

Unittest for stringtools

December 30, 2019

Contents

1	Test Information	4
1.1	Test Candidate Information	4
1.2	Unittest Information	4
1.3	Test System Information	4
2	Statistic	4
2.1	Test-Statistic for testrun with python 2.7.17 (final)	4
2.2	Test-Statistic for testrun with python 3.6.9 (final)	5
2.3	Coverage Statistic	5
3	Tested Requirements	6
3.1	Stream Definition	6
3.2	Stream to Human readable String	6
3.2.1	Hexadecimal Values	6
3.2.2	Number of Bytes	7
3.2.3	CRLF-Filter	7
3.3	Stream Compression	8
3.3.1	Compress	8
3.3.2	Extract	10
3.4	Carriagereturn Seperation Protocol (CSP)	11
3.4.1	Frame creation	11
3.4.2	Frame creation error	13
3.4.3	Frame processing	14
3.4.4	Frame processing - Input data type error	15
3.5	Serial Transfer Protocol (STP)	16
3.5.1	Frame creation	16
3.5.2	Frame creation - Start pattern and end pattern inside a message	17
3.5.3	Frame processing	18
3.5.4	Frame processing - Input data type error	19
3.5.5	Frame processing - Start pattern and end pattern inside a message	20
3.5.6	Frame processing - Data before the start pattern	20
3.5.7	Frame processing - Incorrect start patterns	21
3.5.8	Frame processing - Incorrect end pattern	22
3.5.9	Frame processing - After state corruption	23

A	Trace for testrun with python 2.7.17 (final)	25
A.1	Tests with status Info (18)	25
A.1.1	Hexadecimal Values	25
A.1.2	Number of Bytes	25
A.1.3	CRLF-Filter	26
A.1.4	Compress	26
A.1.5	Extract	27
A.1.6	Frame creation	28
A.1.7	Frame creation error	28
A.1.8	Frame processing	29
A.1.9	Frame processing - Input data type error	29
A.1.10	Frame creation	30
A.1.11	Frame creation - Start pattern and end pattern inside a message	31
A.1.12	Frame processing	31
A.1.13	Frame processing - Input data type error	32
A.1.14	Frame processing - Start pattern and end pattern inside a message	33
A.1.15	Frame processing - Data before the start pattern	34
A.1.16	Frame processing - Incorrect start patterns	35
A.1.17	Frame processing - Incorrect end pattern	36
A.1.18	Frame processing - After state corruption	38
B	Trace for testrun with python 3.6.9 (final)	39
B.1	Tests with status Info (18)	39
B.1.1	Hexadecimal Values	39
B.1.2	Number of Bytes	39
B.1.3	CRLF-Filter	40
B.1.4	Compress	40
B.1.5	Extract	41
B.1.6	Frame creation	42
B.1.7	Frame creation error	42
B.1.8	Frame processing	43

B.1.9	Frame processing - Input data type error	43
B.1.10	Frame creation	44
B.1.11	Frame creation - Start pattern and end pattern inside a message	45
B.1.12	Frame processing	45
B.1.13	Frame processing - Input data type error	46
B.1.14	Frame processing - Start pattern and end pattern inside a message	47
B.1.15	Frame processing - Data before the start pattern	48
B.1.16	Frame processing - Incorrect start patterns	49
B.1.17	Frame processing - Incorrect end pattern	50
B.1.18	Frame processing - After state corruption	52
C	Test-Coverage	53
C.1	stringtools	53
C.1.1	stringtools.__init__.py	53
C.1.2	stringtools.csp.py	56
C.1.3	stringtools.stp.py	57

1 Test Information

1.1 Test Candidate Information

The Module `stringtools` is designed to support functionality for strings (e.g. transfer strings via a bytestream, compressing, extracting, ...). For more Information read the sphinx documentation.

Library Information

Name	stringtools
State	Released
Supported Interpreters	python2, python3
Version	88a3eed174bd2239a6c1d928081e5b6d

Dependencies

1.2 Unittest Information

Unittest Information

Version	d720b191532eb97d77fbb9aafb67181a
Testruns with	python 2.7.17 (final), python 3.6.9 (final)

1.3 Test System Information

System Information

Architecture	64bit
Distribution	LinuxMint 19.3 tricia
Hostname	ahorn
Kernel	5.0.0-37-generic (#40 18.04.1-Ubuntu SMP Thu Nov 14 12:06:39 UTC 2019)
Machine	x86_64
Path	/user_data/data/dirk/prj/modules/stringtools/unittest
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 2.7.17 (final)

Number of tests	18
Number of successfull tests	18
Number of possibly failed tests	0
Number of failed tests	0

Executionlevel	Full Test (all defined tests)
Time consumption	0.015s

2.2 Test-Statistic for testrun with python 3.6.9 (final)

Number of tests	18
Number of successfull tests	18
Number of possibly failed tests	0
Number of failed tests	0

Executionlevel	Full Test (all defined tests)
Time consumption	0.014s

2.3 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
stringtools	100.0%	97.1%
stringtools.__init__.py	100.0%	
stringtools.csp.py	100.0%	
stringtools.stp.py	100.0%	

3 Tested Requirements

3.1 Stream Definition

A Stream is from class bytes for python3 and from type str for python2.

3.2 Stream to Human readable String

3.2.1 Hexadecimal Values

Description

A Stream shall be converted to a human readable String containing all bytes as hexadecimal values seperated by a Space.

Reason for the implementation

Make non printable characters printable.

Fitcriterion

A stream shall be converted at least once and the hex values shall exist in the returnvalue in the correct order.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.1!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (23)
Start-Time:	2019-12-30 18:01:07,740
Finished-Time:	2019-12-30 18:01:07,741
Time-Consumption	0.001s

Testsummary:

Info	Checking test pattern de ad be ef (<type 'str'>).
Success	Pattern included all relevant information in the correct order.

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.1!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/__init__.py (23)
Start-Time:	2019-12-30 18:01:08,140
Finished-Time:	2019-12-30 18:01:08,141
Time-Consumption	0.001s

Testsummary:

Info	Checking test pattern de ad be ef (<class 'bytes'>).
Success	Pattern included all relevant information in the correct order.

3.2.2 Number of Bytes

Description

The Length of a Stream surrounded by brackets shall be included in the human readable string.

Reason for the implementation

Show the length of a Stream without counting the seperated values.

Fitcriterion

The described pattern including the decimal number of bytes is included in the string for at least one Stream.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.2!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (24)
Start-Time:	2019-12-30 18:01:07,741
Finished-Time:	2019-12-30 18:01:07,741
Time-Consumption	0.000s

Testsummary:

Info	Checking test pattern with length 4.
Success	'(4)' is in '(4): de ad be ef' at position 0

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.2!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (24)
Start-Time:	2019-12-30 18:01:08,141
Finished-Time:	2019-12-30 18:01:08,141
Time-Consumption	0.000s

Testsummary:

Info	Checking test pattern with length 4.
Success	'(4)' is in '(4): de ad be ef' at position 0

3.2.3 CRLF-Filter

Description

The module stringtools shall have a method to replace carriage return and line feed to their escaped representation.

Reason for the implementation

Replace these characters to make output printable (e.g. for logging a string based protocol).

Fitcriterion

Filter at least one string and check at least one CR and one LF representation.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.3!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (25)
Start-Time:	2019-12-30 18:01:07,741
Finished-Time:	2019-12-30 18:01:07,741
Time-Consumption	0.000s

Testsummary:

Info	Checking test pattern with length 4.
Success	Returnvalue of linefeed_filter is correct (Content 'test//r//n123//r//n' and Type is <type 'str'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.3!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (25)
Start-Time:	2019-12-30 18:01:08,141
Finished-Time:	2019-12-30 18:01:08,142
Time-Consumption	0.000s

Testsummary:

Info	Checking test pattern with length 4.
Success	Returnvalue of linefeed_filter is correct (Content b'test//r//n123//r//n' and Type is <class 'bytes'>).

3.3 Stream Compression

3.3.1 Compress

Description

The module stringtools shall have a method compressing a Stream with gzip.

Reason for the implementation

Speed up transfer with low transfer rate.

Fitcriterion

Compressed Stream is extractable and results in the original data.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.4!

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.5!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (29)
Start-Time:	2019-12-30 18:01:07,742
Finished-Time:	2019-12-30 18:01:07,743
Time-Consumption	0.000s

Testsummary:

Info	Extracting stream: (26): 1f 8b 08 00 34 e0 04 5d 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00
Success	Extracted data is correct (Content '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff' and Type is <type 'str'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.5!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (29)
Start-Time:	2019-12-30 18:01:08,142
Finished-Time:	2019-12-30 18:01:08,143
Time-Consumption	0.000s

Testsummary:

Info	Extracting stream: (26): 1f 8b 08 00 34 e0 04 5d 02 ff 63 60 40 01 ff 51 01 00 2d 8a 7d de 1e 00 00 00
Success	Extracted data is correct (Content '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff' and Type is <class 'str'>).

3.4 Carriagereturn Seperation Protocol (CSP)

The Carriagereturn Seperation Protocol shall use carriage return as the end pattern for message seperation.

3.4.1 Frame creation

Description

The CSP module shall support a method to create a Frame from a stream.

Reason for the implementation

Simple message or frame generation for streams (e.g. Keyboard (user input), RFID-Reader, ...).

Fitcriterion

Creation of a testframe and checking the result.

Unittest for stringtools

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.6!

