Unittest for `stringtools`

May 15, 2026

Contents

1 Test Information

1.1 Testobject 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.

Information	
Name	stringtools
State	Released
Version	0.4.2
Dependencies	
	Python standards

1.2 Testdata Information

Information	
Version	0.4.2
Testruns	Library test

1.3 Testsystem Information

System Information	
Architecture	64bit
Distribution	Debian GNU/Linux 13 trixie
Hostname	ahorn
Kernel	6.12.86+deb13-amd64 (#1 SMP PREEMPT_DYNAMIC Debian 6.12.86-1 (2026-05-08))
Machine	x86_64
Path	/home/dirk/work/unittest_collection/stringtools/unittest/src
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun: Library test

Number of tests	**21**
Number of successfull tests	**21**
Number of possibly failed tests	**0**
Number of failed tests	**0**
Executionlevel	Full Test (all defined tests)
Time consumption	0.017s

2.2 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
stringtools	100.0%	96.9%
stringtools.__init__.py	100.0%	
stringtools.csp.py	100.0%	
stringtools.stp.py	100.0%	

3 Tested Requirements for testrun: Library test

3.1 Human readable value representations

3.1.1 Physical representation (REQ-0019)

Description

The library stringtools shall have a method physical_repr, transforming a float or integer value to a string with a 1 to 3 digit value followed by the physical prefix for the unit.

Testresult

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (12)
Start-Time:	2026-05-15 18:33:48,214
Finished-Time:	2026-05-15 18:33:48,217
Time-Consumption	0.003s

Testsummary:	
Success	Physical representation for 1.17e-10 is correct (Content '117p' and Type is <class 'str'>).
Success	Physical representation for 5.4e-08 is correct (Content '54n' and Type is <class 'str'>).
Success	Physical representation for 2.53e-05 is correct (Content '25.3u' and Type is <class 'str'>).
Success	Physical representation for 0.1 is correct (Content '100m' and Type is <class 'str'>).
Success	Physical representation for 0 is correct (Content '0' and Type is <class 'str'>).
Success	Physical representation for 1 is correct (Content '1' and Type is <class 'str'>).
Success	Physical representation for 1000 is correct (Content '1k' and Type is <class 'str'>).
Success	Physical representation for 1005001 is correct (Content '1.01M' and Type is <class 'str'>).
Success	Physical representation for 1004000000 is correct (Content '1G' and Type is <class 'str'>).
Success	Physical representation for 1003000000000 is correct (Content '1T' and Type is <class 'str'>).
Success	Physical representation for 10000000000000000 is correct (Content '10P' and Type is <class 'str'>).
Success	Physical representation for 17.17 is correct (Content '17.17' and Type is <class 'str'>).
Success	Physical representation for 117000 is correct (Content '117k' and Type is <class 'str'>).
Success	Physical representation for 117.17 is correct (Content '117.2' and Type is <class 'str'>).

3.1.2 Time representation (REQ-0020)

Description

The library stringtools shall have a method physical_repr, transforming an integer value to a time string like HH:MM:SS.

Testresult

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

Testrun:	

Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (13)
Start-Time:	2026-05-15 18:33:48,217
Finished-Time:	2026-05-15 18:33:48,219
Time-Consumption	0.001s

Testsummary:	
Success	Time representation for 59 is correct (Content '00:59' and Type is <class 'str'>).
Success	Time representation for 60 is correct (Content '01:00' and Type is <class 'str'>).
Success	Time representation for 3599 is correct (Content '59:59' and Type is <class 'str'>).
Success	Time representation for 3600 is correct (Content '01:00:00' and Type is <class 'str'>).
Success	Time representation for 86399 is correct (Content '23:59:59' and Type is <class 'str'>).
Success	Time representation for 86400 is correct (Content '1D' and Type is <class 'str'>).
Success	Time representation for 86459 is correct (Content '1D 00:59' and Type is <class 'str'>).
Success	Time representation for 90000 is correct (Content '1D 01:00:00' and Type is <class 'str'>).

3.1.3 Fraction representation (REQ-0021)

Description

The library `stringtools` shall have a method `frac_repr`, transforming a float or integer value to a fraction string with a limited denominator.

