Python TSIP Documentation

Release 0.3.2

Markus Juenemann

Oct 28, 2017

Contents

Contents:

CHAPTER 1

About Python-TSIP

Python-TSIP is a Python package for parsing and creating TSIP packets. The Trimble Standard Interface Protocol (TSIP) is the binary protocol spoken by the GPS receivers sold by Trimble Navigation Ltd. (http://www.trimble.com).

Python-TSIP is available under the "BSD 2-Clause Simplified License".

CHAPTER 2

Status

Almost the full set of TSIP command and report packets understood by the Copernicus II receiver has been implemented but so far only some of them have been tested against the actual GPS. Implementing a complete set of tests against an actual Copernicus II receiver is currently work in progress. Presumably Trimble Thunderbolt and Thunderbolt-E are also supported as they appear to implement a subset of the commands/reports of the (newer) Copernicus II receiver. I don't have access to any other Trimble products.

Python-TSIP is automatically tested against the following Python versions.

- Python 2.6
- Python 2.7
- Python 3.3
- Python 3.4
- Python 3.5
- pypy
- pypy3

The tests currently fail on the following Python versions.

- Python 3.2 (syntax error in the coverage module, it may work otherwise)
- Jython (can't get Tox to work with jython)

The master branch equals the latest release. The develop branch represents the latest development but may not always pass all tests.

CHAPTER 3

Example

The following code shows how to receive the current GPS time from the receiver.

- Command packet 0x21 requests the current GPS time.
- Report packet 0x41 contains the current GPS time. Its fields are accessible by index.

```
import tsip
import serial

# Open serial connection to Copernicus II receiver
serial_conn = serial.Serial('/dev/ttyS0', 38400)
gps_conn = tsip.GPS(serial_conn)

# Prepare and send command packet 0x21
command = tsip.Packet(0x21)
gps_conn.write(command)

while True:      # should implement timeout here!!!
    report = gps_conn.read()
    if report[0] == 0x41:
        print 'GPS time of week .......: %f' % (report[1])
        print 'Extended GPS week number: %d' % (report[2])
        print 'GPS UTC offset .........: %f' % (report[3])
        break
```

More examples can be found in the *docs/examples/* folder.

CHAPTER 4

Reference

python-TSIP provides high-level and low-level APIs for communicating with Trimble GPS receivers. Both API are available by importing the *tsip* module.

The high-level API provides two classes, *tsip.Packet* for encoding and decoding TSIP packets, and *tsip.GPS*, for sending these packets to a GPS and receving packets from the GPS.

The low-level API provides one class, *tsip.gps* (lower-case, in violation of PEP-8!) for communicating with a GPS and functions for encoding and decoding TSIP packets as binary strings.

In most cases a developer will only use the two classes of the high-level API.

High-level API

Communicating with a GPS (*tsip.GPS* class)

The high-level API provides the *tsip.GPS()* class for sending *tsip.Packet()* commands to a Trimble GPS and reading *tsip.Packet()* reports from the GPS.

```
>>> import tsip
>>> import serial     # pySerial (https://pypi.python.org/pypi/pyserial)
>>> serial_conn = serial.Serial('/dev/ttyS0', 9600)
>>> gps_conn = tsip.GPS(serial_conn)
>>> command = Packet(0x21)
>>> gps_conn.write(command)
>>> while True:       # should implement timeout here!!!
...     report = gps_conn.read()
...         if report[0] == 0x41:
...             print 'GPS time of week .......: %f' % (report[1])
...             print 'Extended GPS week number: %d' % (report[2])
...             print 'GPS UTC offset .........: %f' % (report[3])
...             break
```

Instances of *tsip.GPS* can also be iterated over.

```
>>> for packet in gps_conn:
...     print packet[0]
```

TSIP Packets (*tsip.Packet* class)

Not all Trimble GPS receivers support all TSIP packets. Check the official documentation for more details and additional information.

Warning: This section is not up-to-date.