Testrun: python 2.7.17 (final)
 Caller: /user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (33)
 Start-Time: 2019-12-30 18:01:07,743
 Finished-Time: 2019-12-30 18:01:07,743
 Time-Consumption 0.000s

Testsummary:

Info Creating testframe for ":testframe: for csp"
Success CSP-Frame is correct (Content ':testframe: for csp/n' and Type is <type 'str'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.6!

Testrun: python 3.6.9 (final)
 Caller: /user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (33)
 Start-Time: 2019-12-30 18:01:08,143
 Finished-Time: 2019-12-30 18:01:08,143
 Time-Consumption 0.000s

Testsummary:

Info Creating testframe for 'b':testframe: for csp"
Success CSP-Frame is correct (Content b':testframe: for csp/n' and Type is <class 'bytes'>).

3.4.2 Frame creation error

Description

The Frame creation Method shall raise ValueError, if a frame separation character is in the Source-String.

Reason for the implementation

String including separation charcter will be splitted in pieces while processing after transport.

Fitcriterion

ValueError is raised for at least one String including the separation character.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.7!

Testrun: python 2.7.17 (final)
 Caller: /user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (34)
 Start-Time: 2019-12-30 18:01:07,743
 Finished-Time: 2019-12-30 18:01:07,743
 Time-Consumption 0.000s

Testsummary:

Info Creating testframe for ":testframe: for csp"
Success CSP-Frame is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.7!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (34)
Start-Time:	2019-12-30 18:01:08,143
Finished-Time:	2019-12-30 18:01:08,143
Time-Consumption	0.000s

Testsummary:

Info	Creating testframe for 'b':testframe: for csp"
Success	CSP-Frame is correct (Content <class 'ValueError'> and Type is <class 'type'>).

3.4.3 Frame processing

Description

The CSP Module shall support a class including a method to process stream snippets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.8!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (35)
Start-Time:	2019-12-30 18:01:07,743
Finished-Time:	2019-12-30 18:01:07,744
Time-Consumption	0.000s

Testsummary:

Info	Processing testframe: ":testframe: for csp/n" in two snippets
Success	First processed CSP-Snippet is correct (Content [] and Type is <type 'list'>).
Success	Final processed CSP-Frame is correct (Content [' :testframe: for csp'] and Type is <type 'list'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.8!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (35)
Start-Time:	2019-12-30 18:01:08,143
Finished-Time:	2019-12-30 18:01:08,144

Time-Consumption 0.000s

Testsummary:

Info Processing testframe: 'b':testframe: for csp/n" in two snippets
Success First processed CSP-Snippet is correct (Content [] and Type is <class 'list'>).
Success Final processed CSP-Frame is correct (Content [b':testframe: for csp'] and Type is <class 'list'>).

3.4.4 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.9!

Testrun: python 2.7.17 (final)
 Caller: /user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (36)
 Start-Time: 2019-12-30 18:01:07,744
 Finished-Time: 2019-12-30 18:01:07,745
 Time-Consumption 0.001s

Testsummary:

Info Processing wrong data (list)
Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success Buffer still empty is correct (Content "" and Type is <type 'str'>).
Info Processing wrong data (int)
Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success Buffer still empty is correct (Content "" and Type is <type 'str'>).
Info Processing wrong data (unicode)
Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success Buffer still empty is correct (Content "" and Type is <type 'str'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.9!

Testrun: python 3.6.9 (final)

Caller: /user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (36)
 Start-Time: 2019-12-30 18:01:08,144
 Finished-Time: 2019-12-30 18:01:08,145
 Time-Consumption 0.001s

Testsummary:

Info Processing wrong data (list)
Success Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success Buffer still empty is correct (Content b" and Type is <class 'bytes'>).
Info Processing wrong data (int)
Success Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success Buffer still empty is correct (Content b" and Type is <class 'bytes'>).
Info Processing wrong data (str)
Success Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success Buffer still empty is correct (Content b" and Type is <class 'bytes'>).

3.5 Serial Transfer Protocol (STP)

The Serial Transfer Protocol shall use a start pattern and an end pattern to identify a message in a stream. Both patterns shall be a two byte values starting with the same (sync-)byte.

3.5.1 Frame creation

Description

A frame creation method shall create a frame out of given input data.

Reason for the implementation

Message or Frame generation for streams (e.g. data transfer via bluetooth, ethernet, ...).

Fitcriterion

Creation of a testframe and checking the result.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.10!

Testrun: python 2.7.17 (final)
 Caller: /user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (40)
 Start-Time: 2019-12-30 18:01:07,745
 Finished-Time: 2019-12-30 18:01:07,745
 Time-Consumption 0.000s

Testsummary:

Info Creating testframe for "testframe for stp"
Success STP-Frame is correct (Content '<testframe for stp:>' and Type is <type 'str'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.10!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (40)
Start-Time:	2019-12-30 18:01:08,145
Finished-Time:	2019-12-30 18:01:08,145
Time-Consumption	0.000s

Testsummary:

Info	Creating testframe for 'b'testframe for stp'
Success	STP-Frame is correct (Content b':<testframe for stp:>' and Type is <class 'bytes'>).

3.5.2 Frame creation - Start pattern and end pattern inside a message

Description

The frame creation method shall support existence of the start or end pattern in the data to be framed.

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Fitcriterion

Creation of a testframe out of data including at least one start pattern and one end pattern and checking the result.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.11!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (41)
Start-Time:	2019-12-30 18:01:07,746
Finished-Time:	2019-12-30 18:01:07,746
Time-Consumption	0.000s

Testsummary:

Info	Creating testframe including start and end pattern for "testframe for :<stp:>"
Success	STP-Frame is correct (Content ':<testframe for :=<stp:=>:' and Type is <type 'str'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.11!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/...init...py (41)
Start-Time:	2019-12-30 18:01:08,145
Finished-Time:	2019-12-30 18:01:08,145
Time-Consumption	0.000s

Testsummary:

Info	Creating testframe including start and end pattern for 'b'testframe for :<stp:>'
Success	STP-Frame is correct (Content b':<testframe for :=<stp:=>:>' and Type is <class 'bytes'>).

3.5.3 Frame processing

Description

The STP Module shall support a class including a method to process stream snippets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.12!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (42)
Start-Time:	2019-12-30 18:01:07,746
Finished-Time:	2019-12-30 18:01:07,747
Time-Consumption	0.001s

Testsummary:

Info	Processing testframe: ":<testframe for stp:>"
Success	First processed STP snippet is correct (Content [] and Type is <type 'list'>).
Success	Final processed STP snippet is correct (Content ['testframe for stp'] and Type is <type 'list'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.12!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (42)
Start-Time:	2019-12-30 18:01:08,145
Finished-Time:	2019-12-30 18:01:08,146
Time-Consumption	0.001s

Testsummary:

Info	Processing testframe: 'b':<testframe for stp:>'
Success	First processed STP snippet is correct (Content [] and Type is <class 'list'>).
Success	Final processed STP snippet is correct (Content [b'testframe for stp'] and Type is <class 'list'>).

3.5.4 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.13!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (43)
Start-Time:	2019-12-30 18:01:07,747
Finished-Time:	2019-12-30 18:01:07,748
Time-Consumption	0.001s

Testsummary:

Info	Processing wrong data (list)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content "" and Type is <type 'str'>).
Info	Processing wrong data (int)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content "" and Type is <type 'str'>).
Info	Processing wrong data (unicode)
Success	Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).
Success	Buffer still empty is correct (Content "" and Type is <type 'str'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.13!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (43)
Start-Time:	2019-12-30 18:01:08,146
Finished-Time:	2019-12-30 18:01:08,147
Time-Consumption	0.001s

Testsummary:

Info	Processing wrong data (list)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success	Buffer still empty is correct (Content b"" and Type is <class 'bytes'>).

Info Processing wrong data (int)
Success Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success Buffer still empty is correct (Content b" and Type is <class 'bytes'>).
Info Processing wrong data (str)
Success Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Success Buffer still empty is correct (Content b" and Type is <class 'bytes'>).

3.5.5 Frame processing - Start pattern and end pattern inside a message

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.14!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (44)
Start-Time:	2019-12-30 18:01:07,748
Finished-Time:	2019-12-30 18:01:07,749
Time-Consumption	0.001s

Testsummary:

Info Processing testframe: " :<testframe for :=<stp:=>:>"
Success Processed STP-Frame is correct (Content ['testframe for :<stp:>'] and Type is <type 'list'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.14!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (44)
Start-Time:	2019-12-30 18:01:08,147
Finished-Time:	2019-12-30 18:01:08,148
Time-Consumption	0.001s

Testsummary:

Info Processing testframe: 'b':<testframe for :=<stp:=>:>"
Success Processed STP-Frame is correct (Content [b'testframe for :<stp:>'] and Type is <class 'list'>).

3.5.6 Frame processing - Data before the start pattern

Description

Data before the start pattern shall be ignored. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.15!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (45)
Start-Time:	2019-12-30 18:01:07,749
Finished-Time:	2019-12-30 18:01:07,749
Time-Consumption	0.001s

Testsummary:

Info	Processing testframe: " _:<testframe for stp:>"
Success	Processed STP-Frame is correct (Content ['testframe for stp'] and Type is <type 'list'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.15!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (45)
Start-Time:	2019-12-30 18:01:08,148
Finished-Time:	2019-12-30 18:01:08,149
Time-Consumption	0.001s

Testsummary:

Info	Processing testframe: 'b' _:<testframe for stp:>"
Success	Processed STP-Frame is correct (Content [b'testframe for stp'] and Type is <class 'list'>).

