txHL7 Documentation

Release 0.1.0

John Paulett

September 08, 2020

Contents

txHL7 provides a network-level HL7 implementation, the Minimal Lower Level Protocol (MLLP) using Python's twisted.

CHAPTER 1

Current Status

txHL7 is still *alpha* quality software. The API can still drastically change before a "1.0" release. Currently, only the bare-minimum of MLLP is implemented. However, we are working towards more complete support of the specification.

CHAPTER 2

python-hl7 vs txHL7

python-hl7 and txHL7 are complementary python modules for handling HL7 data.

- txHL7 provides a network-level HL7 implementation (MLLP)
- python-hl7 provides a HL7 parsing implementation

txHL7 and python-hl7 can (and often are) used together. But, the modular approach allows a developer to substitute out either component. For example, a developer may not wish to use Twisted, instead he may elect to implement the TCP server using `socket` or `asyncore`. Likewise, a developer may wish to use an alternate HL7 parsing routine, but still use txHL7.

As of the *0.2.0 release*, there is a streamlined way to use python-hl7 as the parser for txHL7, via the `txHL7.receiver.AbstractHL7Receiver`.

CHAPTER 3

txHL7 vs twisted-hl7

twisted-hl7 is the previous project name for txHL7. The "tx" prefix better follows twisted's Community Code recommendations.

txHL7 is not an official twisted project.

Note: Please update setup dependencies and imports to txHL7. See *0.3.0 - November 2014*

CHAPTER 4

Contents

4.1 Usage

Run simple demo server on default port 2575:

```
twistd --nodaemon mllp
```

Run simple server with a *custom receiver* on port 7575:

```
twistd --nodaemon mllp --endpoint tcp:7575 --receiver myreceiver.Receiver
```

Options help:

```
twistd mllp --help
```

Note: Installation of this package may result in a warning which can be ignored:

```
package init file 'twisted/plugins/__init__.py' not found (or not a regular file)
```

4.1.1 Direct Factory & Reactor Usage

twistd & the mllp plugin are not required. You are able to directly start a twisted reactor or application, instantiating the, `txHL7.mllp.MLLPFactory` passing into it an instance of an `txHL7.receiver.IHL7Receiver`:

```
from twisted.internet import reactor
from txHL7.mllp import MLLPFactory
from myreceiver import Receiver

def run(port):
    receiver = Receiver()
    factory = MLLPFactory(receiver)

    reactor = get_reactor()
    reactor.listenTCP(port, factory)
    reactor.run()

run(6666)
```

4.2 Custom Receivers

txHL7 ships with a simple example of a `twisted.receiver.LoggingReceiver`, but for most cases, you will want to execute some custom actions when a message is received. To do this, you will need to define your own receiver, which implements `txHL7.receiver.IHL7Receiver`.

`txHL7.receiver.IHL7Receiver` only requires a few methods to be implemented (and if you look further down this document, you will find the even easier `AbstractHL7Receiver`):

- `parseMessage` provides the parsing logic to transform the pipe-delimited message into something more useful. It will return an instance of `txHL7.receiver.MessageContainer`. The `MessageContainer` is important, because it implements how build an HL7 ACK message.
- `handleMessage` receives the parsed `txHL7.receiver.MessageContainer` and is where you should put your business logic. It must return a `twisted.internet.defer.Deferred` instance.
- Internally txHL7 treats data as unicode, and `getCodec` provides the codec to use to decode the bytestring into unicode.

If you wish to use python-hl7 to parse the message, `txHL7.receiver.AbstractHL7Receiver` makes your job even easier. You just need to implement `handleMessage` and optionally `getCodec`.