Testresult

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (14)
Start-Time:	2026-05-15 18:33:48,219
Finished-Time:	2026-05-15 18:33:48,220
Time-Consumption	0.001s

Testsummary:	
Success	Fraction representation for 17.4 is correct (Content '87/5' and Type is <class 'str'>).
Success	Fraction representation for 0.25 is correct (Content '1/4' and Type is <class 'str'>).
Success	Fraction representation for 0.1 is correct (Content '1/10' and Type is <class 'str'>).
Success	Fraction representation for 0.01666667 is correct (Content '1/60' and Type is <class 'str'>).

3.2 Stream to Human readable String

3.2.1 Hexadecimal Values (REQ-0001)

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.1.4!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (17)
Start-Time:	2026-05-15 18:33:48,220
Finished-Time:	2026-05-15 18:33:48,220
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 (REQ-0002)

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**. See also full trace in section A.1.5!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (18)
Start-Time:	2026-05-15 18:33:48,221
Finished-Time:	2026-05-15 18:33:48,221
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 (REQ-0005)

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.1.6!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (19)
Start-Time:	2026-05-15 18:33:48,221
Finished-Time:	2026-05-15 18:33:48,221
Time-Consumption	0.000s

Testsummary:	
Info	Checking with test data: b'test/r/n123/r/n'.
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 (REQ-0003)

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.1.7!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (22)
Start-Time:	2026-05-15 18:33:48,221
Finished-Time:	2026-05-15 18:33:48,222
Time-Consumption	0.001s

Testsummary:	
Info	Compressing Streams result in differnt streams with the same input stream. Therefore the test will compare the decompressed data.

Info	Compressing stream: (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff
Info	Extracting stream: (26): 1f 8b 08 00 ec 4a 07 6a 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 ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <class 'bytes'>).

3.3.2 Extract (REQ-0004)

Description
The module stringtools shall have a method extracting a Stream with gzip.

Reason for the implementation
Speed up transfer with low transfer rate.

Fitcriterion
Extracted Stream is equal to the original compressed data.

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (23)
Start-Time:	2026-05-15 18:33:48,222
Finished-Time:	2026-05-15 18:33:48,223
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 ff ff ff ff ff ff ff ff ff ff ff ff ff ff' and Type is <class 'str'>).

3.4 Carriagereturn Seperation Protocol (CSP)

3.4.1 Frame creation (REQ-0006)

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**. See also full trace in section A.1.9!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (27)
Start-Time:	2026-05-15 18:33:48,223
Finished-Time:	2026-05-15 18:33:48,223
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 (REQ-0007)

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
ValueErroro 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.1.10!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (28)
Start-Time:	2026-05-15 18:33:48,223
Finished-Time:	2026-05-15 18:33:48,223
Time-Consumption	0.000s

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

3.4.3 Frame processing (REQ-0008)

Description
The CSP Module shall support a class including a method to process stream snipets 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.1.11!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (29)
Start-Time:	2026-05-15 18:33:48,224
Finished-Time:	2026-05-15 18:33:48,224
Time-Consumption	0.001s

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 (REQ-0009)

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.1.12!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (30)
Start-Time:	2026-05-15 18:33:48,224
Finished-Time:	2026-05-15 18:33:48,225
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)

3.5.1 Frame creation (REQ-0010)

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.1.13!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (34)
Start-Time:	2026-05-15 18:33:48,226
Finished-Time:	2026-05-15 18:33:48,226
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 (REQ-0015)

Description
The frame creation method shall support existance 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.1.14!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (35)
Start-Time:	2026-05-15 18:33:48,226
Finished-Time:	2026-05-15 18:33:48,226
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 (REQ-0011)

Description

The STP Module shall support a class including a method to process stream snipets 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.1.15!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (36)
Start-Time:	2026-05-15 18:33:48,226
Finished-Time:	2026-05-15 18:33:48,227
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 (REQ-0012)

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.1.16!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (37)
Start-Time:	2026-05-15 18:33:48,227
Finished-Time:	2026-05-15 18:33:48,229
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 (REQ-0013)

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

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (38)
Start-Time:	2026-05-15 18:33:48,229
Finished-Time:	2026-05-15 18:33:48,229
Time-Consumption	0.000s

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 (REQ-0014)

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.1.18!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (39)
Start-Time:	2026-05-15 18:33:48,229
Finished-Time:	2026-05-15 18:33:48,230
Time-Consumption	0.000s

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 (REQ-0016)

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.1.19!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (40)
Start-Time:	2026-05-15 18:33:48,230
Finished-Time:	2026-05-15 18:33:48,231
Time-Consumption	0.001s

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 (REQ-0017)

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.1.20!