3.5.7 Frame processing - Incorrect start patterns

Description

On receiving an incorrect start pattern, STP shall stay in ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty start patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.16!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (46)
Start-Time:	2019-12-30 18:01:07,749
Finished-Time:	2019-12-30 18:01:07,751
Time-Consumption	0.002s

Testsummary:

Info	Processing data with an insufficient start pattern.
-------------	---

Success	Return value list if processing incorrect start of frame is correct (Content [[]] and Type is <type 'list'>).
Success	State after processing incorrect start of frame is correct (Content 0 and Type is <type 'int'>).
Info	Processing data with an insufficient start pattern (two times sync).
Success	Return value list if processing data_sync twice is correct (Content [[]] and Type is <type 'list'>).
Success	State after processing data_sync twice is correct (Content 1 and Type is <type 'int'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.16!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (46)
Start-Time:	2019-12-30 18:01:08,149
Finished-Time:	2019-12-30 18:01:08,152
Time-Consumption	0.003s

Testsummary:

Info	Processing data with an insufficient start pattern.
Success	Return value list if processing incorrect start of frame is correct (Content [[]] and Type is <class 'list'>).
Success	State after processing incorrect start of frame is correct (Content 0 and Type is <class 'int'>).
Info	Processing data with an insufficient start pattern (two times sync).
Success	Return value list if processing data_sync twice is correct (Content [[]] and Type is <class 'list'>).
Success	State after processing data_sync twice is correct (Content 1 and Type is <class 'int'>).

3.5.8 Frame processing - Incorrect end pattern

Description

On receiving an incorrect end pattern, STP shall change to state STORE_DATA, in case of a start pattern, to ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty end patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.17!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (47)
Start-Time:	2019-12-30 18:01:07,751
Finished-Time:	2019-12-30 18:01:07,756
Time-Consumption	0.005s

Testsummary:

Info	Processing data with an insufficient end pattern.
-------------	---

Success	Return value list if processing data_sync and data again after start of frame is correct (Content [[]] and Type is <type 'list'>).
Success	State after processing data_sync and data again after start of frame is correct (Content 0 and Type is <type 'int'>).
Success	Buffer size after processing data with insufficient end pattern is correct (Content 0 and Type is <type 'int'>).
Info	Processing data with an insufficient end pattern (start pattern instead of end pattern).
Success	Return value list if processing 2nd start of frame is correct (Content [[]] and Type is <type 'list'>).
Success	State after processing 2nd start of frame is correct (Content 3 and Type is <type 'int'>).
Success	Buffer size after processing 2nd start of frame is correct (Content 0 and Type is <type 'int'>).
Info	Processing data with an insufficient end pattern (two times sync instead of end pattern).
Success	Return value list if processing data_sync twice after start of frame is correct (Content [[]] and Type is <type 'list'>).
Success	State after processing data_sync twice after start of frame is correct (Content 1 and Type is <type 'int'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.17!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (47)
Start-Time:	2019-12-30 18:01:08,152
Finished-Time:	2019-12-30 18:01:08,155
Time-Consumption	0.003s

Testsummary:

Info	Processing data with an insufficient end pattern.
Success	Return value list if processing data_sync and data again after start of frame is correct (Content [[]] and Type is <class 'list'>).
Success	State after processing data_sync and data again after start of frame is correct (Content 0 and Type is <class 'int'>).
Success	Buffer size after processing data with insufficient end pattern is correct (Content 0 and Type is <class 'int'>).
Info	Processing data with an insufficient end pattern (start pattern instead of end pattern).
Success	Return value list if processing 2nd start of frame is correct (Content [[]] and Type is <class 'list'>).
Success	State after processing 2nd start of frame is correct (Content 3 and Type is <class 'int'>).
Success	Buffer size after processing 2nd start of frame is correct (Content 0 and Type is <class 'int'>).
Info	Processing data with an insufficient end pattern (two times sync instead of end pattern).
Success	Return value list if processing data_sync twice after start of frame is correct (Content [[]] and Type is <class 'list'>).
Success	State after processing data_sync twice after start of frame is correct (Content 1 and Type is <class 'int'>).

3.5.9 Frame processing - After state corruption

Description

The state of STP shall be set to IDLE, after an unknown state was recognised. The currently processed data shall be

processed again. An error shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**. See also full trace in section A.1.18!

Testrun:	python 2.7.17 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (48)
Start-Time:	2019-12-30 18:01:07,756
Finished-Time:	2019-12-30 18:01:07,758
Time-Consumption	0.001s

Testsummary:

Info	Corrupting stp state and processing data.
Success	Return value list if processing start of a frame after state had been corrupted is correct (Content [[]] and Type is <type 'list'>).
Success	State after processing start of a frame after state had been corrupted is correct (Content 3 and Type is <type 'int'>).
Success	Buffer size after corrupting stp state is correct (Content 2 and Type is <type 'int'>).

Testresult

This test was passed with the state: **Success**. See also full trace in section B.1.18!

Testrun:	python 3.6.9 (final)
Caller:	/user_data/data/dirk/prj/modules/stringtools/unittest/src/tests/_init_.py (48)
Start-Time:	2019-12-30 18:01:08,155
Finished-Time:	2019-12-30 18:01:08,156
Time-Consumption	0.001s

Testsummary:

Info	Corrupting stp state and processing data.
Success	Return value list if processing start of a frame after state had been corrupted is correct (Content [[]] and Type is <class 'list'>).
Success	State after processing start of a frame after state had been corrupted is correct (Content 3 and Type is <class 'int'>).
Success	Buffer size after corrupting stp state is correct (Content 2 and Type is <class 'int'>).

A Trace for testrun with python 2.7.17 (final)

A.1 Tests with status Info (18)

A.1.1 Hexadecimal Values

Description

A Stream shall be converted to a human readable String containing all bytes as hexadecimal values seperated by a Space.

Reason for the implementation

Make non printable characters printable.

Fitcriterion

A stream shall be converted at least once and the hex values shall exist in the returnvalue in the correct order.

Testresult

This test was passed with the state: **Success**.

Info Checking test pattern de ad be ef (<type 'str'>).

Success Pattern included all relevant information in the correct order.

```
Return value of hexlify is (4): de ad be ef
```

```
Using upper string for comparison: (4): DE AD BE EF
```

```
"DE" found in "(4): DE AD BE EF"... Reducing pattern
```

```
"AD" found in "AD BE EF"... Reducing pattern
```

```
"BE" found in "BE EF"... Reducing pattern
```

```
"EF" found in "EF"... Reducing pattern
```

A.1.2 Number of Bytes

Description

The Length of a Stream surrounded by brakets shall be included in the human readable string.

Reason for the implementation

Show the length of a Stream without counting the seperated values.

Fitcriterion

The described pattern including the decimal number of bytes is included in the string for at least one Stream.

Testresult

This test was passed with the state: **Success**.

Info Checking test pattern with length 4.

Success '(4)' is in '(4): de ad be ef' at position 0

A.1.3 CRLF-Filter

Description

The module stringtools shall have a method to replace carriage return and line feed to their escaped representation.

Reason for the implementation

Replace these characters to make output printable (e.g. for logging a string based protocol).

Fitcriterion

Filter at least one string and check at least one CR and one LF representation.

Testresult

This test was passed with the state: **Success**.

Info Checking test pattern with length 4.

Success Returnvalue of linefeed_filter is correct (Content 'test//r//n123//r//n' and Type is <type 'str'>).

Result (Returnvalue of linefeed_filter): 'test\\r\\n123\\r\\n' (<type 'str'>)

Expectation (Returnvalue of linefeed_filter): result = 'test\\r\\n123\\r\\n' (<type 'str'>)

A.1.4 Compress

Description

The module stringtools shall have a method compressing a Stream with gzip.

Reason for the implementation

Speed up transfer with low transfer rate.

Fitcriterion

Compressed Stream is extractable and results in the original data.

A.1.6 Frame creation

Description

The CSP module shall support a method to create a Frame from a stream.

Reason for the implementation

Simple message or frame generation for streams (e.g. Keyboard (user input), RFID-Reader, ...).

Fitcriterion

Creation of a testframe and checking the result.

Testresult

This test was passed with the state: **Success**.

Info Creating testframe for ":testframe: for csp"

Success CSP-Frame is correct (Content ':testframe: for csp/n' and Type is <type 'str'>).

Result (CSP-Frame): ':testframe: for csp\n' (<type 'str'>)

Expectation (CSP-Frame): result = ':testframe: for csp\n' (<type 'str'>)

A.1.7 Frame creation error

Description

The Frame creation Method shall raise ValueError, if a frame separation character is in the Source-String.

Reason for the implementation

String including separation charcter will be splitted in pieces while processing after transport.

Fitcriterion

ValueError is raised for at least one String including the separation character.

Testresult

This test was passed with the state: **Success**.

Info Creating testframe for ":testframe: for csp"

Success CSP-Frame is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Result (CSP-Frame): <type 'exceptions.ValueError'> (<type 'type'>)

Expectation (CSP-Frame): result = <type 'exceptions.ValueError'> (<type 'type'>)

A.1.8 Frame processing

Description

The CSP Module shall support a class including a method to process stream snippets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

This test was passed with the state: **Success**.

Info Processing testframe: ":testframe: for csp/n" in two snippets

CSP: Leaving data in buffer (to be processed next time): (10): 3a 74 65 73 74 66 72 61 6d 65

CSP: message identified - (19): 3a 74 65 73 74 66 72 61 6d 65 3a 20 66 6f 72 20 63 73 70

Success First processed CSP-Snippet is correct (Content [] and Type is <type 'list'>).

