pynetdicom3 Documentation

Release 1.0

scaramallion

Apr 28, 2018

Contents:

1 ApplicationEntity **1**
- 1.1 Initialisation . . . 1
- 1.2 Service Class User . . . 1
- 1.3 Service Class Provider . . . 1
- 1.4 Association . . . 2

2 Introduction to Association **3**

3 Examples **5**
- 3.1 Service Class User Examples . . . 5
- 3.2 SCP examples . . . 7

4 Applications **9**
- 4.1 echoscu . . . 9
- 4.2 echoscp . . . 11

5 Package documentation **13**
- 5.1 ApplicationEntity . . . 13
- 5.2 Association . . . 13
- 5.3 SOP Class . . . 13
- 5.4 ACSE . . . 13
- 5.5 DIMSE . . . 13
- 5.6 DUL . . . 13

6 Indices and tables **15**

CHAPTER 1

ApplicationEntity

ApplicationEntity (or AE) is the main class for constructing a DICOM Application Entity.

1.1 Initialisation

```
from pydicom.uid import ImplicitVRLittleEndian
from pynetdicom3 import AE, VerificationSOPClass

ae = AE(ae_title='PYNETDICOM',
        port=11112,
        scu_sop_class=[VerificationSOPClass],
        scp_sop_class=[VerificationSOPClass],
        transfer_syntax=[ImplicitVRLittleEndian])
```

1.2 Service Class User

1.2.1 Maximum PDU Size

Each association request can specify its own maximum PDU receive size. A value of None indicates that there is no maximum size limit. >>> ae = AE(scu_sop_class=[VerificationSOPClass] >>> assoc1 = ae.associate('127.0.0.1', 11112, max_pdu=16382) >>> assoc2 = ae.associate('127.0.0.1', 11112, max_pdu=None)

1.3 Service Class Provider

The following parameters should be set prior to calling *start()*.

1.3.1 AE Title Matching

The called and/or calling AE title can be set to be required to match against an expected value, with the association being rejected if they fail. A value of ‘’ means that no matching will be performed. >>> ae = AE(port=11112, scp_sop_class=[VerificationSOPClass]) >>> ae.require_calling_aet = ‘CALLING_AET’ >>> ae.require_called_aet = ‘CALLED_AET’ >>> ae.start()

1.3.2 Maximum Number of Associations

The maximum number of simultaneous associations that the AE will support. Any additional association requests be rejected with a reason of ‘Local limit exceeded’. >>> ae = AE(port=11112, scp_sop_class=[VerificationSOPClass]) >>> ae.maximum_associations = 3 >>> ae.start()

1.3.3 Timeouts

Timeouts for ACSE, DIMSE and network messages >>> ae = AE(scu_sop_class=[VerificationSOPClass]) >>> ae.dimse_timeout = 30 >>> ae.acse_timeout = 60 >>> ae.network_timeout = 60 >>> ae.start()

1.3.4 Maximum PDU Size

A value of None indicates that there is no maximum PDU size limit. >>> ae = AE(port=11112, scp_sop_class=[VerificationSOPClass] >>> ae.maximum_pdu_size = 16382 >>> ae.start()

1.4 Association

CHAPTER 2

Introduction to Association

CHAPTER 3

Examples

3.1 Service Class User Examples

3.1.1 Verification SCU

The most basic way of verifying and troubleshooting a connection with a peer Application Entity (AE) is to send a DICOM C-ECHO, which utilises the *Verification SOP Class*

```
from pynetdicom3 import AE

# The Verification SOP Class has a UID of 1.2.840.10008.1.1
ae = AE(scu_sop_class=['1.2.840.10008.1.1'])

# Try and associate with the peer AE
#   Returns the Association thread
print('Requesting Association with the peer')
assoc = ae.associate(addr, port)

if assoc.is_established:
    print('Association accepted by the peer')
    # Send a DIMSE C-ECHO request to the peer
    # status is a pydicom Dataset object with (at a minimum) a
    # (0000, 0900) Status element
    status = assoc.send_c_echo()

    # Output the response from the peer
    if status:
        print('C-ECHO Response: 0x{0:04x}'.format(status.Status))

    # Release the association
    assoc.release()
elif assoc.is_rejected:
    print('Association was rejected by the peer')
```

```
elif assoc.is_aborted:
    print('Received an A-ABORT from the peer during Association')
```

3.1.2 Storage SCU

A common use of a DICOM SCU is to use a DICOM C-STORE message to send DICOM datasets to peer AEs.

```
from pydicom import read_file
from pynetdicom3 import AE
from pynetdicom3 import StorageSOPClassList

# StorageSOPClassList contains all the Standard SOP Classes supported
#   by the Storage Service Class (see PS3.4 Annex B.5)
ae = AE(scu_sop_class=StorageSOPClassList)

# Try and associate with the peer AE
#   Returns the Association thread
print('Requesting Association with the peer')
assoc = ae.associate(addr, port)

if assoc.is_established:
    print('Association accepted by the peer')

    # Read the DICOM dataset from file 'dcmfile'
    dataset = read_file('dcmfile')

    # Send a DIMSE C-STORE request to the peer
    status = assoc.send_c_echo(dataset)
    print('C-STORE status: %s' %status)

    # Release the association
    assoc.release()
```

