
pydatacoll Documentation

Release 0.1alpha

jeffchen

January 25, 2016

1	Installation	3
2	Quick links	5
3	Documentation	7
3.1	User's guide	7
3.2	Developer's guide	8
3.3	RESTful API	8
4	License	11
5	Indices and tables	13

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](#).

Installation

Prerequisites: PyDataColl runs on Python 3.5+. In addition to the requirements showed in requirements.txt, 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.
 - **hiredis** is a C library which can provide up to a 10x speed improvement in parsing responses from the Redis server. This is an alternative redis client and PyDataColl will use it automatically if possible.
1. Download the latest source code [hosted on github](#), open a shell (*CMD* in Windows, *Terminal* in Linux and Mac OS X) and go to the source code directory, install required package using pip:

```
pip install -r requirements.txt
```

2. Then run the following to start PyDataColl server:

```
python -m pydatacoll.api_server
```

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

3. 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)
```

4. Congratulations! 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.

Quick links

- [Source \(github\)](#)
- [Docs](#)

Documentation

This documentation is also available in [PDF formats](#).

3.1 User's guide

3.1.1 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](#) when I worked as an engineer at GDT(A [SGCC](#) 's subsidiary).

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](#).

3.1.2 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
  ...
```

3.1.3 Device

not done yet:(

3.1.4 Term

not done yet:(

3.1.5 Item

not done yet:(

3.2 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

3.3 RESTful API

currently support API:

Method	URL(parameter surrounded by curly braces should be 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}

License

PyDataColl is offered under the Apache 2 license.

Indices and tables

- `genindex`
- `modindex`
- `search`