4.2.1 Example Receiver

A simple example:

```
from txHL7.receiver import AbstractHL7Receiver
from twisted.internet import defer

class ExampleReceiver(AbstractHL7Receiver):
    def handleMessage(self, container):
        message = container.message

        # Our business logic
        mrn = message.segment('PID')[3][0]
        # Do something with mrn

        # We succeeded, so ACK back (default is AA)
        return defer.succeed(container.ack())

    def getCodec(self):
        # Our messages are encoded in Windows-1252
        # WARNING this is an example and is not universally true! You will
        # need to figure out the encoding you are receiving.
        return 'cp1252'
```

We can launch this receiver with:

```
twistd --nodaemon mllp --receiver example.ExampleReceiver
```

4.2.2 Deferring to a Thread

Remember that twisted is non-blocking. Blocking operations should ideally be executed outside the main loop. One way to accomplish this is to defer to a thread in twisted [1].

[1] https://twistedmatrix.com/documents/current/core/howto/threading.html

Warning: It is up to you to ensure that your application is thread-safe.

Here is an example that calls a blocking operation in `importMessage`, which is defered to a thread in `handleMessage`. We additionally, show catching an error are returning a reject message:

```
from txHL7.receiver import AbstractHL7Receiver
from twisted.internet import threads

class ThreadedReceiver(AbstractHL7Receiver):
    def saveMessage(self, message):
        try:
            self.database.insert(message)
            return message.ack()
        except:
            # reject the message
            return message.ack(ack_code='AR')

    def handleMessage(self, message):
         return threads.deferToThread(self.saveMessage, message)
```

4.3 API

4.3.1 Receivers

class txHL7.receiver.**AbstractHL7Receiver**
Bases: txHL7.receiver.AbstractReceiver

Abstract base class implementation of txHL7.receiver.IHL7Receiver

Return type txHL7.receiver.HL7MessageContainer

message_cls
alias of HL7MessageContainer

class txHL7.receiver.**AbstractReceiver**
Bases: builtins.object

Abstract base class implementation of txHL7.receiver.IHL7Receiver

message_cls
alias of MessageContainer

class txHL7.receiver.**HL7MessageContainer**(*raw_message*)
Bases: txHL7.receiver.MessageContainer

Message implementation that parses using python-hl7

ack(*ack_code='AA'*)
Return HL7 ACK created from the source message.

Return type unicode

interface txHL7.receiver.**IHL7Receiver**
Interface that must be implemented by MLLP protocol receiver instances

parseMessage(*raw_message*)
Clients should parse the message and return an instance of txHL7.receiver.MessageContainer or subclass.

Return type txHL7.receiver.MessageContainer

handleMessage(*message_container*)
Clients must implement handleMessage, which takes a message_container argument that is the txHL7.receiver.MessageContainer instance returned from txHL7.receiver.IHL7Receiver.parseMessage(). The implementation, if non-blocking, may directly return the ack/nack message or can return the ack/nack within a twisted.internet.defer.Deferred. If the implementation involves any blocking code, the implementation must return the result as twisted.internet.defer.Deferred (possibly by using twisted.internet.threads.deferToThread()), to prevent the event loop from being blocked.

getCodec()
Clients should return the codec name [1]_ and error handling scheme [2]_, used when decoding into unicode.

Return type tuple(codec, errors)

getTimeout()
Clients should return the idle timeout in seconds, or None for no timeout

Return type int

class txHL7.receiver.**LoggingReceiver**
Bases: txHL7.receiver.AbstractHL7Receiver

Simple MLLP receiver implementation that logs and ACKs messages.

class txHL7.receiver.**MessageContainer**(*raw_message*)
Bases: builtins.object

Base class for messages returned from txHL7.receiver.IHL7Receiver.parseMessage() and passed to txHL7.receiver.IHL7Receiver.handleMessage()

ack(*ack_code='AA'*)
Return unicode acknowledgement message, or None for no ACK.

ack_code options are one of *AA* (accept), *AR* (reject), *AE* (error)

Return type unicode

err(*failure*)
Handle a twisted errback twisted.python.failure.Failure failure. Subclasses can override to log errors or handle them in a different way. Default implementation returns a rejection ACK.