3.1.3 Query/Retrieve - Find SCU

Query the peer AE to see if it contains any Instances with attributes matching those specified by the user-created *dataset*.

```
from pydicom.dataset import Dataset

from pynetdicom3 import AE
from pynetdicom3 import QueryRetrieveSOPClassList

# QueryRetrieveSOPClassList contains the SOP Classes supported
#   by the Query/Retrieve Service Class (see PS3.4 Annex C.6)
ae = AE(scu_sop_class=QueryRetrieveSOPClassList)

# Try and associate with the peer AE
#   Returns the Association thread
print('Requesting Association with the peer')
assoc = ae.associate(addr, port)

if assoc.is_established:
    print('Association accepted by the peer')
```

```
# Creat a new DICOM dataset with the attributes to match against
#   In this case match any patient's name at the PATIENT query
#   level. See PS3.4 Annex C.6 for the complete list of possible
#   attributes and query levels.
dataset = Dataset()
dataset.PatientName = '*'
dataset.QueryRetrieveLevel = "PATIENT"

# Send a DIMSE C-FIND request to the peer
#   query_model is the Query/Retrieve Information Model to use
#   and is one of 'W', 'P', 'S', 'O'
#       'W' - Modality Worklist (1.2.840.10008.5.1.4.31)
#       'P' - Patient Root (1.2.840.10008.5.1.4.1.2.1.1)
#       'S' - Study Root (1.2.840.10008.5.1.4.1.2.2.1)
#       'O' - Patient/Study Only (1.2.840.10008.5.1.4.1.2.3.1)
responses = assoc.send_c_find(dataset, query_model='P')

for (status, dataset) in responses:
    # While status is pending we should get the matching datasets
    if status == 'Pending':
        print(dataset)
    elif status == 'Success':
        print('C-FIND finished, releasing the association')
    elif status == 'Cancel':
        print('C-FIND cancelled, releasing the association')
    elif status == 'Failure':
        print('C-FIND failed, releasing the association')

# Release the association
assoc.release()
```

3.2 SCP examples

CHAPTER 4

Applications

4.1 echoscu

```
echoscu [options] peer port
```

4.1.1 Description

The `echoscu` application implements a Service Class User (SCU) for the *Verification SOP Class* (UID 1.2.840.10008.1.1)[1]. It establishes an Association with a peer Application Entity (AE) which it then sends a DICOM C-ECHO-RQ message[2] and waits for a response. The application can be used to verify basic DICOM connectivity.

The following simple example shows what happens when it is succesfully run on an SCP that supports the *Verification SOP Class*:

```
user@host: echoscu 192.168.2.1 11112
user@host:
```

When attempting to send a C-ECHO to an SCP that doesn't support the *Verification SOP Class*:

```
user@host: echoscu 192.168.2.1 11112
E: No Acceptable Presentation Contexts
user@host:
```

When attempting to associate with a non-DICOM peer

```
user@host: echoscu 192.168.2.1 11112
E: Association Request Failed: Failed to establish association
E: Peer aborted Association (or never connected)
E: TCP Initialisation Error: (Connection refused)
user@host:
```

[1] See DICOM Standard 2015b PS3.6 Table A-1
[2] See DICOM Standard 2015b PS3.7 Sections 9.1.5 and 9.3.5

Using the `--propose-pc [n]` option, the echoscu application can also propose *n* Presentation Contexts[3] (all with an Abstract Syntax of *Verification SOP Class*) in order to provide debugging assistance for association negotiation

4.1.2 Options

Logging

-q --quiet quiet mode, prints no warnings or errors

-v --verbose verbose mode, prints processing details

-d --debug debug mode, prints debugging information

-ll --log-level [l]evel (str) One of ['critical', 'error', 'warning', 'info', 'debug'], prints logging messages with corresponding level or higher

-lc --log-config [f]ilename (str) use python logging config[4] file f for the logger

Application Entity Titles

-aet --calling-aet [a]etitle (str) set the local AE title (default: ECHOSCU)

-aec --called-aet [a]etitle (str) set the called AE title for the peer AE (default: ANY-SCP)

Association Negotiation Debugging

-pts --propose-ts [n]umber (int) propose n transfer syntaxes (1-3)

Miscellaneous DICOM

-to --timeout [s]econds (int) timeout for connection requests (default: unlimited)

-ta --acse-timeout [s]econds (int) timeout for ACSE messages (default: 30)

-td --dimse-timeout [s]econdsr (int) timeout for DIMSE messages (default: unlimited)

-pdu --max-pdu [n]umber of bytes (int) set maximum receive PDU bytes to n bytes (default: 16384)