Testrun:	
Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (41)
Start-Time:	2026-05-15 18:33:48,231
Finished-Time:	2026-05-15 18:33:48,233
Time-Consumption	0.002s

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 (REQ-0018)

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.1.21!

Testrun:	

Caller:	/home/dirk/work/unittest_collection/stringtools/unittest/src/tests/__init__.py (42)
Start-Time:	2026-05-15 18:33:48,233
Finished-Time:	2026-05-15 18:33:48,234
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: Library test

A.1 Tests with status Info (21)

A.1.1 Physical representation (REQ-0019)

Description

The library stringtools shall have a method physical_repr, transforming a float or integer value to a string with a 1 to 3 digit value followed by the physical prefix for the unit.

Testresult

This test was passed with the state: Success.

Success	Physical representation for 1.17e-10 is correct (Content '117p' and Type is <class 'str'>).

```
Result (Physical representation for 1.17e-10): '117p' (<class 'str'>)
Expectation (Physical representation for 1.17e-10): result = '117p' (<class 'str'>)
```

Success	Physical representation for 5.4e-08 is correct (Content '54n' and Type is <class 'str'>).

```
Result (Physical representation for 5.4e-08): '54n' (<class 'str'>)
Expectation (Physical representation for 5.4e-08): result = '54n' (<class 'str'>)
```

Success	Physical representation for 2.53e-05 is correct (Content '25.3u' and Type is <class 'str'>).

```
Result (Physical representation for 2.53e-05): '25.3u' (<class 'str'>)
Expectation (Physical representation for 2.53e-05): result = '25.3u' (<class 'str'>)
```

Success	Physical representation for 0.1 is correct (Content '100m' and Type is <class 'str'>).

```
Result (Physical representation for 0.1): '100m' (<class 'str'>)
Expectation (Physical representation for 0.1): result = '100m' (<class 'str'>)
```

Success	Physical representation for 0 is correct (Content '0' and Type is <class 'str'>).

```
Result (Physical representation for 0): '0' (<class 'str'>)
Expectation (Physical representation for 0): result = '0' (<class 'str'>)
```

Success	Physical representation for 1 is correct (Content '1' and Type is <class 'str'>).

```
Result (Physical representation for 1): '1' (<class 'str'>)
Expectation (Physical representation for 1): result = '1' (<class 'str'>)
```

Success	Physical representation for 1000 is correct (Content '1k' and Type is <class 'str'>).

```
Result (Physical representation for 1000): '1k' (<class 'str'>)
```

```
Expectation (Physical representation for 1000): result = '1k' (<class 'str'>)
```

Success	Physical representation for 1005001 is correct (Content '1.01M' and Type is <class 'str'>).

```
Result (Physical representation for 1005001): '1.01M' (<class 'str'>)
Expectation (Physical representation for 1005001): result = '1.01M' (<class 'str'>)
```

Success	Physical representation for 1004000000 is correct (Content '1G' and Type is <class 'str'>).

```
Result (Physical representation for 1004000000): '1G' (<class 'str'>)
Expectation (Physical representation for 1004000000): result = '1G' (<class 'str'>)
```

Success	Physical representation for 1003000000000 is correct (Content '1T' and Type is <class 'str'>).

```
Result (Physical representation for 1003000000000): '1T' (<class 'str'>)
Expectation (Physical representation for 1003000000000): result = '1T' (<class 'str'>)
```

Success	Physical representation for 10000000000000000 is correct (Content '10P' and Type is <class 'str'>).

```
Result (Physical representation for 10000000000000000): '10P' (<class 'str'>)
Expectation (Physical representation for 10000000000000000): result = '10P' (<class 'str'>)
```

Success	Physical representation for 17.17 is correct (Content '17.17' and Type is <class 'str'>).

```
Result (Physical representation for 17.17): '17.17' (<class 'str'>)
Expectation (Physical representation for 17.17): result = '17.17' (<class 'str'>)
```

Success	Physical representation for 117000 is correct (Content '117k' and Type is <class 'str'>).

```
Result (Physical representation for 117000): '117k' (<class 'str'>)
Expectation (Physical representation for 117000): result = '117k' (<class 'str'>)
```

Success	Physical representation for 117.17 is correct (Content '117.2' and Type is <class 'str'>).

```
Result (Physical representation for 117.17): '117.2' (<class 'str'>)
Expectation (Physical representation for 117.17): result = '117.2' (<class 'str'>)
```

A.1.2 Time representation (REQ-0020)

Description

The library `stringtools` shall have a method `physical_repr`, transforming an integer value to a time string like `HH:MM:SS`.