Result (First processed CSP-Snippet): [] (<type 'list'>)

Expectation (First processed CSP-Snippet): result = [] (<type 'list'>)

Success Final processed CSP-Frame is correct (Content [':testframe: for csp'] and Type is <type 'list'>).

Result (Final processed CSP-Frame): [':testframe: for csp'] (<type 'list'>)

Expectation (Final processed CSP-Frame): result = [':testframe: for csp'] (<type 'list'>)

A.1.9 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

This test was passed with the state: **Success**.

Info Processing wrong data (list)

Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Result (Wrong data exception): <type 'exceptions.ValueError'> (<type 'type'>)

Expectation (Wrong data exception): result = <type 'exceptions.ValueError'> (<type 'type'>)

Success Buffer still empty is correct (Content "" and Type is <type 'str'>).

Result (Buffer still empty): '' (<type 'str'>)

Expectation (Buffer still empty): result = '' (<type 'str'>)

Info Processing wrong data (int)

Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Result (Wrong data exception): <type 'exceptions.ValueError'> (<type 'type'>)

Expectation (Wrong data exception): result = <type 'exceptions.ValueError'> (<type 'type'>)

Success Buffer still empty is correct (Content "" and Type is <type 'str'>).

Result (Buffer still empty): '' (<type 'str'>)

Expectation (Buffer still empty): result = '' (<type 'str'>)

Info Processing wrong data (unicode)

Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Result (Wrong data exception): <type 'exceptions.ValueError'> (<type 'type'>)

Expectation (Wrong data exception): result = <type 'exceptions.ValueError'> (<type 'type'>)

Success Buffer still empty is correct (Content "" and Type is <type 'str'>).

Result (Buffer still empty): '' (<type 'str'>)

Expectation (Buffer still empty): result = '' (<type 'str'>)

A.1.10 Frame creation

Description

A frame creation method shall create a frame out of given input data.

Reason for the implementation

Message or Frame generation for streams (e.g. data transfer via bluetooth, ethernet, ...).

Fitcriterion

Creation of a testframe and checking the result.

Testresult

This test was passed with the state: **Success**.

Info	Creating testframe for "testframe for stp"
-------------	--

Success STP-Frame is correct (Content ':<testframe for stp:>' and Type is <type 'str'>).

Result (STP-Frame):	':<testframe for stp:>' (<type 'str'>)
Expectation (STP-Frame):	result = ':<testframe for stp:>' (<type 'str'>)

A.1.11 Frame creation - Start pattern and end pattern inside a message

Description

The frame creation method shall support existence of the start or end pattern in the data to be framed.

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Fitcriterion

Creation of a testframe out of data including at least one start pattern and one end pattern and checking the result.

Testresult

This test was passed with the state: **Success**.

Info	Creating testframe including start and end pattern for "testframe for :<stp:>"
-------------	--

Success STP-Frame is correct (Content ':<testframe for :=<stp:=>:>' and Type is <type 'str'>).

Result (STP-Frame):	':<testframe for :=<stp:=>:>' (<type 'str'>)
Expectation (STP-Frame):	result = ':<testframe for :=<stp:=>:>' (<type 'str'>)

A.1.12 Frame processing

Description

The STP Module shall support a class including a method to process stream snippets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

This test was passed with the state: **Success**.

```

Info Processing testframe: "':<testframe for stp:>"
-----
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_IDLE
STP: message identified - (17): 74 65 73 74 66 72 61 6d 65 20 66 6f 72 20 73 74 70
-----
Success First processed STP snippet is correct (Content [] and Type is <type 'list'>).
-----
Result (First processed STP snippet): [ ] (<type 'list'>)
Expectation (First processed STP snippet): result = [ ] (<type 'list'>)
-----
Success Final processed STP snippet is correct (Content ['testframe for stp'] and Type is <type 'list'>).
-----
Result (Final processed STP snippet): [ 'testframe for stp' ] (<type 'list'>)
Expectation (Final processed STP snippet): result = [ 'testframe for stp' ] (<type 'list'>)

```

A.1.13 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

This test was passed with the state: **Success**.

Info Processing wrong data (list)

Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Result (Wrong data exception): <type 'exceptions.ValueError'> (<type 'type'>)

Expectation (Wrong data exception): result = <type 'exceptions.ValueError'> (<type 'type'>)

Success Buffer still empty is correct (Content "" and Type is <type 'str'>).

Result (Buffer still empty): '' (<type 'str'>)

Expectation (Buffer still empty): result = '' (<type 'str'>)

Info Processing wrong data (int)

Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Result (Wrong data exception): <type 'exceptions.ValueError'> (<type 'type'>)

Expectation (Wrong data exception): result = <type 'exceptions.ValueError'> (<type 'type'>)

Success Buffer still empty is correct (Content "" and Type is <type 'str'>).

Result (Buffer still empty): '' (<type 'str'>)

Expectation (Buffer still empty): result = '' (<type 'str'>)

Info Processing wrong data (unicode)

Success Wrong data exception is correct (Content <type 'exceptions.ValueError'> and Type is <type 'type'>).

Result (Wrong data exception): <type 'exceptions.ValueError'> (<type 'type'>)

Expectation (Wrong data exception): result = <type 'exceptions.ValueError'> (<type 'type'>)

Success Buffer still empty is correct (Content "" and Type is <type 'str'>).

Result (Buffer still empty): '' (<type 'str'>)

Expectation (Buffer still empty): result = '' (<type 'str'>)

A.1.14 Frame processing - Start pattern and end pattern inside a message

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Testresult

This test was passed with the state: **Success**.

```

Info Processing testframe: "':<testframe for :=<stp:=>:'"
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: store sync pattern (3a 3d) received => changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: store sync pattern (3a 3d) received => changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_IDLE
STP: message identified - (21): 74 65 73 74 66 72 61 6d 65 20 66 6f 72 20 3a 3c 73 74 70 3a 3e
    
```

```

Success Processed STP-Frame is correct (Content ['testframe for :<stp:>'] and Type is <type 'list'>).
Result (Processed STP-Frame): [ 'testframe for :<stp:>' ] (<type 'list'>)
Expectation (Processed STP-Frame): result = [ 'testframe for :<stp:>' ] (<type 'list'>)
    
```

A.1.15 Frame processing - Data before the start pattern

Description

Data before the start pattern shall be ignored. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**.

```

Info Processing testframe: "':<testframe for stp:>'
STP: no data sync (5f) received => ignoring byte
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_IDLE
STP: message identified - (17): 74 65 73 74 66 72 61 6d 65 20 66 6f 72 20 73 74 70
    
```

```

Success Processed STP-Frame is correct (Content ['testframe for stp'] and Type is <type 'list'>).
    
```

```
Result (Processed STP-Frame): [ 'testframe for stp' ] (<type 'list'>)
```

```
Expectation (Processed STP-Frame): result = [ 'testframe for stp' ] (<type 'list'>)
```

A.1.16 Frame processing - Incorrect start patterns

Description

On receiving an incorrect start pattern, STP shall stay in ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty start patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**.

Info Processing data with an insufficient start pattern.

```
Sending ':1' to stp.
```

```
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
```

```
STP: no start pattern (3a 31) received => changing state STP_STATE_ESCAPE_1 -> STP_STATE_IDLE
```

Success Return value list if processing incorrect start of frame is correct (Content [[]] and Type is <type 'list'>).

```
Result (Return value list if processing incorrect start of frame): [ [ ] ] (<type 'list'>)
```

```
Expectation (Return value list if processing incorrect start of frame): result = [ [ ] ]  
↪ (<type 'list'>)
```

Success State after processing incorrect start of frame is correct (Content 0 and Type is <type 'int'>).

```
Result (State after processing incorrect start of frame): 0 (<type 'int'>)
```

```
Expectation (State after processing incorrect start of frame): result = 0 (<type 'int'>)
```

Info Processing data with an insufficient start pattern (two times sync).

```
Sending '::' to stp.
```

```
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
```

```
STP: 2nd data sync (3a) received => keep state
```

Success Return value list if processing data_sync twice is correct (Content [[]] and Type is <type 'list'>).

```
Result (Return value list if processing data_sync twice): [ [ ] ] (<type 'list'>)
```

```
Expectation (Return value list if processing data_sync twice): result = [ [ ] ] (<type  
↪ 'list'>)
```

Success State after processing data_sync twice is correct (Content 1 and Type is <type 'int'>).

```
Result (State after processing data_sync twice): 1 (<type 'int'>)
```

```
Expectation (State after processing data_sync twice): result = 1 (<type 'int'>)
```

A.1.17 Frame processing - Incorrect end pattern

Description

On receiving an incorrect end pattern, STP shall change to state STORE_DATA, in case of a start pattern, to ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty end patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**.

Info Processing data with an insufficient end pattern.

Sending ':<te:d' to stp.

STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->

↳ STP_STATE_STORE_DATA

STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2

STP: data (64) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE

STP: Chunking "(2): 74 65" from buffer

Success Return value list if processing data_sync and data again after start of frame is correct (Content [[]] and Type is <type 'list'>).

Result (Return value list if processing data_sync and data again after start of frame): [[]] (<type 'list'>)

Expectation (Return value list if processing data_sync and data again after start of frame):

↳ result = [[]] (<type 'list'>)

Success State after processing data_sync and data again after start of frame is correct (Content 0 and Type is <type 'int'>).