Return type unicode

4.3.2 MLLP

class txHL7.mllp.**MinimalLowerLayerProtocol**
Bases: twisted.internet.protocol.Protocol, twisted.protocols.policies.TimeoutMixin

Minimal Lower-Layer Protocol (MLLP) takes the form:

<VT>[HL7 Message]<FS><CR>

References:

4.3.3 MLLP Plugin

class twisted.plugins.mllp_plugin.**MLLPServiceMaker**
Bases: builtins.object

Service maker for the MLLP server.

makeService(*options*)
Construct a server using MLLPFactory.

Return type twisted.application.internet.StreamServerEndpointService

options
alias of Options

class twisted.plugins.mllp_plugin.**Options**
Bases: twisted.python.usage.Options

Define the options accepted by the twistd mllp plugin

4.4 Change Log

4.4.1 0.5.0 - September 2020

- Upgrade to python-hl7 v0.4.0 * Trailing carriage return in Message now passed along per Message Construction Rules. Requires python-hl7 v0.4.0.
  - MSH.9.1.3 ("ACK") added
- Dropped support for Python 2.7 & 3.4. Python 3.5 - 3.8 now supported.

4.4.2 0.4.1 - July 2016

- Added IHL7Receiver.getTimeout() interface method. Receivers can return an integer TCP connection idle timeout in seconds.

4.4.3 0.4.0 - June 2016

- Ported to Python 3.4. Support for Python 2.6 dropped. Currently supported platforms are Python 2.7 & 3.4. API remains the same, mostly test changes to more explicitly indicate bytestrings vs unicode strings. Also needed to convert to use zope.interface's @implementer() class advice instead of the implements().
- Use tox as primary test runner.
- Distribute Python wheel.

4.4.4 0.3.0 - November 2014

- Renamed project from twisted-hl7 to txHL7, to be in line with twisted's Community Code recommendations.

Warning: Please update your project to use the `txHL7` import instead of `twistedhl7` and replace "twisted-hl7" with "txHL7" in your setup.py or requirements.txt.
If you perform a `pip uninstall twisted-hl7`, ensure you do it before installing txHL7, since both packages use the `twisted/plugins/mllp_plugin.py` twisted plugin, otherwise the twisted-hl7 uninstall will remove txHL7's version of the plugin.

4.4.5 0.2.2 - November 2014

- Add a description to setup.py. Thanks *Low Kian Seong <https://github.com/lowks>'_*

4.4.6 0.2.1 - November 2014

- Delegate error processing to `txHL7.receiver.MessageContainer.err()`, allowing subclasses to define logic.

4.4.7 0.2.0 - September 2014

- Abstract message into a separate class that is responsible for building ACK. This makes txHL7 useable with other HL7 parsing frameworks. Requires a new `IHL7Receiver.parseMessage()` interface method.
- Add message implementation based on python-hl7, using the new ACK functionality in 0.3.1
- Upgrade to latest python-hl7, v0.3.1, for ACK creation

4.4.8 0.1.0 - August 2014

- twistd plugin. Thanks to Andrew Wason
- Upgrade to latest python-hl7, v0.3.0, for correct MSH indexing

Warning: python-hl7 v0.3.0 breaks backwards compatibility.

4.4.9 0.0.3 - March 2012

- Convert to unicode. As soon as a message string is assembled, decode into unicode, using the codec specified by the implementation of `IHL7Receiver.getCodec()`. When writing an ACK, the message is re-encoded into that codec.

4.4.10 0.0.2 - September 2011

- ACK^001 for acknowledging messages (adds python-hl7 dependency)
- Add errBack for catching unhandled errors that responds with reject (AR) ACK.

4.4.11 0.0.1 - September 2011

- Initial basic MLLP implementation

4.5 License

4.6 Authors

- John Paulett (john -at- paulett.org)
- Andrew Wason

CHAPTER 5

References

- http://archive.hl7.org/v3ballotarchive/v3ballot7/html/foundationdocuments/transport/transport-mllp.htm

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

t