Command Packets

0x1C - Firmware Version 01

Field	Description	Notes
0	0x1c	
1	0x01	

0x1C - Firmware Version 03

Field	Description	Notes
0	0x1c	
1	0x03	

```
>>> command = Packet(0x1c, 0x03)
>>> command[0]  # 0x1c
28
>>> command[1]  # 0x03
3
>>> gps_conn.write(command)
>>> while True:
...     report = gps_conn.read()
...     if report[0] == 0x1c and report.subcode == 0x83:
...         print report
...         break
Packet(0x1c, 0x83, ...)
```

0x1E - Clear Battery Backup, then Reset

Field	Description	Notes
0	0x1e	
1	Reset type	

```
>>> command = Packet(0x1e, 0x46)     # 0x46 = factory reset
>>> command[0]  # 0x1e
30
>>> command[1]  # 0x46
70
>>> gps_conn.write(command)
```

0x1F - Request Software Versions

Field	Description	Notes
0	0x1f	

```
>>> command = Packet(0x1f)
>>> command[0]  # 0x1f
31
>>> gps_conn.write(command)
>>> while True:
...     report = gps_conn.read()
...     if report[0] == 0x45:
...         print report
...         break
Packet(0x45, ...)
```

0x21 - Request Current Time

Field	Description	Notes
0	0x21	

```
>>> command = Packet(0x21)
>>> command[0]  # 0x21
33
>>> gps_conn.write(command)
>>> while True:
...     report = gps_conn.read()
...     if report[0] == 0x41:
...         print report
...         break
Packet(0x41, ...)
```

0x23 - Initial Position (XYZ ECEF)

Field	Description	Notes
0	0x23	
1	X	
2	Y	
3	Z	

```
>>> packet = Packet(0x23, -4130.889, 2896.451, -3889.139)
>>> packet[0]        # 0x23
35
>>> packet[1]        # X
-4130.889
>>> packet[2]        # Y
2896.451
>>> packet[3]        # Z
-3889.139
>>> gps_conn.write(command)
```

0x24 - Request GPS Receiver Position Fix Mode

Field	Description	Notes
0	0x24	

```
>>> command = Packet(0x24)
>>> command[0]        # 0x24
36
>>> gps_conn.write(command)
>>> while True:
...     report = gps_conn.read()
...     if report[0] == 0x6d:
...         print report
...         break
Packet(0x6d
```

0x25 - Initiate Soft Reset & Self Test

Field	Description	Notes
0	0x25	

```
>>> command = Packet(0x25)
>>> command[0]        # 0x25
37
>>> gps_conn.write(command)
```

0x26 - Request Health

Field	Description	Notes
0	0x26	

```
>>> command = Packet(0x26)
>>> command[0]        # 0x26
38
>>> gps_conn.write(command)
>>> while True:
...     report = gps_conn.read()
...     if report[0] == 0x46 or report[0] == 0x4b:
...         print report
...         break
Packet(0x4b
```

0x27 - Request Signal Levels

Field	Description	Notes
0	0x27	

```
>>> command = Packet(0x27)
>>> command[0]      # 0x27
39
>>> gps_conn.write(command)
>>> while True:
```

```
...     report = gps_conn.read()
...     if report[0] == 0x47:
...         print report
...         break
Packet(0x47
```

0x2B - Initial Position (Latitude, Longitude, Altitude)

Field	Description	Notes
0	0x2b	
1	Latitude	
2	Longitude	
3	Alitude	

```
>>> import maths
>>> packet = Packet(0x2b, math.radians(-37.813611), math.radians(144.963056), 30.0)
>>> packet[0]     # 0x2b
43
>>> packet[1]     # radians
-0.6599720140183456
>>> packet[2]     # radians
2.5300826209529208
>>> packet[0]     # metres
30.0
>>> gps_conn.write(command)
```

0x2D - Request Oscillator Offset

Field	Description	Notes
0	0x2d	

```
>>> packet = Packet(0x2d)
>>> packet[0]     # 0x2d
45
>>> gps_conn.write(command)
>>> while True:
...     report = gps_conn.read()
...     if report[0] == 0x4d:
...         print report
...         break
Packet(0x4d
```

0x2E - Set GPS Time

Field	Description	Notes
0	0x2e	
1	GPS time of week	
2	Extended GPS week number	

```
>>> packet = Packet(0x2e,
>>> packet[0]     # 0x2e
```

```
46
>>> gps_conn.write(command)
```

0x31 - Accurate Initial Position (XYZ ECEF)