Testresult

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

Success	Time representation for 59 is correct (Content '00:59' and Type is <class 'str'>).

```
Result (Time representation for 59): '00:59' (<class 'str'>)
Expectation (Time representation for 59): result = '00:59' (<class 'str'>)
```

Success	Time representation for 60 is correct (Content '01:00' and Type is <class 'str'>).

```
Result (Time representation for 60): '01:00' (<class 'str'>)
Expectation (Time representation for 60): result = '01:00' (<class 'str'>)
```

Success	Time representation for 3599 is correct (Content '59:59' and Type is <class 'str'>).

```
Result (Time representation for 3599): '59:59' (<class 'str'>)
Expectation (Time representation for 3599): result = '59:59' (<class 'str'>)
```

Success	Time representation for 3600 is correct (Content '01:00:00' and Type is <class 'str'>).

```
Result (Time representation for 3600): '01:00:00' (<class 'str'>)
Expectation (Time representation for 3600): result = '01:00:00' (<class 'str'>)
```

Success	Time representation for 86399 is correct (Content '23:59:59' and Type is <class 'str'>).

```
Result (Time representation for 86399): '23:59:59' (<class 'str'>)
Expectation (Time representation for 86399): result = '23:59:59' (<class 'str'>)
```

Success	Time representation for 86400 is correct (Content '1D' and Type is <class 'str'>).

```
Result (Time representation for 86400): '1D' (<class 'str'>)
Expectation (Time representation for 86400): result = '1D' (<class 'str'>)
```

Success	Time representation for 86459 is correct (Content '1D 00:59' and Type is <class 'str'>).

```
Result (Time representation for 86459): '1D 00:59' (<class 'str'>)
Expectation (Time representation for 86459): result = '1D 00:59' (<class 'str'>)
```

Success	Time representation for 90000 is correct (Content '1D 01:00:00' and Type is <class 'str'>).

```
Result (Time representation for 90000): '1D 01:00:00' (<class 'str'>)
Expectation (Time representation for 90000): result = '1D 01:00:00' (<class 'str'>)
```

A.1.3 Fraction representation (REQ-0021)

Description

The library `stringtools` shall have a method `frac_repr`, transforming a float or integer value to a fraction string with a limited denominator.

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

Success	Fraction representation for 17.4 is correct (Content '87/5' and Type is <class 'str'>).

```
Result (Fraction representation for 17.4): '87/5' (<class 'str'>)
Expectation (Fraction representation for 17.4): result = '87/5' (<class 'str'>)
```

Success	Fraction representation for 0.25 is correct (Content '1/4' and Type is <class 'str'>).

```
Result (Fraction representation for 0.25): '1/4' (<class 'str'>)
Expectation (Fraction representation for 0.25): result = '1/4' (<class 'str'>)
```

Success	Fraction representation for 0.1 is correct (Content '1/10' and Type is <class 'str'>).

```
Result (Fraction representation for 0.1): '1/10' (<class 'str'>)
Expectation (Fraction representation for 0.1): result = '1/10' (<class 'str'>)
```

Success	Fraction representation for 0.01666667 is correct (Content '1/60' and Type is <class 'str'>).

```
Result (Fraction representation for 0.01666667): '1/60' (<class 'str'>)
Expectation (Fraction representation for 0.01666667): result = '1/60' (<class 'str'>)
```

A.1.4 Hexadecimal Values (REQ-0001)

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
```

A.1.5 Number of Bytes (REQ-0002)

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.1.6 CRLF-Filter (REQ-0005)

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 with test data: b'test/r/n123/r/n'.

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'>)
```

A.1.7 Compress (REQ-0003)

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**.

Info	Compressing Streams result in differnt streams with the same input stream. Therefore the test will compare the decompressed data.

Info	Compressing stream: (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

Info	Extracting stream: (26): 1f 8b 08 00 ec 4a 07 6a 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 ff ff ff ff ff ff ff ff ff ff ff ff ff ff and Type is <class 'bytes'>).

```
Result (Extracted data): (30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff ff ff (<class 'bytes'>)
```

```
Expectation (Extracted data): result = b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
↪  \x00\x00\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff\xff' (<class 'bytes'>)
```

A.1.8 Extract (REQ-0004)

Description
The module stringtools shall have a method extracting a Stream with gzip.

