PyVISA Documentation

Release 1.9.1

PyVISA Authors

Sep 13, 2018

Contents

PyVISA-py is a backend for PyVISA. It implements most of the methods for Message Based communication (Serial/USB/GPIB/Ethernet) using Python and some well developed, easy to deploy and cross platform libraries.

You can select the PyVISA-py backend using **@py** when instantiating the visa Resource Manager:

```
>>> import visa
>>> rm = visa.ResourceManager('@py')
>>> rm.list_resources()
('USB0::0x1AB1::0x0588::DS1K00005888::INSTR')
>>> inst = rm.open_resource('USB0::0x1AB1::0x0588::DS1K00005888::INSTR')
>>> print(inst.query("*IDN?"))
```

That's all! Except for **@py**, the code is exactly what you would write to using the NI-VISA backend for PyVISA.

CHAPTER 1

Installation

Just run the following command in your console:

```
pip install pyvisa-py
```

You can report a problem or ask for features in the issue tracker. Or get the code in GitHub.

CHAPTER 2

FAQ

2.1 Which libraries are used by PyVISA-py?

It depends on the interface type. For **ASRL** and **USB** we use PySerial and PyUSB respectively. For **TCPIP** we use the `socket` module in the Python Standard Library. **GPIB** resources are not currently supported but they are in the plan using linux-gpib.

PySerial version 3.0 or newer is required.

2.2 If I only need TCPIP, do I need to install PySerial and PyUSB?

No. Libraries are loaded on demand.

2.3 How do I know if PyVISA-py is properly installed?

Using the pyvisa information tool. Run in your console:

```
python -m visa info
```

You will get info about PyVISA, the installed backends and their options.

2.4 Which resource types are supported?

Now:

- ASRL INSTR
- USB INSTR
- TCPIP INSTR

- USB RAW
- TCPIP SOCKET
- GPIB INSTR

2.5 Are all VISA attributes and methods implemented?

No. We have implemented those attributes and methods that are most commonly needed. We would like to reach feature parity. If there is something that you need, let us know.

2.6 Why are you developing this?

The National Instruments's VISA is a proprietary library that only works on certain systems. We wanted to provide a compatible alternative.

2.7 Why not using LibreVISA?

LibreVISA is still young. However, you can already use it with the NI backend as it has the same API. We think that PyVISA-py is easier to hack and we can quickly reach feature parity with NI-VISA for message-based instruments.

2.8 Why putting PyVISA in the middle?

Because it allows you to change the backend easily without changing your application. In other projects we implemented classes to call USBTMC devices without PyVISA. But this leads to code duplication or an adapter class in your code. By using PyVISA as a frontend to many backends, we abstract these things from higher level applications.