--repeat [n]umber (int) repeat echo n times

--abort abort association instead of releasing it

4.1.3 DICOM Conformance

The echoscu application supports the following SOP Class as an SCU:

Verification SOP Class	1.2.840.10008.1.1

Unless the `--propose-ts` option is used, the echoscu application will only propose the *Little Endian Implicit VR Transfer Syntax* (UID 1.2.840.10008.1.2). The supported Transfer Syntaxes[5] are:

[3] See DICOM Standard 2015b PS3.8 Sections 7.1.1.13 and 9.3.2.2
[4] See the Python documentation
[5] See DICOM Standard 2015b PS3.5 Section 10 and Annex A

```
Little Endian Implicit VR       1.2.840.10008.1.2
Little Endian Explicit VR       1.2.840.10008.1.2.1
Big Endian Explicit VR          1.2.840.10008.1.2.2
```

4.2 echoscp

```
echoscp [options] port
```

4.2.1 Description

The `echoscp` application implements a Service Class Provider (SCP) for the *Verification SOP Class* (UID 1.2.840.10008.1.1)[1]. It establishes an Association with peer Application Entities (AEs) and receives DICOM C-ECHO-RQ[2] message to which it responds with a DICOM C-ECHO-RSP message. The application can be used to verify basic DICOM connectivity.

The following example shows what happens when it is started and receives a C-ECHO from a peer:

```
user@host: echoscp 11112
```

More information is available when a connection is received while running with the `-v` option:

```
user@host: echoscp 11112 -v
I: Association Received
I: Association Acknowledged
I: Received Echo Request (MsgID 1)
I: Association Released
```

When a peer AE attempts to send non C-ECHO message:

```
user@host: echoscu 192.168.2.1 11112 -v
I: Association Received
I: Association Acknowledged
I: Association Aborted
```

Much more information is available when a connection is received while running with the `-d` option:

```
user@host: echoscp 11112 -d
D: $echosco.py v0.2.0 2016-03-15 $
D:
D: Starting DICOM UL service "Thread-1"
D: PDU Type: Associate Request, PDU Length: 215 + 6 bytes PDU header
D:     01 00 00 00 00 d1 00 01 00 00 41 4e 59 2d 53 43
...
D: Request Parameters:
D: ======================= BEGIN A-ASSOCIATE-RQ ============================
D: Their Implementation Class UID: 1.2.826.0.1.3680043.9.381.0.9.0
...
I: Received Echo Request (MsgID 1)
...
I: Association Released
D: DICOM UL service "Thread-1" stopped
```

[1] See DICOM Standard 2015b PS3.6 Table A-1
[2] See DICOM Standard 2015b PS3.7 Sections 9.1.5 and 9.3.5

4.2.2 Options

Logging

-q --quiet quiet mode, prints no warnings or errors

-v --verbose verbose mode, prints processing details

-d --debug debug mode, prints debugging information

-ll --log-level [l]evel (str) One of ['critical', 'error', 'warning', 'info', 'debug'], prints logging messages with corresponding level l or higher

-lc --log-config [f]ilename (str) use python logging config[3] file f for the logger

Application Entity Titles

-aet --aetitle [a]etitle (str) set my AE title (default: ECHOSCP)

Miscellaneous DICOM

-to --timeout [s]econds (int) timeout for connection requests (default: unlimited)

-ta --acse-timeout [s]econds (int) timeout for ACSE messages (default: 30)

-td --dimse-timeout [s]econdsr (int) timeout for DIMSE messages (default: unlimited)

-pdu --max-pdu [n]umber of bytes (int) set maximum receive PDU bytes to n bytes (default: 16384)

Preferred Transfer Syntaxes

-x= --prefer-uncompr prefer explicit VR local byte order (default)

-xe --prefer-little prefer explicit VR little endian transfer syntax

-xb --prefer-big prefer explicit VR big endian transfer syntax

-xi --implicit accept implicit VR little endian transfer syntax only

4.2.3 DICOM Conformance

The `echoscp` application supports the following SOP Class as an SCP:

Verification SOP Class	1.2.840.10008.1.1

The supported Transfer Syntaxes[4] are:

Little Endian Implicit VR	1.2.840.10008.1.2
Little Endian Explicit VR	1.2.840.10008.1.2.1
Big Endian Explicit VR	1.2.840.10008.1.2.2

[3] See DICOM Standard 2015b PS3.8 Sections 7.1.1.13 and 9.3.2.2
[4] See the Python documentation

CHAPTER 5

Package documentation

5.1 ApplicationEntity

5.2 Association

5.3 SOP Class

5.4 ACSE

5.5 DIMSE

5.5.1 DIMSEServiceProvider

5.5.2 DIMSE primitives

5.5.3 DIMSEMessage

5.6 DUL

5.6.1 DULServiceProvider

5.6.2 PDU

5.6.3 PDU primitives

CHAPTER 6

Indices and tables

- genindex
- modindex
- search