Reason for the implementation
Speed up transfer with low transfer rate.

Fitcriterion
Extracted Stream is equal to the original compressed data.

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

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 ff ff ff ff ff ff ff ff ff ff ff ff ff ff' and Type is <class 'str'>).

```
Result (Extracted data): '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff ff ff ff ff ff
↪  ff ff ff ff ff ff ff ff' (<class 'str'>)
```

```
Expectation (Extracted data): result = '(30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff
↪  ff ff ff ff ff ff ff ff ff ff ff ff ff' (<class 'str'>)
```

A.1.9 Frame creation (REQ-0006)

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'>)
```

A.1.10 Frame creation error (REQ-0007)

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
ValueErroro 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/n'

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'>)
```

A.1.11 Frame processing (REQ-0008)

Description
The CSP Module shall support a class including a method to process stream snipets 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

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'>)
```

A.1.12 Frame processing - Input data type error (REQ-0009)

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'>)
```

A.1.13 Frame creation (REQ-0010)

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'>)
```

A.1.14 Frame creation - Start pattern and end pattern inside a message (REQ-0015)

Description
The frame creation method shall support existance 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'>)
```

A.1.15 Frame processing (REQ-0011)

Description
The STP Module shall support a class including a method to process stream snipets 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:>"

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'>)
```

A.1.16 Frame processing - Input data type error (REQ-0012)

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'>)
```

A.1.17 Frame processing - Start pattern and end pattern inside a message (REQ-0013)

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

Info	Processing testframe: 'b':<testframe for :=<stp:⇒:>"

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'>)
```

A.1.18 Frame processing - Data before the start pattern (REQ-0014)

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:>"

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'>)
```

A.1.19 Frame processing - Incorrect start patterns (REQ-0016)

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.
```

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.
```

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'>)
```

A.1.20 Frame processing - Incorrect end pattern (REQ-0017)

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.
```

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.
```

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.
```

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'>)
```

A.1.21 Frame processing - After state corruption (REQ-0018)

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.
Setting state of stp to 255.
Sending b':<te' to stp.
```

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'>)
```

B Test-Coverage

B.1 stringtools

The line coverage for stringtools was 100.0%
The branch coverage for stringtools was 96.9%

B.1.1 stringtools.__init__.py

The line coverage for stringtools.__init__.py was 100.0%
The branch coverage for stringtools.__init__.py was 96.9%

```
"""
stringtools (Stringtools)
=========================

**Author:**

* Dirk Alders <sudo-dirk@mount-mockery.de>

**Description:**

    This Module supports functionality around string operations.

**Submodules:**

* :mod:`stringtools.csp`
* :mod:`stringtools.stp`
* :func:`gzip_compress`
* :func:`gzip_extract`
* :func:`hexlify`

**Unittest:**

        See also the :download:`unittest <stringtools/_testresults_/unittest.pdf>` documentation.

**Module Documentation:**
```

```
26
27 """
28
29 __VERSION__ = "0.4.2"
30 __DEPENDENCIES__ = ["Python standards"]
31
32 import fractions
33 import gzip
34 import logging
35 import time
36
37 from stringtools import csp, stp
38
39 try:
40     from config import APP_NAME as ROOT_LOGGER_NAME
41 except ImportError:
42     ROOT_LOGGER_NAME = "root"
43 logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)
44
45 __DESCRIPTION__ = """The Module {\\tt %s} is designed to support functionality for strings (e.g.
     transfer strings via a bytestream, compressing, extracting, ...).
46 For more Information read the sphinx documentation.""" % __name__.replace("_", "\\_")
47 """The Module Description"""
48
49 __all__ = ["gzip_compress", "gzip_extract", "hexlify", "csp", "stp"]
50
51
52 def physical_value_repr(value, unit="", divider=1000):
53     prefix = {
54         -4: "p",
55         -3: "n",
56         -2: "u",
57         -1: "m",
58         0: "",
59         1: "k",
60         2: "M",
61         3: "G",
62         4: "T",
63         5: "P",
64     }
65     u = 0
66     while u > -4 and u < 5 and (value >= divider or value < 1.0) and value != 0:
67         if value >= 1:
68             u += 1
69             value /= divider
70         else:
71             u -= 1
72             value *= divider
73     if u == 0:
74         ext = ""
75     else:
76         ext = prefix[u]
77     #
78     if value < 100.0:
79         value = "%.2f" % (value)
80     else:
81         value = "%.1f" % (value)
82     while value.find(".") > -1 and (value.endswith("0") or value.endswith(".")):
83         value = value[:-1]
84     return value + ext + unit
85
86
```