Field	Description	Notes
0	0x31	
1	Latitude	
2	Longitude	
3	Alitude	

```
>>> packet = Packet(0x2b, math.radians(-37.813611), math.radians(144.963056), 30.0)
>>> packet[0]      # 0x31
49
>>> packet[1]      # radians
-0.6599720140183456
>>> packet[2]      # radians
2.5300826209529208
>>> packet[0]      # metres
30.0
>>> gps_conn.write(command)
```

0x32 - Accurate Initial Position, (Latitude, Longitude, Altitude)

Field	Description	Notes
0	0x32	
1	None	
2	DESC	
3	DESC	
4	DESC	

```
>>> packet = Packet(0x32, 1.0, 1.0, 1.0)
>>> packet[0]      # 0x32
50
```

0x35 - Set Request I/O Options

Field	Description	Notes
0	0x35	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	

```
>>> packet = Packet(0x35, 100, 100, 100, 100)
>>> packet[0]      # 0x35
53
```

0x37 - Request Status and Values of Last Position and Velocity

Field	Description	Notes
0	0x37	
1	None	

```
>>> packet = Packet(0x37)
>>> packet[0]      # 0x37
55
```

0x38 - Request/Load Satellite System Data

Field	Description	Notes
0	0x38	
1	None	
2	DESC	
3	DESC	
4	DESC	

```
>>> packet = Packet(0x38, 100, 100, 100)
>>> packet[0]      # 0x38
56
```

0x3A - Request Last Raw Measurement

Field	Description	Notes
0	0x3a	
1	None	
2	DESC	

```
>>> packet = Packet(0x3a, 100)
>>> packet[0]      # 0x3a
58
```

0x3C - Request Current Satellite Tracking Status

Field	Description	Notes
0	0x3c	
1	None	
2	DESC	

```
>>> packet = Packet(0x3c, 100)
>>> packet[0]      # 0x3c
60
```

0x69 - Receiver Acquisition Sensitivity Mode

Field	Description	Notes
0	0x69	
1	None	

```
>>> packet = Packet(0x69)
>>> packet[0]      # 0x69
105
```

0x7E - TAIP Message Output

Field	Description	Notes
0	0x7e	
1	None	

```
>>> packet = Packet(0x7e)
>>> packet[0]      # 0x7e
126
```

0x8E-17 - Request Last Position or Auto-Report Position in UTM Single Precision Format

Field	Description	Notes
0	0x8e	
1	0x17	

```
>>> packet = Packet(0x8e, 0x17)
>>> packet[0]      # 0x8e
142
>>> packet[1]  # 0x17
23
```

0x8E-20 - Request Last Fix with Extra Information

Field	Description	Notes
0	0x8e	
1	0x20	

```
>>> packet = Packet(0x8e, 0x20)
>>> packet[0]      # 0x8e
142
>>> packet[1]  # 0x20
32
```

0x8E-21 - Request Accuracy Information

Field	Description	Notes
0	0x8e	
1	0x21	

```
>>> packet = Packet(0x8e, 0x21)
>>> packet[0]      # 0x8e
142
>>> packet[1]  # 0x21
33
```

0x8E-23 - Request Last Compact Fix Information

Field	Description	Notes
0	0x8e	
1	0x23	
2	DESC	

```
>>> packet = Packet(0x8e, 0x23, 100)
>>> packet[0]      # 0x8e
142
>>> packet[1]  # 0x23
35
```

0x8E-26 - Non-Volatile Memory Storage

Field	Description	Notes
0	0x8e	
1	0x26	

```
>>> packet = Packet(0x8e, 0x26)
>>> packet[0]      # 0x8e
142
>>> packet[1]  # 0x26
38
```

0x8E-2A - Request Fix and Channel Tracking Info, Type 1

Field	Description	Notes
0	0x8e	
1	0x2a	

```
>>> packet = Packet(0x8e, 0x2a)
>>> packet[0]      # 0x8e
142
>>> packet[1]  # 0x2a
42
```

0x8E-2B - Request Fix and Channel Tracking Info, Type 2

Field	Description	Notes
0	0x8e	
1	0x2b	

```
>>> packet = Packet(0x8e, 0x2b)
>>> packet[0]      # 0x8e
142
>>> packet[1]   # 0x2b
43
```

