Unittest for `socket_protocol`

December 25, 2020

Contents

1 Test Information

1.1 Test Candidate Information

The Module socket_protocol is designed to pack and unpack data for serial transportation. For more Information read the sphinx documentation.

Library Information	
Name	socket_protocol
State	Released
Supported Interpreters	python2, python3
Version	1ef858a12d357a31af69d84774037597
Dependencies	
stringtools	3eac28a80770a728e1f521fadb92868d

1.2 Unittest Information

Unittest Information	
Version	7075d2ce05c6d6ff54b3fe52f4e996ab
Testruns with	python 2.7.18 (final), python 3.8.5 (final)

1.3 Test System Information

System Information	
Architecture	64bit
Distribution	Linux Mint 20 ulyana
Hostname	ahorn
Kernel	5.4.0-58-generic (#64-Ubuntu SMP Wed Dec 9 08:16:25 UTC 2020)
Machine	x86_64
Path	/user_data/data/dirk/prj/unittest/socket_protocol/unittest
System	Linux
Username	dirk

2 Statistic

2.1 Test-Statistic for testrun with python 2.7.18 (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	12.599s

2.2 Test-Statistic for testrun with python 3.8.5 (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	12.573s

2.3 Coverage Statistic

Module- or Filename	Line-Coverage	Branch-Coverage
socket_protocol	98.7%	98.7%
socket_protocol.__init__.py	98.7%	

3 Testcases with no corresponding Requirement

3.1 Summary for testrun with python 2.7.18 (final)

3.1.1 socket_protocol.pure_json_protocol: Authentification processed without secret.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (44)
Start-Time:	2020-12-25 15:03:34,394
Finished-Time:	2020-12-25 15:03:34,396
Time-Consumption	0.002s

Testsummary:

Info	Authentification with no secret definition (pure_json_protocol).
Success	Return value of authentification is correct (Content False and Type is <type 'bool'>).

3.1.2 socket_protocol.pure_json_protocol: Authentification required, but not processed/correctly processed.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (43)
Start-Time:	2020-12-25 15:03:32,979
Finished-Time:	2020-12-25 15:03:34,393
Time-Consumption	1.414s

Testsummary:

Info	Authentification with different secrets for request and response instance (pure_json_protocol).
Success	Return value of authentification is correct (Content False and Type is <type 'bool'>).
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Response Status (Authentification required) transfered via pure_json_protocol is correct (Content 2 and Type is <type 'int'>).
Success	Response Data (no data) transfered via pure_json_protocol is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

3.1.3 socket_protocol.pure_json_protocol: Checksum corumpation while sending.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (37)
Start-Time:	2020-12-25 15:03:30,348
Finished-Time:	2020-12-25 15:03:30,856
Time-Consumption	0.508s

Testsummary:

Info	Send data with wrong checksum by pure_json_protocol.
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Callback executed variable is correct (Content False and Type is <type 'bool'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

3.1.4 socket_protocol.pure_json_protocol: Incompatible Callback return value(s).

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (45)
Start-Time:	2020-12-25 15:03:34,396
Finished-Time:	2020-12-25 15:03:34,808
Time-Consumption	0.412s

Testsummary:

Info	Send and received data with incompatible callback (pure_json_protocol).
Success	Exception (TypeError) detection variable is correct (Content True and Type is <type 'bool'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Exception (TypeError) detection variable is correct (Content True and Type is <type 'bool'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

3.1.5 socket_protocol.pure_json_protocol: No Callback at response instance for the request.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (42)
Start-Time:	2020-12-25 15:03:32,269

Finished-Time:	2020-12-25 15:03:32,978
Time-Consumption	0.709s

Testsummary:

Info	Send data, but no callback registered (pure_json_protocol).
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Response Status (Request has no callback. Data buffered.) transfered via pure_json_protocol is correct (Content 1 and Type is <type 'int'>).
Success	Response Data (no data) transfered via pure_json_protocol is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

3.1.6 socket_protocol.pure_json_protocol: Register a second Callback with the same service_id.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (32)
Start-Time:	2020-12-25 15:03:28,422
Finished-Time:	2020-12-25 15:03:29,133
Time-Consumption	0.711s

Testsummary:

Info	Send and received data by pure_json_protocol.
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

3.1.7 socket_protocol.pure_json_protocol: Send and receive check including authentification.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (28)

Start-Time:	2020-12-25 15:03:24,869
Finished-Time:	2020-12-25 15:03:26,284
Time-Consumption	1.415s

Testsummary:

Info	Send and received data by pure_json_protocol.
Success	Return value of authentification is correct (Content True and Type is <type 'bool'>).
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

3.1.8 socket_protocol.pure_json_protocol: Send and receive check.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (27)
Start-Time:	2020-12-25 15:03:24,158
Finished-Time:	2020-12-25 15:03:24,869
Time-Consumption	0.711s

Testsummary:

Info	Send and received data by pure_json_protocol.
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

3.1.9 socket_protocol.pure_json_protocol: Timeout measurement for authentification and send method.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (38)
Start-Time:	2020-12-25 15:03:30,857
Finished-Time:	2020-12-25 15:03:32,268
Time-Consumption	1.411s

Testsummary:

Success	Timeout for authentification is correct (Content 0.2013099193572998 in [0.2 ... 0.22000000000000003] and Type is <type 'float'>).
Success	Timeout for authentification is correct (Content 0.502190113067627 in [0.5 ... 0.55] and Type is <type 'float'>).
Success	Timeout for send method is correct (Content 0.20104289054870605 in [0.2 ... 0.22000000000000003] and Type is <type 'float'>).
Success	Timeout for send method is correct (Content 0.501816987991333 in [0.5 ... 0.55] and Type is <type 'float'>).