```
87  def time_repr(seconds):
88      days = seconds / (24 * 60 * 60)
89      seconds = seconds % (24 * 60 * 60)
90      if seconds >= 60 * 60:
91          rv = time.strftime("%H:%M:%S", time.gmtime(seconds))
92      else:
93          rv = time.strftime("%M:%S", time.gmtime(seconds))
94      if days >= 1:
95          rv = "%dD %s" % (days, rv)
96      if rv.endswith(" 00:00"):
97          rv = rv[:-6]
98      return rv
99
100
101 def frac_repr(value):
102     f = fractions.Fraction(value).limit_denominator()
103     return "%s/%s" % (f.numerator, f.denominator)
104
105
106 def gzip_compress(s, compresslevel=9):
107     """
108     Method to compress a stream of bytes.
109
110     :param str s: The bytestream (string) to be compressed
111     :param int compresslevel: An optional compressionn level (default is 9)
112     :return: The compressed bytestream
113     :rtype: str
114
115     **Example:**
116
117     .. literalinclude:: stringtools/_examples_/gzip_compress.py
118
119     Will result to the following output:
120
121     .. literalinclude:: stringtools/_examples_/gzip_compress.log
122     """
123     rv = None
124     t = time.time()
125     rv = gzip.compress(s, compresslevel)
126     if rv is not None:
127         logger.debug("GZIP: Finished to compress a string (compression_rate=%.3f, consumed_time
    =%.1fs).", len(rv) / float(len(s)), time.time() - t)
128     return rv
129
130
131 def gzip_extract(s):
132     """
133     Method to extract data from a compress stream of bytes.
134
135     :param str s: The compressed bytestream (string) to be extracted
136     :return: The extracted data
137     :rtype: str
138
139     **Example:**
140
141     .. literalinclude:: stringtools/_examples_/gzip_extract.py
142
143     Will result to the following output:
144
145     .. literalinclude:: stringtools/_examples_/gzip_extract.log
146     """
```

```
147     t = time.time()
148     rv = None
149     rv = gzip.decompress(s)
150     if rv is not None:
151         logger.debug("GZIP: Finished to extract a string (compression_rate=%.3f, consumed_time
        =%.1fs).", len(s) / float(len(rv)), time.time() - t)
152     return rv
153 
154 
155 def hexlify(s):
156     """Method to hexlify a string.
157 
158     :param str s: A string including the bytes to be hexlified.
159     :returns: The hexlified string
160     :rtype: str
161 
162     **Example:**
163 
164     .. literalinclude:: stringtools/_examples_/hexlify.py
165 
166     Will result to the following output:
167 
168     .. literalinclude:: stringtools/_examples_/hexlify.log
169     """
170     rv = "(%d):" % len(s)
171     for byte in s:
172         rv += " %02x" % byte
173     return rv
174 
175 
176 def linefeed_filter(s):
177     """Method to change linefeed and carriage return to '\\\\n' and '\\\\r'
178 
179     :param str s: A string including carriage return and/ or linefeed.
180     :returns: A string with converted carriage return and/ or linefeed.
181     :rtype: str
182     """
183     return s.replace(b"\r", b"\\r").replace(b"\n", b"\\n")
```

B.1.2 stringtools.csp.py

The line coverage for stringtools.csp.py was 100.0%
The branch coverage for stringtools.csp.py was 96.9%

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 #
4 """
5 stringtools.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
       messages via an serial interface.
```

```
15
16 **Submodules:**
17
18 * :class:`stringtools.csp.csp`
19 * :func:`stringtools.csp.build_frame`
20 """
21
22 import stringtools
23
24 import logging
25 import sys
26
27 try:
28     from config import APP_NAME as ROOT_LOGGER_NAME
29 except ImportError:
30     ROOT_LOGGER_NAME = "root"
31 logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)
32
33 DATA_SEPERATOR = b"\n"
34
35
36 class csp(object):
37     """This class extracts messages from an "csp-stream".
38
39     **Example:**
40
41     .. literalinclude:: stringtools/_examples_/csp.csp.py
42
43     Will result to the following output:
44
45     .. literalinclude:: stringtools/_examples_/csp.csp.log
46     """
47
48     LOG_PREFIX = "CSP:"
49
50     def __init__(self, seperator=DATA_SEPERATOR):
51         self.__buffer__ = b""
52         self.__seperator__ = seperator
53
54     def process(self, data):
55         """
56         This processes a byte out of a "stp-stream".
57
58         :param bytes data: A byte stream
59         :returns: A list of the extracted message(s)
60         :rtype: list
61         """
62         rv = (self.__buffer__ + data).split(self.__seperator__)
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:: stringtools/_examples_/csp.build_frame.py
81
82      Will result to the following output:
83
84      .. literalinclude:: stringtools/_examples_/csp.build_frame.log
85      """
86      if seperator in msg:
87          raise ValueError
88      else:
89          return msg + seperator
```