0x8E-4F - Set PPS Width

Field	Description	Notes
0	0x8e	
1	0x4f	

```
>>> packet = Packet(0x8e, 0x4f)
>>> packet[0]      # 0x8e
142
>>> packet[1]   # 0x4f
79
```

0xBB - Navigation Configuration

Field	Description	Notes
0	0xbb	
1	None	

```
>>> packet = Packet(0xbb)
>>> packet[0]      # 0xbb
187
```

0xBC - Protocol Configuration

Field	Description	Notes
0	0xbc	
1	None	
2	DESC	

```
>>> packet = Packet(0xbc, 100)
>>> packet[0]      # 0xbc
188
```

0xC0 - Graceful Shutdown and Go To Standby Mode

Field	Description	Notes
0	0xc0	
1	None	

```
>>> packet = Packet(0xc0)
>>> packet[0]      # 0xc0
192
```

0xC2 - SBAS SV Mask.

Field	Description	Notes
0	0xc2	
1	None	

```
>>> packet = Packet(0xc2)
>>> packet[0]      # 0xc2
194
```

Report Packets

0x41 - GPS Time

Field	Description	Notes
0	0x41	
1	None	
2	DESC	
3	DESC	
4	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x41:
...     packet[1]    #
1.0
...     packet[2]    #
100
...     packet[3]    #
1.0
```

0x42 - Single-Precision Position Fix, XYZ ECEF

Field	Description	Notes
0	0x42	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x42:
...     packet[1]    #
1.0
...     packet[2]    #
1.0
...     packet[3]    #
1.0
```

```
...     packet[4]    #
1.0
```

0x43 - Velocity Fix, XYZ ECEF

Field	Description	Notes
0	0x43	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x43:
...     packet[1]    #
1.0
...     packet[2]    #
1.0
...     packet[3]    #
1.0
...     packet[4]    #
1.0
...     packet[5]    #
1.0
```

0x45 - Software Version Information

Field	Description	Notes
0	0x45	
1	DESC	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	
7	DESC	
8	DESC	
9	DESC	
10	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x45:
...     packet[1]    #
100
...     packet[2]    #
100
```

```
...     packet[3]    #
100
...     packet[4]    #
100
...     packet[5]    #
100
...     packet[6]    #
100
...     packet[7]    #
100
...     packet[8]    #
100
...     packet[9]    #
100
...     packet[10]   #
100
```

0x46 - Health of Receiver

Field	Description	Notes
0	0x46	
1	None	
2	DESC	
3	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x46:
...     packet[1]      # None
None
...     packet[2]    #
100
...     packet[3]    #
100
```

0x47 - Signal Levels for all Satellites

Field	Description	Notes
0	0x47	
1	None	
2	DESC	
3	DESC	
4	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x47:
...     packet[1]      # None
None
...     packet[2]    #
100
```

```
...     packet[3]    #
100
...     packet[4]    #
1.0
```

0x4A - Single Precision LLA Position Fix

Field	Description	Notes
0	0x4a	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x4a:
...     packet[1]       # None
None
...     packet[2]    #
1.0
...     packet[3]    #
1.0
...     packet[4]    #
1.0
...     packet[5]    #
1.0
...     packet[6]    #
1.0
```

0x4B - Machine/Code ID and Additional Status

Field	Description	Notes
0	0x4b	
1	None	
2	DESC	
3	DESC	
4	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x4b:
...     packet[1]       # None
None
...     packet[2]    #
100
...     packet[3]    #
100
```

```
...     packet[4]     #
100
```

0x4D - Oscillator Offset

Field	Description	Notes
0	0x4d	
1	None	
2	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x4d:
...     packet[1]       # None
None
...     packet[2]     #
1.0
```

0x4E - Response to Set GPS Time

Field	Description	Notes
0	0x4e	
1	None	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x4e:
...     packet[1]       # None
None
```

0x55 - I/O Options

Field	Description	Notes
0	0x55	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x55:
...     packet[1]       # None
None
...     packet[2]     #
100
...     packet[3]     #
```