Result (State after processing data_sync and data again after start of frame): 0 (<type 'int'>)

Expectation (State after processing data_sync and data again after start of frame): result = 0 (<type 'int'>)

Success Buffer size after processing data with insufficient end pattern is correct (Content 0 and Type is <type 'int'>).

Result (Buffer size after processing data with insufficient end pattern): 0 (<type 'int'>)

Expectation (Buffer size after processing data with insufficient end pattern): result = 0 (<type 'int'>)

Info Processing data with an insufficient end pattern (start pattern instead of end pattern).

Sending '<te:<' to stp.

STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
 ↪ STP_STATE_STORE_DATA

STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_2 ->
 ↪ STP_STATE_STORE_DATA

STP: Chunking "(2): 74 65" from buffer

Success Return value list if processing 2nd start of frame is correct (Content [[]] and Type is <type 'list'>).

Result (Return value list if processing 2nd start of frame): [[]] (<type 'list'>)

Expectation (Return value list if processing 2nd start of frame): result = [[]] (<type
 ↪ 'list'>)

Success State after processing 2nd start of frame is correct (Content 3 and Type is <type 'int'>).

Result (State after processing 2nd start of frame): 3 (<type 'int'>)

Expectation (State after processing 2nd start of frame): result = 3 (<type 'int'>)

Success Buffer size after processing 2nd start of frame is correct (Content 0 and Type is <type 'int'>).

Result (Buffer size after processing 2nd start of frame): 0 (<type 'int'>)

Expectation (Buffer size after processing 2nd start of frame): result = 0 (<type 'int'>)

Info Processing data with an insufficient end pattern (two times sync instead of end pattern).

Sending '<te::' to stp.

STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
 ↪ STP_STATE_STORE_DATA

STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2

STP: second data sync (3a) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1

STP: Chunking "(2): 74 65" from buffer

Success Return value list if processing data_sync twice after start of frame is correct (Content [[]] and Type is <type 'list'>).

Result (Return value list if processing data_sync twice after start of frame): [[]]
 ↪ (<type 'list'>)

Expectation (Return value list if processing data_sync twice after start of frame): result =
 ↪ [[]] (<type 'list'>)

Success State after processing data_sync twice after start of frame is correct (Content 1 and Type is <type 'int'>).

Result (State after processing data_sync twice after start of frame): 1 (<type 'int'>)

Expectation (State after processing data_sync twice after start of frame): result = 1 (<type
 ↪ 'int'>)

A.1.18 Frame processing - After state corruption

Description

The state of STP shall be set to IDLE, after an unknown state was recognised. The currently processed data shall be processed again. An error shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**.

Info Corrupting stp state and processing data.

Sending ';<te' to stp.

STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->

↳ STP_STATE_STORE_DATA

Setting state of stp to 255.

Sending ';<te' to stp.

STP: unknown state (255) => adding value (3a) back to data again and changing state ->

↳ STP_STATE_IDLE

STP: Chunking "(2): 74 65" from buffer

STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->

↳ STP_STATE_STORE_DATA

Success Return value list if processing start of a frame after state had been corrupted is correct (Content [[]] and Type is <type 'list'>).

Result (Return value list if processing start of a frame after state had been corrupted): [[]] (<type 'list'>)

Expectation (Return value list if processing start of a frame after state had been corrupted): result = [[]] (<type 'list'>)

Success State after processing start of a frame after state had been corrupted is correct (Content 3 and Type is <type 'int'>).

Result (State after processing start of a frame after state had been corrupted): 3 (<type 'int'>)

Expectation (State after processing start of a frame after state had been corrupted): result = 3 (<type 'int'>)

Success Buffer size after corrupting stp state is correct (Content 2 and Type is <type 'int'>).

Result (Buffer size after corrupting stp state): 2 (<type 'int'>)

Expectation (Buffer size after corrupting stp state): result = 2 (<type 'int'>)

B Trace for testrun with python 3.6.9 (final)

B.1 Tests with status Info (18)

B.1.1 Hexadecimal Values

Description

A Stream shall be converted to a human readable String containing all bytes as hexadecimal values seperated by a Space.

Reason for the implementation

Make non printable characters printable.

Fitcriterion

A stream shall be converted at least once and the hex values shall exist in the returnvalue in the correct order.

Testresult

This test was passed with the state: **Success**.

Info Checking test pattern de ad be ef (<class 'bytes'>).

Success Pattern included all relevant information in the correct order.

```
Return value of hexlify is (4): de ad be ef
```

```
Using upper string for comparison: (4): DE AD BE EF
```

```
"DE" found in "(4): DE AD BE EF"... Reducing pattern
```

```
"AD" found in "AD BE EF"... Reducing pattern
```

```
"BE" found in "BE EF"... Reducing pattern
```

```
"EF" found in "EF"... Reducing pattern
```

B.1.2 Number of Bytes

Description

The Length of a Stream surrounded by brakets shall be included in the human readable string.

Reason for the implementation

Show the length of a Stream without counting the seperated values.

Fitcriterion

The described pattern including the decimal number of bytes is included in the string for at least one Stream.

Testresult

This test was passed with the state: **Success**.

Info Checking test pattern with length 4.

Success '(4)' is in '(4): de ad be ef' at position 0

B.1.3 CRLF-Filter

Description

The module stringtools shall have a method to replace carriage return and line feed to their escaped representation.

Reason for the implementation

Replace these characters to make output printable (e.g. for logging a string based protocol).

Fitcriterion

Filter at least one string and check at least one CR and one LF representation.

Testresult

This test was passed with the state: **Success**.

Info Checking test pattern with length 4.

Success Returnvalue of linefeed_filter is correct (Content b'test\r\n123\r\n' and Type is <class 'bytes'>).

Result (Returnvalue of linefeed_filter): b'test\r\n123\r\n' (<class 'bytes'>)

Expectation (Returnvalue of linefeed_filter): result = b'test\r\n123\r\n' (<class
↪ 'bytes'>)

B.1.4 Compress

Description

The module stringtools shall have a method compressing a Stream with gzip.

Reason for the implementation

Speed up transfer with low transfer rate.

Fitcriterion

Compressed Stream is extractable and results in the original data.

B.1.6 Frame creation

Description

The CSP module shall support a method to create a Frame from a stream.

Reason for the implementation

Simple message or frame generation for streams (e.g. Keyboard (user input), RFID-Reader, ...).

Fitcriterion

Creation of a testframe and checking the result.

Testresult

This test was passed with the state: **Success**.

Info Creating testframe for 'b':testframe: for csp"

Success CSP-Frame is correct (Content b':testframe: for csp/n' and Type is <class 'bytes'>).

Result (CSP-Frame): b':testframe: for csp\n' (<class 'bytes'>)

Expectation (CSP-Frame): result = b':testframe: for csp\n' (<class 'bytes'>)

B.1.7 Frame creation error

Description

The Frame creation Method shall raise ValueError, if a frame separation character is in the Source-String.

Reason for the implementation

String including separation charcter will be splitted in pieces while processing after transport.

Fitcriterion

ValueError is raised for at least one String including the separation character.

Testresult

This test was passed with the state: **Success**.

Info Creating testframe for 'b':testframe: for csp"

Success CSP-Frame is correct (Content <class 'ValueError'> and Type is <class 'type'>).

Result (CSP-Frame): <class 'ValueError'> (<class 'type'>)

Expectation (CSP-Frame): result = <class 'ValueError'> (<class 'type'>)

B.1.8 Frame processing

Description

The CSP Module shall support a class including a method to process stream snippets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

This test was passed with the state: **Success**.

Info Processing testframe: 'b':testframe: for csp/n" in two snippets

CSP: Leaving data in buffer (to be processed next time): (10): 3a 74 65 73 74 66 72 61 6d 65

CSP: message identified - (19): 3a 74 65 73 74 66 72 61 6d 65 3a 20 66 6f 72 20 63 73 70

Success First processed CSP-Snippet is correct (Content [] and Type is <class 'list'>).

Result (First processed CSP-Snippet): [] (<class 'list'>)

Expectation (First processed CSP-Snippet): result = [] (<class 'list'>)

Success Final processed CSP-Frame is correct (Content [b':testframe: for csp'] and Type is <class 'list'>).

Result (Final processed CSP-Frame): [b':testframe: for csp'] (<class 'list'>)

Expectation (Final processed CSP-Frame): result = [b':testframe: for csp'] (<class 'list'>)

B.1.9 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

This test was passed with the state: **Success**.

Info Processing wrong data (list)

Success Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

Result (Wrong data exception): <class 'ValueError'> (<class 'type'>)

Expectation (Wrong data exception): result = <class 'ValueError'> (<class 'type'>)

Success Buffer still empty is correct (Content b" and Type is <class 'bytes'>).

Result (Buffer still empty): b'' (<class 'bytes'>)

Expectation (Buffer still empty): result = b'' (<class 'bytes'>)

Info Processing wrong data (int)

Success Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

Result (Wrong data exception): <class 'ValueError'> (<class 'type'>)

Expectation (Wrong data exception): result = <class 'ValueError'> (<class 'type'>)

Success Buffer still empty is correct (Content b" and Type is <class 'bytes'>).

Result (Buffer still empty): b'' (<class 'bytes'>)

Expectation (Buffer still empty): result = b'' (<class 'bytes'>)

Info Processing wrong data (str)

Success Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).