B.1.3 stringtools.stp.py

The line coverage for stringtools.stp.py was 100.0%
The branch coverage for stringtools.stp.py was 96.9%

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  #
4  """
5  stringtools.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
       receive messages via an serial interface.
15
16 **Submodules:**
17
18 * :class:`stringtools.stp.stp`
19 * :func:`stringtools.stp.build_frame`
20 """
21
22 import stringtools
23
24 import logging
25 import sys
26
27 try:
28     from config import APP_NAME as ROOT_LOGGER_NAME
29 except ImportError:
30     ROOT_LOGGER_NAME = "root"
31 logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)
32
```

```
DATA_SYNC = b"\x3a"
"""The data sync byte"""
DATA_CLEAR_BUFFER = b"\x3c"
"""The clear buffer byte ('\\\\x3a\\\\x3c' -> start of message)"""
DATA_VALID_MSG = b"\x3e"
"""The valid message byte ('\\\\x3a\\\\x3e' -> end of message)"""
DATA_STORE_SYNC_VALUE = b"\x3d"
"""The store sync value byte ('\\\\x3a\\\\x3d' -> '\\\\x3a' inside a message)"""

STP_STATE_IDLE = 0x00
"""Idle state definition (default)"""
STP_STATE_ESCAPE_1 = 0x01
"""Escape 1 state definition ('\\\\x3a\\\\x3c' found)"""
STP_STATE_ESCAPE_2 = 0x02
"""Escape 2 state definition ('\\\\x3a' found inside a message)"""
STP_STATE_STORE_DATA = 0x03
"""Store data state definition (start of message found; data will be stored)"""


class stp(object):
    """This class extracts messages from an "stp-stream".

    **Example:**

    .. literalinclude:: stringtools/_examples_/stp.stp.py

    Will result to the following output:

    .. literalinclude:: stringtools/_examples_/stp.stp.log
    """

    LOG_PREFIX = "STP:"

    def __init__(self):
        self.state = STP_STATE_IDLE
        self.__buffer__ = b""
        self.__clear_buffer__()

    def __clear_buffer__(self):
        if len(self.__buffer__) > 0:
            logger.warning('%s Chunking "%s" from buffer', self.LOG_PREFIX, stringtools.hexlify(
    self.__buffer__))
        self.__buffer__ = b""

    def process(self, data):
        """
        This processes a byte out of a "stp-stream".

        :param bytes data: A byte stream
        :returns: The extracted message or None, if no message is identified yet
        :rtype: str
        """
        if type(data) is list:
            raise TypeError
        #
        rv = []
        #
        while len(data) > 0:
            b = bytes([data[0]])
            data = data[1:]
            #
```

