pynetdicom Documentation

Release 0.8.1

Patrice Munger

May 24, 2017

Contents

Contents:

CHAPTER 1

Getting Started

Introduction

pynetdicom is a pure python package implementing the DICOM network protocol. It uses the pydicom package.

pynetdicom makes it easy to create DICOM clients (SCUs) or servers (SCPs). The following service classes are currently supported, both as SCU and SCP:

- Verification
- Storage
- Query/Retrieve

pynetdicom is easy to install and use, and because it is a pure python package, it should run anywhere python runs.

You can find examples in *here*.

pynetdicom is hosted here.

License

pynetdicom uses the MIT license.

Prerequisites

- python 2.4 and higher
- pydicom 0.9.7 and above

Installation

Here are the installation options:

- pynetdicom is registered at PyPi, so it can be installed with pip:

```
pip install pynetdicom
```

- download the source package , uncompress and install with:

```
python setup.py install
```

- A windows installer is also available.
- To get the latest revision you can clone the source tree with:

```
hg clone https://github.com/patmun/pynetdicom
```

Support

Please join the pynetdicom discussion group to ask questions, give feedback, post example code for others – in other words for any discussion about the pynetdicom code. New versions, major bug fixes, etc. will also be announced through the group.

CHAPTER 2

Typical use cases

Send DICOM data from python

pynetdicom allows DICOM data (datasets) to be sent to a remote SCP directly from python.

get source (storescu.py)

```
#!/usr/bin/python
"""
Storage SCU example.

This demonstrates a simple application entity that support the RT Plan
Storage SOP Class as SCU. For this example to work, there must be an
SCP listening on the specified host and port.

For help on usage,
python storescu.py -h
"""
import sys
import argparse
from netdicom import AE
from netdicom.SOPclass import StorageSOPClass, VerificationSOPClass
from dicom.UID import ExplicitVRLittleEndian, ImplicitVRLittleEndian, \
    ExplicitVRBigEndian
from dicom import read_file


# parse commandline
parser = argparse.ArgumentParser(description='storage SCU example')
parser.add_argument('remotehost')
parser.add_argument('remoteport', type=int)
parser.add_argument('file', nargs='+')
parser.add_argument('-aet', help='calling AE title', default='PYNETDICOM')
parser.add_argument('-aec', help='called AE title', default='REMOTESCU')
parser.add_argument('-implicit', action='store_true',
                    help='negociate implicit transfer syntax only',
                    default=False)
parser.add_argument('-explicit', action='store_true',
                    help='negociate explicit transfer syntax only',
                    default=False)

args = parser.parse_args()

if args.implicit:
    ts = [ImplicitVRLittleEndian]
elif args.explicit:
    ts = [ExplicitVRLittleEndian]
else:
    ts = [
        ExplicitVRLittleEndian,
        ImplicitVRLittleEndian,
        ExplicitVRBigEndian
    ]

# call back


def OnAssociateResponse(association):
    print "Association response received"

# create application entity
MyAE = AE(args.aet, 0, [StorageSOPClass,  VerificationSOPClass], [], ts)
MyAE.OnAssociateResponse = OnAssociateResponse

# remote application entity
RemoteAE = dict(Address=args.remotehost, Port=args.remoteport, AET=args.aec)

# create association with remote AE
print "Request association"
assoc = MyAE.RequestAssociation(RemoteAE)

if not assoc:
    print "Could not establish association"
    sys.exit(1)
# perform a DICOM ECHO, just to make sure remote AE is listening
print "DICOM Echo ... ",
st = assoc.VerificationSOPClass.SCU(1)
print 'done with status "%s"' % st

# create some dataset
for ii in args.file:
    print
    print ii
    d = read_file(ii)
    print "DICOM StoreSCU ... ",
    try:
        st = assoc.SCU(d, 1)
        print 'done with status "%s"' % st
    except:
        raise
        print "problem", d.SOPClassUID
print "Release association"
assoc.Release(0)
```

```
# done
MyAE.Quit()
```

Receive DICOM data from python

Create a storage SCP which can accept DICOM data from remote SCUs. On reception of the data, a user-defined callback function is called.

`get source (storescp.py)`

```
"""
Storage SCP example.

This demonstrates a simple application entity that support a number of
Storage service classes. For this example to work, you need an SCU
sending to this host on specified port.

For help on usage,
python storescp.py -h
"""

import argparse
from netdicom import AE, logger_setup, debug
from netdicom.SOPclass import StorageSOPClass, VerificationSOPClass
from dicom.dataset import Dataset, FileDataset
import tempfile

# parse commandline
parser = argparse.ArgumentParser(description='storage SCP example')
parser.add_argument('port', type=int)
parser.add_argument('-aet', help='AE title of this server',
                    default='PYNETDICOM')
args = parser.parse_args()

#logger_setup()
#debug(True)


# callbacks
def OnAssociateRequest(association):
    print "association requested"


def OnAssociateResponse(association):
    print "Association response received"


def OnReceiveEcho(self):
    print "Echo received"


def OnReceiveStore(SOPClass, DS):
    #print "Received C-STORE"
    # do something with dataset. For instance, store it on disk.
    file_meta = Dataset()
    file_meta.MediaStorageSOPClassUID = DS.SOPClassUID
    file_meta.MediaStorageSOPInstanceUID = "1.2.3"  # !! Need valid UID here
    file_meta.ImplementationClassUID = "1.2.3.4"  # !!! Need valid UIDs here
    filename = '%s/%s.dcm' % (tempfile.gettempdir(), DS.SOPInstanceUID)
    ds = FileDataset(filename, {}, file_meta=file_meta, preamble="\0" * 128)
    ds.update(DS)
    ds.is_little_endian = True
    ds.is_implicit_VR = True
    ds.save_as(filename)
    #print "File %s written" % filename
    # must return appropriate status
    return SOPClass.Success


# setup AE
MyAE = AE(args.aet, args.port, [], [StorageSOPClass, VerificationSOPClass])
MyAE.OnAssociateRequest = OnAssociateRequest
MyAE.OnAssociateResponse = OnAssociateResponse
MyAE.OnReceiveStore = OnReceiveStore
MyAE.OnReceiveEcho = OnReceiveEcho

# start AE
print "starting AE ... ",
MyAE.start()
print "done"
MyAE.QuitOnKeyboardInterrupt()
```