Result (Wrong data exception): <class 'ValueError'> (<class 'type'>)

Expectation (Wrong data exception): result = <class 'ValueError'> (<class 'type'>)

Success Buffer still empty is correct (Content b" and Type is <class 'bytes'>).

Result (Buffer still empty): b'' (<class 'bytes'>)

Expectation (Buffer still empty): result = b'' (<class 'bytes'>)

B.1.10 Frame creation

Description

A frame creation method shall create a frame out of given input data.

Reason for the implementation

Message or Frame generation for streams (e.g. data transfer via bluetooth, ethernet, ...).

Fitcriterion

Creation of a testframe and checking the result.

Testresult

This test was passed with the state: **Success**.

Info Creating testframe for 'b'testframe for stp'

Success STP-Frame is correct (Content b':<testframe for stp:>' and Type is <class 'bytes'>).

Result (STP-Frame): b':<testframe for stp:>' (<class 'bytes'>)

Expectation (STP-Frame): result = b':<testframe for stp:>' (<class 'bytes'>)

B.1.11 Frame creation - Start pattern and end pattern inside a message

Description

The frame creation method shall support existence of the start or end pattern in the data to be framed.

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Fitcriterion

Creation of a testframe out of data including at least one start pattern and one end pattern and checking the result.

Testresult

This test was passed with the state: **Success**.

Info Creating testframe including start and end pattern for 'b'testframe for :<stp:>'

Success STP-Frame is correct (Content b':<testframe for :=<stp:=>:>' and Type is <class 'bytes'>).

Result (STP-Frame): b':<testframe for :=<stp:=>:>' (<class 'bytes'>)

Expectation (STP-Frame): result = b':<testframe for :=<stp:=>:>' (<class 'bytes'>)

B.1.12 Frame processing

Description

The STP Module shall support a class including a method to process stream snippets of variable length. This Method shall return an empty list, if no frame has been detected, otherwise it shall return a list including detected frame(s).

Reason for the implementation

Support message analysis of a stream with every size.

Fitcriterion

At least one frame given in at least two snippets is identified correctly.

Testresult

This test was passed with the state: **Success**.

```

Info Processing testframe: 'b':<testframe for stp:>'
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_IDLE
STP: message identified - (17): 74 65 73 74 66 72 61 6d 65 20 66 6f 72 20 73 74 70

Success First processed STP snippet is correct (Content [] and Type is <class 'list'>).
Result (First processed STP snippet): [ ] (<class 'list'>)
Expectation (First processed STP snippet): result = [ ] (<class 'list'>)

Success Final processed STP snippet is correct (Content [b'testframe for stp'] and Type is <class 'list'>).
Result (Final processed STP snippet): [ b'testframe for stp' ] (<class 'list'>)
Expectation (Final processed STP snippet): result = [ b'testframe for stp' ] (<class 'list'>)

```

B.1.13 Frame processing - Input data type error

Description

If the input data is not bytes for python3 or str for python 2, the process method shall raise TypeError.

Reason for the implementation

Type restriction.

Fitcriterion

At least the following types return TypeError (list, int, str for python3, unicode for python 2).

Testresult

This test was passed with the state: **Success**.

Info	Processing wrong data (list)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Result (Wrong data exception): <class 'ValueError'> (<class 'type'>)	
Expectation (Wrong data exception): result = <class 'ValueError'> (<class 'type'>)	
Success	Buffer still empty is correct (Content b" and Type is <class 'bytes'>).
Result (Buffer still empty): b'' (<class 'bytes'>)	
Expectation (Buffer still empty): result = b'' (<class 'bytes'>)	
Info	Processing wrong data (int)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Result (Wrong data exception): <class 'ValueError'> (<class 'type'>)	
Expectation (Wrong data exception): result = <class 'ValueError'> (<class 'type'>)	
Success	Buffer still empty is correct (Content b" and Type is <class 'bytes'>).
Result (Buffer still empty): b'' (<class 'bytes'>)	
Expectation (Buffer still empty): result = b'' (<class 'bytes'>)	
Info	Processing wrong data (str)
Success	Wrong data exception is correct (Content <class 'ValueError'> and Type is <class 'type'>).
Result (Wrong data exception): <class 'ValueError'> (<class 'type'>)	
Expectation (Wrong data exception): result = <class 'ValueError'> (<class 'type'>)	
Success	Buffer still empty is correct (Content b" and Type is <class 'bytes'>).
Result (Buffer still empty): b'' (<class 'bytes'>)	
Expectation (Buffer still empty): result = b'' (<class 'bytes'>)	

B.1.14 Frame processing - Start pattern and end pattern inside a message

Reason for the implementation

Possibility to send any kind of data (including the patterns).

Testresult

This test was passed with the state: **Success**.

```

Info Processing testframe: 'b':<testframe for :=<stp:=>>'
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: store sync pattern (3a 3d) received => changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: store sync pattern (3a 3d) received => changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_IDLE
STP: message identified - (21): 74 65 73 74 66 72 61 6d 65 20 66 6f 72 20 3a 3c 73 74 70 3a 3e
    
```

```

Success Processed STP-Frame is correct (Content [b'testframe for :<stp:>'] and Type is <class 'list'>).
Result (Processed STP-Frame): [ b'testframe for :<stp:>' ] (<class 'list'>)
Expectation (Processed STP-Frame): result = [ b'testframe for :<stp:>' ] (<class 'list'>)
    
```

B.1.15 Frame processing - Data before the start pattern

Description

Data before the start pattern shall be ignored. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**.

```

Info Processing testframe: 'b':<testframe for stp:>'
STP: no data sync (5f) received => ignoring byte
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↳ STP_STATE_STORE_DATA
STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2
STP: end pattern (3a 3e) received => storing message and changing state STP_STATE_ESCAPE_2 ->
↳ STP_STATE_IDLE
STP: message identified - (17): 74 65 73 74 66 72 61 6d 65 20 66 6f 72 20 73 74 70
    
```

```

Success Processed STP-Frame is correct (Content [b'testframe for stp'] and Type is <class 'list'>).
    
```

```
Result (Processed STP-Frame): [ b'testframe for stp' ] (<class 'list'>)
```

```
Expectation (Processed STP-Frame): result = [ b'testframe for stp' ] (<class 'list'>)
```

B.1.16 Frame processing - Incorrect start patterns

Description

On receiving an incorrect start pattern, STP shall stay in ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty start patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**.

Info Processing data with an insufficient start pattern.

```
Sending b':1' to stp.
```

```
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
```

```
STP: no start pattern (3a 31) received => changing state STP_STATE_ESCAPE_1 -> STP_STATE_IDLE
```

Success Return value list if processing incorrect start of frame is correct (Content [[]] and Type is <class 'list'>).

```
Result (Return value list if processing incorrect start of frame): [ [ ] ] (<class 'list'>)
```

```
Expectation (Return value list if processing incorrect start of frame): result = [ [ ] ]  
↪ (<class 'list'>)
```

Success State after processing incorrect start of frame is correct (Content 0 and Type is <class 'int'>).

```
Result (State after processing incorrect start of frame): 0 (<class 'int'>)
```

```
Expectation (State after processing incorrect start of frame): result = 0 (<class 'int'>)
```

Info Processing data with an insufficient start pattern (two times sync).

```
Sending b'::' to stp.
```

```
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
```

```
STP: 2nd data sync (3a) received => keep state
```

Success Return value list if processing data_sync twice is correct (Content [[]] and Type is <class 'list'>).

```
Result (Return value list if processing data_sync twice): [ [ ] ] (<class 'list'>)
```

```
Expectation (Return value list if processing data_sync twice): result = [ [ ] ] (<class  
↪ 'list'>)
```

Success State after processing data_sync twice is correct (Content 1 and Type is <class 'int'>).

```
Result (State after processing data_sync twice): 1 (<class 'int'>)
```

```
Expectation (State after processing data_sync twice): result = 1 (<class 'int'>)
```

B.1.17 Frame processing - Incorrect end pattern

Description

On receiving an incorrect end pattern, STP shall change to state STORE_DATA, in case of a start pattern, to ESCAPE_1, in case of data sync was received twice or back to state IDLE in all other faulty end patterns starting with data sync. A warning shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**.

Info Processing data with an insufficient end pattern.

Sending b':<te:d' to stp.

STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->

↳ STP_STATE_STORE_DATA

STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2

STP: data (64) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE

STP: Chunking "(2): 74 65" from buffer

Success Return value list if processing data_sync and data again after start of frame is correct (Content [[]] and Type is <class 'list'>).

Result (Return value list if processing data_sync and data again after start of frame): [[]] (<class 'list'>)

Expectation (Return value list if processing data_sync and data again after start of frame):

↳ result = [[]] (<class 'list'>)

Success State after processing data_sync and data again after start of frame is correct (Content 0 and Type is <class 'int'>).

Result (State after processing data_sync and data again after start of frame): 0 (<class 'int'>)

Expectation (State after processing data_sync and data again after start of frame): result = 0 (<class 'int'>)

Success Buffer size after processing data with insufficient end pattern is correct (Content 0 and Type is <class 'int'>).

Result (Buffer size after processing data with insufficient end pattern): 0 (<class 'int'>)

Expectation (Buffer size after processing data with insufficient end pattern): result = 0 (<class 'int'>)

Info Processing data with an insufficient end pattern (start pattern instead of end pattern).

Sending b':<te:<' to stp.

STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
 ↪ STP_STATE_STORE_DATA

STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_2 ->
 ↪ STP_STATE_STORE_DATA

STP: Chunking "(2): 74 65" from buffer

Success Return value list if processing 2nd start of frame is correct (Content [[]] and Type is <class 'list'>).