```
93              if self.state == STP_STATE_IDLE:
94                  if b == DATA_SYNC:
95                      self.state = STP_STATE_ESCAPE_1
96                      logger.debug("%s data sync (%02x) received => changing state STP_STATE_IDLE
    -> STP_STATE_ESCAPE_1", self.LOG_PREFIX, ord(b))
97                  else:
98                      logger.warning("%s no data sync (%02x) received => ignoring byte", self.
    LOG_PREFIX, ord(b))
99              elif self.state == STP_STATE_ESCAPE_1:
100                 if b == DATA_CLEAR_BUFFER:
101                     logger.debug(
102                         "%s start pattern (%02x %02x) received => changing state
    STP_STATE_ESCAPE_1 -> STP_STATE_STORE_DATA",
103                         self.LOG_PREFIX,
104                         ord(DATA_SYNC),
105                         ord(b),
106                     )
107                     self.state = STP_STATE_STORE_DATA
108                     self.__clear_buffer__()
109                 elif b != DATA_SYNC:
110                     self.state = STP_STATE_IDLE
111                     logger.warning(
112                         "%s no start pattern (%02x %02x) received => changing state
    STP_STATE_ESCAPE_1 -> STP_STATE_IDLE",
113                         self.LOG_PREFIX,
114                         ord(DATA_SYNC),
115                         ord(b),
116                     )
117                 else:
118                     logger.warning("%s 2nd data sync (%02x) received => keep state", self.
    LOG_PREFIX, ord(b))
119             elif self.state == STP_STATE_STORE_DATA:
120                 if b == DATA_SYNC:
121                     self.state = STP_STATE_ESCAPE_2
122                     logger.debug("%s data sync (%02x) received => changing state
    STP_STATE_STORE_DATA -> STP_STATE_ESCAPE_2", self.LOG_PREFIX, ord(b))
123                 else:
124                     self.__buffer__ += b
125             elif self.state == STP_STATE_ESCAPE_2:
126                 if b == DATA_CLEAR_BUFFER:
127                     logger.warning(
128                         "%s start pattern (%02x %02x) received => changing state
    STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA",
129                         self.LOG_PREFIX,
130                         ord(DATA_SYNC),
131                         ord(b),
132                     )
133                     self.state = STP_STATE_STORE_DATA
134                     self.__clear_buffer__()
135                 elif b == DATA_VALID_MSG:
136                     self.state = STP_STATE_IDLE
137                     logger.debug(
138                         "%s end pattern (%02x %02x) received => storing message and changing
    state STP_STATE_ESCAPE_2 -> STP_STATE_IDLE",
139                         self.LOG_PREFIX,
140                         ord(DATA_SYNC),
141                         ord(b),
142                     )
143                     rv.append(self.__buffer__)
144                     self.__buffer__ = b""
145                 elif b == DATA_STORE_SYNC_VALUE:
146                     self.state = STP_STATE_STORE_DATA
147                     logger.debug(
```

```
148                         "%s store sync pattern (%02x %02x) received => changing state
    STP_STATE_ESCAPE_2 -> STP_STATE_STORE_DATA",
149                         self.LOG_PREFIX,
150                         ord(DATA_SYNC),
151                         ord(b),
152                     )
153                     self.__buffer__ += DATA_SYNC
154                 elif b == DATA_SYNC:
155                     self.state = STP_STATE_ESCAPE_1
156                     logger.warning(
157                         "%s second data sync (%02x) received => changing state STP_STATE_ESCAPE_2
     -> STP_STATE_ESCAPE_1", self.LOG_PREFIX, ord(b)
158                     )
159                     self.__clear_buffer__()
160                 else:
161                     self.state = STP_STATE_IDLE
162                     logger.warning("%s data (%02x) received => changing state STP_STATE_ESCAPE_2
    -> STP_STATE_IDLE", self.LOG_PREFIX, ord(b))
163                     self.__clear_buffer__()
164             else:
165                 logger.error(
166                     "%s unknown state (%s) => adding value (%02x) back to data again and changing
     state -> STP_STATE_IDLE",
167                     self.LOG_PREFIX,
168                     repr(self.state),
169                     ord(b),
170                 )
171                 self.state = STP_STATE_IDLE
172                 self.__clear_buffer__()
173                 data = b + data
174         for msg in rv:
175             logger.debug("%s message identified - %s", self.LOG_PREFIX, stringtools.hexlify(msg))
176         return rv
177 
178 
179 def build_frame(data):
180     """This Method builds an "stp-frame" to be transfered via a stream.
181 
182     :param str data: A String (Bytes) to be framed
183     :returns: The "stp-framed" message to be sent
184     :rtype: str
185 
186     **Example:**
187 
188     .. literalinclude:: stringtools/_examples_/stp.build_frame.py
189 
190     Will result to the following output:
191 
192     .. literalinclude:: stringtools/_examples_/stp.build_frame.log
193     """
194     rv = DATA_SYNC + DATA_CLEAR_BUFFER
195 
196     for byte in data:
197         byte = bytes([byte])
198         if byte == DATA_SYNC:
199             rv += DATA_SYNC + DATA_STORE_SYNC_VALUE
200         else:
201             rv += byte
202 
203     rv += DATA_SYNC + DATA_VALID_MSG
204     return rv
```