PyVISA Documentation

Release 0.5.2.dev60+g500d1de

PyVISA Authors

Oct 05, 2021

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.

Currently Pyvisa-py support the following resources:

- TCPIP INSTR
- TCPIP SOCKET
- GPIB INSTR
- ASRL INSTR
- USB INSTR
- USB RAW

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

CHAPTER 1

Installation

Pyvisa-py is available on PyPI and can be easily installed using pip:

```
pip install pyvisa-py
```

Pyvisa-py runs on Python 3.6+.

If you do not install any extra library pyvisa-py will only be able to access tcpip resources. The following sections will describe what extra libraries you need to install and how to configure them to use other resources.

1.1 Ethernet resources: TCPIP INSTR/SOCKET

Pyvisa-py relies on `socket` module in the Python Standard Library to interact with the instrument which you do not need to install any extra library to access those resources.

1.2 Serial resources: ASRL INSTR

To access serial resources, you should install PySerial. Version 3.0 or newer is required. No special configuration is required.

1.3 GPIB resources: GPIB INSTR

On all platforms, using **GPIB** resources requires to install a gpib driver. On Windows, it is install as part of NI-VISA or Keysight VISA for example. On MacOSX, you should install the NI-488 library from National instrument. On Linux, you can use a commercial driver (NI) or the linux-gpib project.

On Linux, linux-gpib comes with Python bindings so you do not have to install any extra library. On all systems with GPIB device drivers, GPIB support is available through gpib-ctypes.

You should not have to perform any special configuration after the install.

1.4 USB resources: USB INSTR/RAW

For **USB** resources, you need to install PyUSB. PyUSB relies on USB driver library such as libusb 0.1, libusb 1.0, libusbx, libusb-win32 and OpenUSB that you should also install. Please refer to PyUSB documentation for more details.

On Unix system, one may have to modify udev rules to allow non-root access to the device you are trying to connect to. The following tutorial describes how to do it http://ask.xmodulo.com/change-usb-device-permission-linux.html.

On Windows, you may have to uninstall the USBTMC specific driver installed by Windows and re-install a generic driver.

Note that on Windows, devices that are already open cannot be detected and will not be returned by `ResourceManager.list_resources`.

Another useful reference for how to configure your system is h ttps://github.com/python-ivi/python-usbtmc.

1.5 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.

1.6 Using the development version

You can install the latest development version (at your own risk) directly form GitHub:

```
$ pip install -U git+https://github.com/pyvisa/pyvisa-py.git
```

CHAPTER 2

FAQ

2.1 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.

2.2 Why are you developing this?

The IVI compliant VISA implementation available (National Instruments's VISA , Keysight, Tektronik, etc) are proprietary libraries that only works on certain systems. We wanted to provide a compatible alternative.

2.3 Can PyVISA-py be used from a VM?

Because PyVISA-py access hardware resources (such as USB ports) running from a VM can cause issues, such as unexpected timeouts because the VM does not receive the response. You may be able to set the VM in such that it works but you should refer to your VM manual. (see https://github.com/pyvisa/pyvisa-py/issues/243 for the kind of issue it can cause)

2.4 Can PyVISA-py be used from a Docker container?

As the Windows variant of Docker cannot forward neighter USB ports nor GPIB interfaces the obvious choice would be to connect via TCP/IP. The problem of a Docker container is, that idle connections are disconnected by the VPN garbage collection. For this reason it is reasonable to enable keepalive packets. For this the VISA attribute *VI_ATTR_TCPIP_KEEPALIVE* has been modified to work for all TCP/IP instruments. Enabling this option can be done with:

```
inst.set_visa_attribute(pyvisa.constants.ResourceAttribute.tcpip_keepalive, True)
```

where *inst* is an active TCP/IP visa session. (see https://tech.xing.com/a-reason-for-unexplained-connection-timeouts-on-kubernetes-docker-abd041cf7e02 if you want to read more about connection dropping in docker containers)

2.5 Why not using LibreVISA?

LibreVISA is still young and appears mostly unmaintained at this point. However, you can already use it with the IVI backend as it has the same API. We think that PyVISA-py is easier to hack and we can quickly reach feature parity with other IVI-VISA implementation for message-based instruments.

2.6 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.