pydatacoll Documentation

Release 0.1

jeffchen

January 26, 2016

Contents

PyDataColl is a SCADA-like system which use Python as the main language. It originally inspired by an old program I made as for a core part of a large EMS.

PyDataColl can be roughly divided into three parts:

- An APIServer provides RESTful Services for client to pull/push data from/to devices and perform generic CRUD on devices, terms and items.
- A DeviceManager that manages all devices and terms connected to the system, listens messages send by APIServer that perform CRUD on devices and terms. It may be combined with some plugins to perform generic operation such as data checking, database saving and formula calculation.
- Many devices and terms under control of DeviceManager operate with coded data over communication channels(TCP/IP) so as to provide control of remote equipment(meter or sensor). Each type of Device can communicate with one type of meter with specify protocol, such as Modbus TCP, IEC 60870-5-104.

CHAPTER 1

Installation

Automatic installation:

```
pip install PyDataColl
```

PyDataColl is listed in PyPI and can be installed with `pip` or `easy_install`. Note that the source distribution includes unittest that are not present when PyDataColl is installed in this way, so you may wish to download a copy of the source tarball as well.

Manual installation: Download source code:

```
tar xvzf pydatacoll-0.1.tar.gz
cd pydatacoll-0.1
python setup.py build
sudo python setup.py install
```

The PyDataColl source code is hosted on GitHub.

Prerequisites: PyDataColl runs on Python 3.5+. In addition to the requirements which will be installed automatically by `pip` or `setup.py install`, the following optional packages may be useful:

- Redis is heavily used by PyDataColl as NoSQL databases and IPC. If you deploy PyDataColl in local, make sure you have installed and started the Redis server.
- MySQL is used by DbSaver plugin to store device data in real-time. If you deploy PyDataColl in local, make sure you have installed and started the MySQL server.
- ujson is an ultra fast JSON encoder and decoder written in pure C with bindings for Python. This is an alternative json library and PyDataColl will use it automatically if possible.

CHAPTER 2

Quick Start

1. Simply run the following to start PyDataColl server(add "-h" to see all available parameter):

```
pydatacoll
```

Note: To stop server, press CTRL+C to exit.

2. Visit http://localhost:8080 in browser to see the server information, if success, you will find something like this:

```
PyDataColl is running, available API:
method: GET      URL: http://localhost:8080/
method: GET      URL: http://localhost:8080/api/v1/device_protocols
method: GET      URL: http://localhost:8080/api/v1/devices
(...more omitted)
```

3. The server is running now. You can send request to server with your favorite http client! check RESTful API to see the API list.

Platforms: PyDataColl should run on any Unix-like platform, although for the best performance and scalability only Linux (with `epoll`) and BSD (with `kqueue`) are recommended for production deployment (even though Mac OS X is derived from BSD and supports kqueue, its networking performance is generally poor so it is recommended only for development use). PyDataColl will also run on Windows, although this configuration is not officially supported and is recommended only for development use.

CHAPTER 3

Quick links

- Source (github)
- Docs

CHAPTER 4

Documentation

This documentation is also available in PDF formats.

4.1 User's guide

4.1.1 Introduction

PyDataColl is a SCADA-like system which use Python as the main language. It originally inspired by an old program I made as for a core part of a large EMS.

PyDataColl can be roughly divided into three parts:

- An APIServer provides RESTful Services for client to pull/push data from/to devices and perform generic CRUD on devices, terms and items.
- A DeviceManager that manages all devices and terms connected to the system, listens messages send by APIServer that perform CRUD on devices and terms. It may be combined with some plugins to perform generic operation such as data checking, database saving and formula calculation.
- Many devices and terms under control of DeviceManager operate with coded data over communication channels(TCP/IP) so as to provide control of remote equipment(meter or sensor). Each type of Device can communicate with one type of meter with specify protocol, such as Modbus TCP, IEC 60870-5-104.

4.1.2 ModelStructure

In PyDataColl there are three type of model: **DEVICE**, **TERM**, **ITEM**. their relationship can be shown as this:

```
Device1:
    ->Term1:
        ->Item1
        ->Item2
        ->...
    ->Term2:
        ->Item1
        ->...
    ->...
Device2:
    ->Term3
        ...
```

4.1.3 Device

not done yet:(

4.1.4 Term

not done yet:(

4.1.5 Item

not done yet:(

4.2 Developer's guide

This is paragraph text *before* the table.

Column 1 Column 2 Foo Put two (or more) spaces as a field separator. Bar Even very very long lines like these are fine, as long as you do not put in line endings here

This is paragraph text *after* the table. * sfs

4.3 RESTful API

currently support API:

Method	URL(parameter surrounded by curly braces should replaced by real value)
GET	/
GET	/api/v1/device_protocols
GET	/api/v1/devices
GET	/api/v1/devices/{device_id}
GET	/api/v1/devices/{device_id}/terms
GET	/api/v1/devices/{device_id}/terms/{term_id}/items/{item_id}/datas
GET	/api/v1/devices/{device_id}/terms/{term_id}/items/{item_id}/datas/{index}
GET	/api/v1/items
GET	/api/v1/items/{item_id}
GET	/api/v1/term_protocols
GET	/api/v1/terms
GET	/api/v1/terms/{term_id}
GET	/api/v1/terms/{term_id}/items
GET	/api/v1/terms/{term_id}/items/{item_id}
POST	/api/v1/device_call
POST	/api/v1/device_ctrl
POST	/api/v1/devices
POST	/api/v1/items
POST	/api/v1/terms
POST	/api/v1/terms/{term_id}/items
POST	/api/v2/term_items
PUT	/api/v1/devices/{device_id}
PUT	/api/v1/items/{item_id}
PUT	/api/v1/terms/{term_id}
PUT	/api/v1/terms/{term_id}/items/{item_id}
DELETE	/api/v1/devices/{device_id}
DELETE	/api/v1/items/{item_id}
DELETE	/api/v1/terms/{term_id}
DELETE	/api/v1/terms/{term_id}/items/{item_id}

CHAPTER 5

License

`PyDataColl` is offered under the Apache 2 license.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search