Result (Return value list if processing 2nd start of frame): [[]] (<class 'list'>)

Expectation (Return value list if processing 2nd start of frame): result = [[]] (<class
 ↪ 'list'>)

Success State after processing 2nd start of frame is correct (Content 3 and Type is <class 'int'>).

Result (State after processing 2nd start of frame): 3 (<class 'int'>)

Expectation (State after processing 2nd start of frame): result = 3 (<class 'int'>)

Success Buffer size after processing 2nd start of frame is correct (Content 0 and Type is <class 'int'>).

Result (Buffer size after processing 2nd start of frame): 0 (<class 'int'>)

Expectation (Buffer size after processing 2nd start of frame): result = 0 (<class 'int'>)

Info Processing data with an insufficient end pattern (two times sync instead of end pattern).

Sending b':<te:;' to stp.

STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1

STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
 ↪ STP_STATE_STORE_DATA

STP: data sync (3a) received => changing state STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2

STP: second data sync (3a) received => changing state STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1

STP: Chunking "(2): 74 65" from buffer

Success Return value list if processing data_sync twice after start of frame is correct (Content [[]] and Type is <class 'list'>).

Result (Return value list if processing data_sync twice after start of frame): [[]]
 ↪ (<class 'list'>)

Expectation (Return value list if processing data_sync twice after start of frame): result =
 ↪ [[]] (<class 'list'>)

Success State after processing data_sync twice after start of frame is correct (Content 1 and Type is <class 'int'>).

Result (State after processing data_sync twice after start of frame): 1 (<class 'int'>)

Expectation (State after processing data_sync twice after start of frame): result = 1 (<class
 ↪ 'int'>)

B.1.18 Frame processing - After state corruption

Description

The state of STP shall be set to IDLE, after an unknown state was recognised. The currently processed data shall be processed again. An error shall be given to the logger.

Reason for the implementation

Robustness against wrong or corrupted data.

Testresult

This test was passed with the state: **Success**.

Info Corrupting stp state and processing data.

```

Sending b':<te' to stp.
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↳ STP_STATE_STORE_DATA
Setting state of stp to 255.
Sending b':<te' to stp.
STP: unknown state (255) => adding value (3a) back to data again and changing state ->
↳ STP_STATE_IDLE
STP: Chunking "(2): 74 65" from buffer
STP: data sync (3a) received => changing state STP_STATE_IDLE -> STP_STATE_ESCAPE_1
STP: start pattern (3a 3c) received => changing state STP_STATE_ESCAPE_1 ->
↳ STP_STATE_STORE_DATA
    
```

Success Return value list if processing start of a frame after state had been corrupted is correct (Content [[]] and Type is <class 'list'>).

```

Result (Return value list if processing start of a frame after state had been corrupted): [ [
↳ ] ] (<class 'list'>)
Expectation (Return value list if processing start of a frame after state had been
↳ corrupted): result = [ [ ] ] (<class 'list'>)
    
```

Success State after processing start of a frame after state had been corrupted is correct (Content 3 and Type is <class 'int'>).

```

Result (State after processing start of a frame after state had been corrupted): 3 (<class
↳ 'int'>)
Expectation (State after processing start of a frame after state had been corrupted): result
↳ = 3 (<class 'int'>)
    
```

Success Buffer size after corrupting stp state is correct (Content 2 and Type is <class 'int'>).

```

Result (Buffer size after corrupting stp state): 2 (<class 'int'>)
Expectation (Buffer size after corrupting stp state): result = 2 (<class 'int'>)
    
```

C Test-Coverage

C.1 stringtools

The line coverage for stringtools was 100.0%

The branch coverage for stringtools was 97.1%

C.1.1 stringtools.__init__.py

The line coverage for stringtools.__init__.py was 100.0%

The branch coverage for stringtools.__init__.py was 97.1%

```

1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #
4 """
5 stringtools (Stringtools)
6 =====
7
8 **Author:**
9
10 * Dirk Alders <sudo-dirk@mount-mockery.de>
11
12 **Description:**
13
14     This Module supports functionality around string operations.
15
16 **Submodules:**
17
18 * :mod:`stringtools.csp`
19 * :mod:`stringtools.stp`
20 * :func:`gzip_compress`
21 * :func:`gzip_extract`
22 * :func:`hexlify`
23
24 **Unittest:**
25
26     See also the :download:`unittest <stringtools/_testresults_/unittest.pdf>` documentation.
27
28 **Module Documentation:**
29
30 """
31
32 from stringtools import stp
33 from stringtools import csp
34 __DEPENDENCIES__ = []
35
36 import gzip
37 import logging
38 import time
39 import sys
40 if sys.version_info < (3, 0):
41     from cStringIO import StringIO
42
43 logger_name = 'STRINGTOOLS'
44 logger = logging.getLogger(logger_name)
45

```

Unittest for stringtools

```
46 __DESCRIPTION__ = """The Module {\\tt %s} is designed to support functionality for strings (e.g.
47     transfer strings via a bytestream, compressing, extracting, ...).
48 For more information read the sphinx documentation.""" % __name__.replace('-', '\\-')
49 """The Module Description"""
50 __INTERPRETER__ = (2, 3)
51 """The Tested Interpreter-Versions"""
52
53 __all__ = ['gzip_compress',
54           'gzip_extract',
55           'hexlify',
56           'csp',
57           'stp']
58
59 def gzip_compress(s, compresslevel=9):
60     """
61     Method to compress a stream of bytes.
62
63     :param str s: The bytestream (string) to be compressed
64     :param int compresslevel: An optional compression level (default is 9)
65     :return: The compressed bytestream
66     :rtype: str
67
68     **Example:**
69
70     .. literalinclude:: ../examples/gzip_compress.py
71
72     Will result to the following output:
73
74     .. literalinclude:: ../examples/gzip_compress.log
75     """
76     rv = None
77     t = time.time()
78     if sys.version_info >= (3, 0):
79         rv = gzip.compress(s, compresslevel)
80     else:
81         buf = StringIO()
82         f = gzip.GzipFile(mode='wb', compresslevel=compresslevel, fileobj=buf)
83         try:
84             f.write(s)
85         finally:
86             f.close()
87             rv = buf.getvalue()
88             buf.close()
89     if rv is not None:
90         logger.debug('GZIP: Finished to compress a string (compression_rate=%0.3f, consumed_time
91     =%0.1fs).', len(rv) / float(len(s)), time.time() - t)
92     return rv
93
94 def gzip_extract(s):
95     """
96     Method to extract data from a compress stream of bytes.
97
98     :param str s: The compressed bytestream (string) to be extracted
99     :return: The extracted data
100     :rtype: str
101
102     **Example:**
103
104     .. literalinclude:: ../examples/gzip_extract.py
105
106     Will result to the following output:
107
108     .. literalinclude:: ../examples/gzip_extract.log
109     """
```

Unittest for stringtools

```
110 t = time.time()
111 rv = None
112 if sys.version_info >= (3, 0):
113     rv = gzip.decompress(s)
114 else:
115     inbuffer = StringIO(s)
116     f = gzip.GzipFile(mode='rb', fileobj=inbuffer)
117     try:
118         rv = f.read()
119     finally:
120         f.close()
121         inbuffer.close()
122 if rv is not None:
123     logger.debug('GZIP: Finished to extract a string (compression_rate=%3f, consumed_time
124     =%1fs).', len(s) / float(len(rv)), time.time() - t)
125     return rv
126
127 def hexlify(s):
128     """Method to hexlify a string.
129
130     :param str s: A string including the bytes to be hexlified.
131     :returns: The hexlified string
132     :rtype: str
133
134     **Example:**
135
136     .. literalinclude:: ../examples/hexlify.py
137
138     Will result to the following output:
139
140     .. literalinclude:: ../examples/hexlify.log
141     """
142     rv = '%d:' % len(s)
143     for byte in s:
144         if sys.version_info >= (3, 0):
145             rv += ' %02x' % byte
146         else:
147             rv += ' %02x' % ord(byte)
148     return rv
149
150
151 def linefeed_filter(s):
152     """Method to change linefeed and carriage return to '\\\\n' and '\\\\r'
153
154     :param str s: A string including carriage return and/ or linefeed.
155     :returns: A string with converted carriage return and/ or linefeed.
156     :rtype: str
157     """
158     if sys.version_info >= (3, 0):
159         return s.replace(b'\r', b'\\r').replace(b'\n', b'\\n')
160     else:
161         return s.replace('\r', '\\r').replace('\n', '\\n')
```

C.1.2 stringtools.csp.py

The line coverage for stringtools.csp.py was 100.0%

The branch coverage for stringtools.csp.py was 97.1%

```

1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #
4 """
5 csp (Carriage-Return seperation protocol)
6 =====
7
8 **Author:**
9
10 * Dirk Alders <sudo-dirk@mount-mockery.de>
11
12 **Description:**
13
14     This module is a submodule of :mod:`stringtools` and creates an frame to transmit and receive
15     messages via an serial interface.
16
17 **Submodules:**
18
19 * :class:`stringtools.csp.csp`
20 * :func:`stringtools.csp.build_frame`
21 """
22 import stringtools
23
24 import logging
25 import sys
26
27 logger_name = 'STRINGTOOLS'
28 logger = logging.getLogger(logger_name)
29
30 DATA_SEPERATOR = b'\n'
31
32
33 class csp(object):
34     """This class extracts messages from an "csp-stream".
35
36     **Example:**
37
38     .. literalinclude:: ../examples/csp.csp.py
39
40     Will result to the following output:
41
42     .. literalinclude:: ../examples/csp.csp.log
43     """
44     LOG_PREFIX = 'CSP:'
45
46     def __init__(self, seperator=DATA_SEPERATOR):
47         self.__buffer__ = b''
48         self.__seperator__ = seperator
49
50     def process(self, data):
51         """
52         This processes a byte out of a "stp-stream".
53
54         :param bytes data: A byte stream
55         :returns: A list of the extracted message(s)
56         :rtype: list
57         """

