout -b `BranchName`)
7. you may create additional branches if modifying multiple parts of the code
8. Write code and Commit your changes locally. An example of a proper `git commit` message can be seen below:

```
Make the example in CONTRIBUTING imperative and concrete ...

Without this patch applied the example commit message in the CONTRIBUTING
document is not a concrete example.  This is a problem because the
contributor is left to imagine what the commit message should look like
based on a description rather than an example.  This patch fixes the
problem by making the example concrete and imperative.
```

(continues on next page)

(continued from previous page)

```
The first line is a real life imperative statement with a ticket number
from our issue tracker.  The body describes the behavior without the␣
↪patch,
why this is a problem, and how the patch fixes the problem when applied.

Resolves Issue: #123
See also: #456, #789
```

9. When you need to synch with upstream (pull the latest changes from main repo into your current branch), do:
    1. `git fetch upstream`
    2. `git merge upstream/master`
10. Check for unnecessary white space with `git diff --check`.
11. Write the necessary unit tests for your changes.
12. Run all the tests to assure nothing else was accidentally broken
13. Push your changes to your forked repository (git push origin `branch`)
14. Perform a pull request on GitHub
15. Your code will be reviewed
16. If your code passes review, your pull request will be accepted

4.1.2 Code Style Guide

For ease of readability and maintainability code for all `pattoo` projects must follow these guidelines. Code that does not comply will not be added to the `master` branch.

1. All `pattoo` projects use the Google Python Style Guide for general style requirements
2. All `pattoo` python projects use the The Chromium Projects Python Style Guidelines for docstrings.
3. Indentations must be multiples of 4 blank spaces. No tabs.
4. All strings must be enclosed in single quotes
5. In addition too being `pylint` compliant, the code must be PEP8 and PEP257 compliant too.
6. There should be no trailing spaces in files

Guidelines to remember

- Always opt for the most pythonic solution to a problem
- Avoid applying idioms from other programming languages
- Import each module with its full path name. ie: from pack.subpack import module
- Use exceptions where appropriate
- Use doc strings
- Try not to have returns at multiple points in a function unless they are failure state returns.
- If you are in the middle of a development session and have to interrupt your work, it is a good idea to write a broken unit test about what you want to develop next. When coming back to work, you will have a pointer to where you were and get back on track faster.

Commits

The `pattoo` projects strive to maintain a proper log of development through well structured git commits. The links below offer insight and advice on the topic of commit messages:

1. https://robots.thoughtbot.com/5-useful-tips-for-a-better-commit-message
2. http://chris.beams.io/posts/git-commit/

Sample .vimrc File for Compliance

You can use this sample .vimrc file to help meet our style requirements

```
" Activate syntax
syntax on
" set number

" Disable automatic comment insertion
autocmd FileType * setlocal formatoptions-=c formatoptions-=r formatoptions-=o

" Delete trailing whitespace
autocmd BufWritePre * :%s/\s\+$//e

" Convert tabs to spaces
set expandtab

" Set tabs to 4 spaces
set tabstop=4

" Set the number of spaces for indentation
set shiftwidth=4

" Switch on highlighting the last used search pattern when the terminal has colors
if &t_Co > 2 || has("gui_running")
  set hlsearch
endif

" Tell vim to remember certain things when we exit
"  '10  :  marks will be remembered for up to 10 previously edited files
"  "100 :  will save up to 100 lines for each register
"  :20  :  up to 20 lines of command-line history will be remembered
"  %    :  saves and restores the buffer list
"  n... :  where to save the viminfo files
set viminfo='10,\"100,:20,%,n~/.viminfo

" Function for viminfo to work
function! ResCur()
  if line("'\"") <= line("$")
    normal! g`"
    return 1
  endif
endfunction

" Function for viminfo to work
augroup resCur
  autocmd!
  autocmd BufWinEnter * call ResCur()
augroup END
```