```
100
...     packet[4]    #
100
...     packet[5]    #
100
```

0x56 - Velocity Fix, East-North-Up (ENU)

Field	Description	Notes
0	0x56	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x56:
...     packet[1]       # None
None
...     packet[2]    #
1.0
...     packet[3]    #
1.0
...     packet[4]    #
1.0
...     packet[5]    #
1.0
...     packet[6]    #
1.0
```

0x57 - Information About Last Computed Fix

Field	Description	Notes
0	0x57	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x57:
...     packet[1]       # None
None
...     packet[2]    #
100
...     packet[3]    #
```

```
100
...     packet[4]    #
1.0
...     packet[5]    #
100
```

0x58 - Satellite System Data/Acknowledge from Receiver

Field	Description	Notes
0	0x58	
1	None	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x58:
...     packet[1]      # None
None
```

0x5A - Raw Measurement Data

Field	Description	Notes
0	0x5a	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	
7	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x5a:
...     packet[1]      # None
None
...     packet[2]    #
100
...     packet[3]    #
1.0
...     packet[4]    #
1.0
...     packet[5]    #
1.0
...     packet[6]    #
1.0
...     packet[7]    #
1.0
```

0x5C - Satellite Tracking Status

Field	Description	Notes
0	0x5c	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	
7	DESC	
8	DESC	
9	DESC	
10	DESC	
11	DESC	
12	DESC	
11	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x5c:
...     packet[1]      # None
None
...     packet[2]    #
100
...     packet[3]    #
100
...     packet[4]    #
100
...     packet[5]    #
100
...     packet[6]    #
1.0
...     packet[7]    #
1.0
...     packet[8]    #
1.0
...     packet[9]    #
1.0
...     packet[10]   #
100
...     packet[11]   #
100
...     packet[12]   #
100
...     packet[11]   #
100
```

0x5F - Diagnostic Use Only

Field	Description	Notes
0	0x5f	
1	None	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x5f:
...     packet[1]       # None
None
```

0x6D - All-In-View Satellite Selection

Field	Description	Notes
0	0x6d	
1	Dimension	
2	PDOP	
3	HDOP	
4	VDOP	
5	TDOP	
x	SV PRN	

```
>>> report = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if report[0] == 0x6d:
...     print report
Packet(0x6d, 3, 10.0, 20.0, 30.0, 40.0, -1, -2, 3)
...     print 'SV in view: %s' % (report[6:])
SV in view: [-1, -2, 3]
```

0x82 - SBAS Correction Status

Field	Description	Notes
0	0x82	
1	SBAS status bits	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x82:
...     packet[1]
2
```

0x83 - Double-Precision XYZ Position Fix and Bias Information

Field	Description	Notes
0	0x83	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x83:
...     packet[1]       # None
None
...     packet[2]     #
1.0
...     packet[3]     #
1.0
...     packet[4]     #
1.0
...     packet[5]     #
1.0
...     packet[6]     #
1.0
```

0x84 - Double-Precision LLA Position Fix and Bias Information

Field	Description	Notes
0	0x84	
1	None	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x84:
...     packet[1]       # None
None
...     packet[2]     #
1.0
...     packet[3]     #
1.0
...     packet[4]     #
1.0
...     packet[5]     #
1.0
...     packet[6]     #
1.0
```

0x8F-15 - Current Datum Values

Field	Description	Notes
0	0x8f	
1	0x15	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	
7	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x8f:
...     packet[1]       # 0x15
21
...     packet[2]     #
100
...     packet[3]     #
1.0
...     packet[4]     #
1.0
...     packet[5]     #
1.0
...     packet[6]     #
1.0
...     packet[7]     #
1.0
```

0x8F-20 - Last Fix with Extra Information (binary fixed point)

Field	Description	Notes
0	0x8f	
1	0x20	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x8f:
...     packet[1]       # 0x20
32
```

0x8F-21 - Request Accuracy Information

Field	Description	Notes
0	0x8f	
1	0x21	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
```