```

Unittest for stringtools

```
58     if sys.version_info < (3, 0):
59         if type(data) is unicode:
60             raise TypeError
61     #
62     rv = (self.__buffer__ + data).split(self.__separator__)
63     self.__buffer__ = rv.pop()
64     if len(self.__buffer__) != 0:
65         logger.debug('%s Leaving data in buffer (to be processed next time): %s', self.
LOG.PREFIX, stringtools.hexlify(self.__buffer__))
66     for msg in rv:
67         logger.info('%s message identified - %s', self.LOG.PREFIX, stringtools.hexlify(msg))
68     return rv
69
70
71 def build_frame(msg, seperator=DATA.SEPERATOR):
72     """This Method builds an "csp-frame" to be transfered via a stream.
73
74     :param str data: A String (Bytes) to be framed
75     :returns: The "csp-framed" message to be sent
76     :rtype: str
77
78     **Example:**
79
80     .. literalinclude:: ../examples/csp.build_frame.py
81
82     Will result to the following output:
83
84     .. literalinclude:: ../examples/csp.build_frame.log
85     """
86     if seperator in msg:
87         raise ValueError
88     else:
89         return msg + seperator
```

C.1.3 stringtools.stp.py

The line coverage for stringtools.stp.py was 100.0%

The branch coverage for stringtools.stp.py was 97.1%

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #
4 """
5 stp (Serial transfer protocol)
6 =====
7
8 **Author:**
9
10 * Dirk Alders <sudo-dirk@mount-mockery.de>
11
12 **Description:**
13
14     This module is a submodule of :mod:`stringtools` and creates an serial frame to transmit and
15     receive messages via an serial interface.
16
17 **Submodules:**
18
19 * :class:`stringtools.stp.stp`
20 * :func:`stringtools.stp.build_frame`
21 """
```

Unittest for stringtools

```

22 import stringtools
23
24 import logging
25 import sys
26
27 logger_name = 'STRINGTOOLS'
28 logger = logging.getLogger(logger_name)
29
30 DATA_SYNC = b'\x3a'
31 """The data sync byte"""
32 DATA_CLEAR_BUFFER = b'\x3c'
33 """The clear buffer byte ('\x3a\x3c' -> start of message)"""
34 DATA_VALID_MSG = b'\x3e'
35 """The valid message byte ('\x3a\x3e' -> end of message)"""
36 DATA_STORE_SYNC_VALUE = b'\x3d'
37 """The store sync value byte ('\x3a\x3d' -> '\x3a' inside a message)"""
38
39 STP_STATE_IDLE = 0x00
40 """Idle state definition (default)"""
41 STP_STATE_ESCAPE_1 = 0x01
42 """Escape 1 state definition ('\x3a\x3c' found)"""
43 STP_STATE_ESCAPE_2 = 0x02
44 """Escape 2 state definition ('\x3a' found inside a message)"""
45 STP_STATE_STORE_DATA = 0x03
46 """Store data state definition (start of message found; data will be stored)"""
47
48
49 class stp(object):
50     """This class extracts messages from an "stp-stream".
51
52     **Example:**
53
54     .. literalinclude:: ../examples/stp.stp.py
55
56     Will result to the following output:
57
58     .. literalinclude:: ../examples/stp.stp.log
59     """
60     LOG_PREFIX = 'STP: '
61
62     def __init__(self):
63         self.state = STP_STATE_IDLE
64         self.__buffer__ = b''
65         self.__clear_buffer__()
66
67     def __clear_buffer__(self):
68         if len(self.__buffer__) > 0:
69             logger.warning('%s Chunking "%s" from buffer', self.LOG_PREFIX, stringtools.hexlify(
70                 self.__buffer__))
71             self.__buffer__ = b''
72
73     def process(self, data):
74         """
75         This processes a byte out of a "stp-stream".
76
77         :param bytes data: A byte stream
78         :returns: The extracted message or None, if no message is identified yet
79         :rtype: str
80         """
81         if type(data) is list:
82             raise TypeError
83         if sys.version_info <= (3, 0):
84             if type(data) is unicode:
85                 raise TypeError

```

```

85     #
86     rv = []
87     #
88     while len(data) > 0:
89         if sys.version_info >= (3, 0):
90             b = bytes([data[0]])
91         else:
92             b = data[0]
93         data = data[1:]
94         #
95         if self.state == STP_STATE_IDLE:
96             if b == DATA_SYNC:
97                 self.state = STP_STATE_ESCAPE_1
98                 logger.debug('%s data sync (%02x) received => changing state STP_STATE_IDLE
-> STP_STATE_ESCAPE_1', self.LOG_PREFIX, ord(b))
99             else:
100                 logger.warning('%s no data sync (%02x) received => ignoring byte', self.
LOG_PREFIX, ord(b))
101             elif self.state == STP_STATE_ESCAPE_1:
102                 if b == DATA_CLEAR_BUFFER:
103                     logger.debug('%s start pattern (%02x %02x) received => changing state
STP_STATE_ESCAPE_1 -> STP_STATE_STORE_DATA', self.LOG_PREFIX, ord(DATA_SYNC), ord(b))
104                     self.state = STP_STATE_STORE_DATA
105                     self._clear_buffer_()
106                 elif b != DATA_SYNC:
107                     self.state = STP_STATE_IDLE
108                     logger.warning('%s no start pattern (%02x %02x) received => changing state
STP_STATE_ESCAPE_1 -> STP_STATE_IDLE', self.LOG_PREFIX, ord(DATA_SYNC), ord(b))
109                 else:
110                     logger.warning('%s 2nd data sync (%02x) received => keep state', self.
LOG_PREFIX, ord(b))
111             elif self.state == STP_STATE_STORE_DATA:
112                 if b == DATA_SYNC:
113                     self.state = STP_STATE_ESCAPE_2
114                     logger.debug('%s data sync (%02x) received => changing state
STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2', self.LOG_PREFIX, ord(b))
115                 else:
116                     self._buffer_ += b
117             elif self.state == STP_STATE_ESCAPE_2:
118                 if b == DATA_CLEAR_BUFFER:
119                     logger.warning('%s start pattern (%02x %02x) received => changing state
STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA', self.LOG_PREFIX, ord(DATA_SYNC), ord(b))
120                     self.state = STP_STATE_STORE_DATA
121                     self._clear_buffer_()
122                 elif b == DATA_VALID_MSG:
123                     self.state = STP_STATE_IDLE
124                     logger.debug('%s end pattern (%02x %02x) received => storing message and
changing state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE', self.LOG_PREFIX, ord(DATA_SYNC), ord(b)
)
125                     rv.append(self._buffer_)
126                     self._buffer_ = b''
127                 elif b == DATA_STORE_SYNC_VALUE:
128                     self.state = STP_STATE_STORE_DATA
129                     logger.debug('%s store sync pattern (%02x %02x) received => changing state
STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA', self.LOG_PREFIX, ord(DATA_SYNC), ord(b))
130                     self._buffer_ += DATA_SYNC
131                 elif b == DATA_SYNC:
132                     self.state = STP_STATE_ESCAPE_1
133                     logger.warning('%s second data sync (%02x) received => changing state
STP_STATE_ESCAPE_2 -> STP_STATE_ESCAPE_1', self.LOG_PREFIX, ord(b))
134                     self._clear_buffer_()
135             else:

```

Unittest for stringtools

```
136         self.state = STP_STATE_IDLE
137         logger.warning('%s data (%02x) received => changing state STP_STATE_ESCAPE_2
-> STP_STATE_IDLE', self.LOG_PREFIX, ord(b))
138         self._clear_buffer_()
139     else:
140         logger.error('%s unknown state (%s) => adding value (%02x) back to data again and
changing state -> STP_STATE_IDLE', self.LOG_PREFIX, repr(self.state), ord(b))
141         self.state = STP_STATE_IDLE
142         self._clear_buffer_()
143         data = b + data
144     for msg in rv:
145         logger.info('%s message identified - %s', self.LOG_PREFIX, stringtools.hexlify(msg))
146     return rv
147
148
149 def build_frame(data):
150     """This Method builds an "stp-frame" to be transfered via a stream.
151
152     :param str data: A String (Bytes) to be framed
153     :returns: The "stp-framed" message to be sent
154     :rtype: str
155
156     **Example:**
157
158     .. literalinclude:: ../examples/stp.build_frame.py
159
160     Will result to the following output:
161
162     .. literalinclude:: ../examples/stp.build_frame.log
163     """
164     rv = DATA_SYNC + DATA_CLEAR_BUFFER
165
166     for byte in data:
167         if sys.version_info >= (3, 0):
168             byte = bytes([byte])
169         if byte == DATA_SYNC:
170             rv += DATA_SYNC + DATA_STORE_SYNC_VALUE
171         else:
172             rv += byte
173
174     rv += DATA_SYNC + DATA_VALID_MSG
175     return rv
```