Query/Retrieve from python

Here is how to query and retrieve some dataset from a Q/R SCP.

get source (qrscu.py)

```
"""
Query/Retrieve SCU AE example.

This demonstrates a simple application entity that support the Patient
Root Find and Move SOP Classes as SCU. In order to receive retrieved
datasets, this application entity must support the CT Image Storage
SOP Class as SCP as well. For this example to work, there must be an
SCP listening on the specified host and port.

For help on usage,
python qrscu.py -h
"""

import argparse
from netdicom.applicationentity import AE
from netdicom.SOPclass import *
from dicom.dataset import Dataset, FileDataset
from dicom.UID import ExplicitVRLittleEndian, ImplicitVRLittleEndian, \
    ExplicitVRBigEndian
import netdicom
# netdicom.debug(True)
import tempfile
```

```
# parse commandline
parser = argparse.ArgumentParser(description='storage SCU example')
parser.add_argument('remotehost')
parser.add_argument('remoteport', type=int)
parser.add_argument('searchstring')
parser.add_argument('-p', help='local server port', type=int, default=0)
parser.add_argument('-aet', help='calling AE title', default='PYNETDICOM')
parser.add_argument('-aec', help='called AE title', default='REMOTESCU')
parser.add_argument('-implicit', action='store_true',
                    help='negociate implicit transfer syntax only',
                    default=False)
parser.add_argument('-explicit', action='store_true',
                    help='negociate explicit transfer syntax only',
                    default=False)

args = parser.parse_args()

if args.implicit:
    ts = [ImplicitVRLittleEndian]
elif args.explicit:
    ts = [ExplicitVRLittleEndian]
else:
    ts = [
        ExplicitVRLittleEndian,
        ImplicitVRLittleEndian,
        ExplicitVRBigEndian
    ]

# call back


def OnAssociateResponse(association):
    print "Association response received"


def OnAssociateRequest(association):
    print "Association resquested"
    return True


def OnReceiveStore(SOPClass, DS):
    print "Received C-STORE", DS.PatientName
    try:
        # do something with dataset. For instance, store it.
        file_meta = Dataset()
        file_meta.MediaStorageSOPClassUID = '1.2.840.10008.5.1.4.1.1.2'
        # !! Need valid UID here
        file_meta.MediaStorageSOPInstanceUID = "1.2.3"
        # !!! Need valid UIDs here
        file_meta.ImplementationClassUID = "1.2.3.4"
        filename = '%s/%s.dcm' % (tempfile.gettempdir(), DS.SOPInstanceUID)
        ds = FileDataset(filename, {},
                         file_meta=file_meta, preamble="\0" * 128)
        ds.update(DS)
        #ds.is_little_endian = True
        #ds.is_implicit_VR = True
        ds.save_as(filename)
        print "File %s written" % filename
    except:
        pass
    # must return appropriate status
    return SOPClass.Success


# create application entity
MyAE = AE(args.aet, args.p, [PatientRootFindSOPClass,
                             PatientRootMoveSOPClass,
                             VerificationSOPClass], [StorageSOPClass], ts)
MyAE.OnAssociateResponse = OnAssociateResponse
MyAE.OnAssociateRequest = OnAssociateRequest
MyAE.OnReceiveStore = OnReceiveStore
MyAE.start()


# remote application entity
RemoteAE = dict(Address=args.remotehost, Port=args.remoteport, AET=args.aec)

# create association with remote AE
print "Request association"
assoc = MyAE.RequestAssociation(RemoteAE)


# perform a DICOM ECHO
print "DICOM Echo ... ",
st = assoc.VerificationSOPClass.SCU(1)
print 'done with status "%s"' % st

print "DICOM FindSCU ... ",
d = Dataset()
d.PatientsName = args.searchstring
d.QueryRetrieveLevel = "PATIENT"
d.PatientID = "*"
st = assoc.PatientRootFindSOPClass.SCU(d, 1)
print 'done with status "%s"' % st

for ss in st:
    if not ss[1]:
        continue
    # print ss[1]
    try:
        d.PatientID = ss[1].PatientID
    except:
        continue
    print "Moving"
    print d
    assoc2 = MyAE.RequestAssociation(RemoteAE)
    gen = assoc2.PatientRootMoveSOPClass.SCU(d, 'PYNETDICOM', 1)
    for gg in gen:
        print
        print gg
    assoc2.Release(0)
    print "QWEQWE"

print "Release association"
assoc.Release(0)
```

```
# done
MyAE.Quit()
```