```
>>> if packet[0] == 0x8f:
...     packet[1]       # 0x21
33
```

0x8F-23 - Request Last Compact Fix Information

Field	Description	Notes
0	0x8f	
1	0x23	
2	DESC	
3	DESC	
4	DESC	
5	DESC	
6	DESC	
7	DESC	
8	DESC	
9	DESC	
10	DESC	
11	DESC	
12	DESC	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x8f:
...     packet[1]       # 0x23
35
...     packet[2]    #
100
...     packet[3]    #
100
...     packet[4]    #
100
...     packet[5]    #
100
...     packet[6]    #
100
...     packet[7]    #
100
...     packet[8]    #
100
...     packet[9]    #
100
...     packet[10]   #
100
...     packet[11]   #
100
...     packet[12]   #
100
```

0x8F-26 - Non-Volatile Memory Status

Field	Description	Notes
0	0x8f	
1	0x26	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x8f:
...     packet[1]       # 0x26
38
```

0x8F-2A - Fix and Channel Tracking Info, Type 1

Field	Description	Notes
0	0x8f	
1	0x2a	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x8f:
...     packet[1]       # 0x2a
42
```

0x8F-2B - Fix and Channel Tracking Info, Type 2

Field	Description	Notes
0	0x8f	
1	0x2b	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x8f:
...     packet[1]       # 0x2b
43
```

0x8F-4F - Set PPS Width

Field	Description	Notes
0	0x8f	
1	0x4f	

```
>>> packet = gps.read()
>>> isinstance(packet, tsip.Packet)
True
>>> if packet[0] == 0x8f:
...     packet[1]       # 0x4f
79
```

Adding new TSIP packets

The high-level API provides a simple mechanism for adding new TSIP packets. TODO: Describe this!

Low-Level API

The low-level API can be used to communicate with a Trimble GPS on a binary level. This may be useful if a TSIP packet has not been implemented in the high-level API. The low-level API requires the developer to be familiar with the TSIP packet structure and "byte-stuffing".

The example below encodes TSIP packet 0x1c:0x01 (Command packet 0x1C:01 - Firmware version) and sends it to the GPS.

```
>>> import tsip
>>> import serial     # pySerial (https://pypi.python.org/pypi/pyserial)
>>> serial_conn = serial.Serial('/dev/ttyS0', 9600)
>>> gps_conn = tsip.gps(serial_conn)          # lower-case tsip.gps!
>>> packet = tsip.frame(tsip.stuff(tsip.DLE + '\x1c\x01' + tsip.DLE + tsip.ETX)
>>> gps_conn.write(packet)
>>> while True:       # should implement timeout here!!!
...     report = tsip.unstuff(tsip.unframe(gps_conn.read()))
...     if report[0] == '\x1c' and report[1] == '\x81':
...         print 'Product name: %s' % report[11:]
...         break
```

CHAPTER 5

python-TSIP API

While the low-level and high-level API are implemented in separate modules both are also directly accessible by importing the top-level *tsip* module.

High-level API

High-level API.

class `tsip.hlapi.`**`Packet`**(**fields*)

TSIP packet.

Check the TSIP reference documentation for the description of individual packets.

The argument(s) to *Packet()* can be either individual values, a single tuple or a single list.

Examples:

```
>>> pkt = Packet(0x1f)                          # Request software versions.
>>> pkt = Packet(0x1e, 0x4b)                    # Request cold-start.
>>> pkt = Packet( (0x23, -37.1, 144.1, 10.0) )  # Set initial position (tuple).
>>> pkt = Packet( [0x8e, 0x4f, 0.1] )           # Set PPS with to 0.1s (list)
```

`pack`()

Return binary format of packet.

The returned string is the binary format of the packet with stuffing and framing applied. It is ready to be sent to the GPS.

classmethod `unpack`(*rawpacket*)

Instantiate *Packet* from binary string.

Parameters **`rawpacket`** (*`String.`*) – TSIP pkt in binary format.

rawpacket must already have framing (DLE...DLE/ETX) removed and byte stuffing reversed.

Low-level API

Low-level API.

tsip.llapi.**frame**(*data*)
Add leading DLE and trailing DLE/ETX to data.

Parameters **data** (*Binary string.*) – TSIP data without leading DLE and trailing DLE/ETX.

Returns TSIP data with leading DLE and trailing DLE/ETX added.

Raise ValueError if *data* already starts with DLE and ends in DLE/ETX.

tsip.llapi.**is_framed**(*packet*)
Check whether a packet contains leading DLE and trailing DLE/ETX.

Parameters **packet** (*Binary string.*) – TSIP packet with or without leading DLE and trailing DLE/ETX.

Returns True if leading DLE and trailing DLE/ETX are still present, False otherwise.

tsip.llapi.**stuff**(*packet*)
Add byte stuffing to TSIP packet. :param packet: TSIP packet with byte stuffing. The packet must already

have been stripped or *ValueError* will be raised.

Returns Packet with byte stuffing.

tsip.llapi.**unframe**(*packet*)
Strip leading DLE and trailing DLE/ETX from packet.

Parameters **packet** (*Binary string.*) – TSIP packet with leading DLE and trailing DLE/ETX.

Returns TSIP packet with leading DLE and trailing DLE/ETX removed.

Raise ValueError if *packet* does not start with DLE and end in DLE/ETX.

tsip.llapi.**unstuff**(*packet*)
Remove byte stuffing from a TSIP packet.

Parameters **packet** (*Binary string.*) – TSIP packet with byte stuffing. The packet must already have been stripped or *ValueError* will be raised.

Returns Packet without byte stuffing.

CHAPTER 6

Credits

Development Lead

- Markus Juenemann (markus@juenemann.net)

Contributors

- Eliot Blennerhassett contributed code, lots of really good ideas and reminded me that other people may use *python-TSIP* in ways I hadn't even anticipated.
- Thanks to Jonathan Eisch for making me re-start working on this project again.
- Daniel Macia Fernandez inspired me to work on "obscure" TSIP packets I never though anyone would ever be interested in. The file *docs/examples/example2.py* is loosely based on code we worked on together.

CHAPTER 7

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/mjuenema/python-TSIP/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

Write Documentation

Python TSIP could always use more documentation, whether as part of the official Python TSIP docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/mjuenema/python-TSIP/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Note: Some of the steps described below may not work yet.

Ready to contribute? Here's how to set up *python-TSIP* for local development. Most of the commands are accessible through the `mMkefile`.

1. Install the gitflow Git add-on. Gitflow implements the work-flow described in A successful Git branching model.
2. Fork the *python-TSIP* repo on GitHub.
3. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/python-TSIP.git
```

4. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv python-TSIP
$ cd python-TSIP/
$ python setup.py develop
```

5. Initialise Gitflow:

```
$ git flow init -d
```

6. Start a new feature or branch:

```
$ git flow feature start <name-of-your-feature>
```

   Now you can make your changes locally.

7. When you're done making changes, check that your changes pass flake8 and the tests:

```
$ make flake8
$ make test
```

8. If you have other Python versions installed, use the *tox* tool to test *python-TSIP* against them, too. You may have to adjust `tox.ini` to match your environment but please don't `git add tox.ini`.

```
$ make tox
```

To get flake8 and tox, just pip install them into your virtualenv. You may have to adjust your `PATH` before running `make tox` so that the respective Python interpreters are found. In my setup I the different Python versions are installed under `/opt/Python-<version>`:

```
$ export PATH=$PATH:`echo /opt/Python-*/bin | tr ' ' ':'`
```

9. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin feature/<name-of-your-feature>
```

10. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/mjuenema/python-TSIP/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ nosetests tests/test_<name>.py
```

CHAPTER 8

History

0.3.2 (28-Oct-2017)

- Added Report Packet 0x58: GPS System Data from the Receiver.
- Fixed Report Packet 0x47: Signals Levels for Tracked Satellites.
- Fixed Report Packet 0x6D: Satellite Selection List.x
- Added *docs/examples/example2.py* wich provides a more comprehensive template program than the example in the README.

0.3.1 (26-Sep-2017)

- Fixed README.rst so it renders properly on PyPi.

0.3.0 (26-Sep-2017)

- Argument to `tsip.Packet()` can now also be a tuple or list (inspired by Criss Swaim).
- Changed development status from alpha to beta.
- Cleaned up lots of code to satisfy flake8.

0.2.0 (03-Dec-2015)

- Rewritten from scratch.
- Implements almost complete set of TSIP commands supported by Trimble Copernicus II and Thunderbolt/Thunderbolt-E GPS receivers.

0.1.0 (20-Jun-2015)

- First release on PyPI.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index