3.1.10 socket_protocol.pure_json_protocol: Wildcard Callback registration for data_id.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (31)
Start-Time:	2020-12-25 15:03:27,708
Finished-Time:	2020-12-25 15:03:28,422
Time-Consumption	0.714s

Testsummary:

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for .
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

3.1.11 socket_protocol.pure_json_protocol: Wildcard Callback registration for service_id and data_id.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (29)
Start-Time:	2020-12-25 15:03:26,285
Finished-Time:	2020-12-25 15:03:26,996
Time-Consumption	0.711s

Testsummary:

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for service_id and data_id.
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

3.1.12 socket_protocol.pure_json_protocol: Wildcard Callback registration for service_id.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (30)
Start-Time:	2020-12-25 15:03:26,996
Finished-Time:	2020-12-25 15:03:27,707
Time-Consumption	0.711s

Testsummary:

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for service_id.
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

3.1.13 socket_protocol.struct_json_protocol: Send and receive check (Twice for coverage of buffer initialisation).

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (33)
Start-Time:	2020-12-25 15:03:29,134
Finished-Time:	2020-12-25 15:03:30,347
Time-Consumption	1.214s

Testsummary:

Info	Send and received data by struct_json_protocol.
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Request Status (Okay) transfered via struct_json_protocol is correct (Content 0 and Type is <type 'int'>).
Success	Request Data transfered via struct_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).
Success	Response Status (Operation not permitted) transfered via struct_json_protocol is correct (Content 5 and Type is <type 'int'>).
Success	Response Data transfered via struct_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).
Success	Return Value (request instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

3.1.14 socket_protocol.struct_json_protocol: Send and receive check.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (26)
Start-Time:	2020-12-25 15:03:23,448
Finished-Time:	2020-12-25 15:03:24,157
Time-Consumption	0.709s

Testsummary:

Info	Send and received data by struct_json_protocol.
Success	Return value of send method is correct (Content True and Type is <type 'bool'>).
Success	Request Status (Okay) transfered via struct_json_protocol is correct (Content 0 and Type is <type 'int'>).

Success	Request Data transfered via struct_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).
Success	Response Status (Operation not permitted) transfered via struct_json_protocol is correct (Content 5 and Type is <type 'int'>).
Success	Response Data transfered via struct_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).
Success	Return Value (request instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).
Success	Return Value (response instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

3.1.15 socket_protocol: Client setting the channel name.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (50)
Start-Time:	2020-12-25 15:03:35,121
Finished-Time:	2020-12-25 15:03:35,432
Time-Consumption	0.311s

Testsummary:

Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content u'ut_client_set_channel_name' and Type is <type 'unicode'>).
Success	Channel name for client is correct (Content u'ut_client_set_channel_name' and Type is <type 'unicode'>).

3.1.16 socket_protocol: Server and Client setting different channel names.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (51)
Start-Time:	2020-12-25 15:03:35,432
Finished-Time:	2020-12-25 15:03:35,744
Time-Consumption	0.312s

Testsummary:

Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content u'ut_server_and_client_set_channel_name' and Type is <type 'unicode'>).
Success	Channel name for client is correct (Content u'ut_server_and_client_set_channel_name' and Type is <type 'unicode'>).

3.1.17 socket_protocol: Server and Client setting the same channel name.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (52)
Start-Time:	2020-12-25 15:03:35,745
Finished-Time:	2020-12-25 15:03:36,056
Time-Consumption	0.312s

Testsummary:

Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content u'ut_server_and_client_set_channel_name' and Type is <type 'unicode'>).
Success	Channel name for client is correct (Content u'ut_server_and_client_set_channel_name' and Type is <type 'unicode'>).

3.1.18 socket_protocol: Server setting the channel name.

Testresult

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

Testrun:	python 2.7.18 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (49)
Start-Time:	2020-12-25 15:03:34,809
Finished-Time:	2020-12-25 15:03:35,120
Time-Consumption	0.311s

Testsummary:

Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content u'ut_server_set_channel_name' and Type is <type 'unicode'>).
Success	Channel name for client is correct (Content u'ut_server_set_channel_name' and Type is <type 'unicode'>).

3.2 Summary for testrun with python 3.8.5 (final)

3.2.1 socket_protocol.pure_json_protocol: Authentification processed without secret.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (44)
Start-Time:	2020-12-25 15:03:47,476
Finished-Time:	2020-12-25 15:03:47,477

Time-Consumption	0.002s

Testsummary:

Info	Authentification with no secret definition (pure_json_protocol).
Success	Return value of authentification is correct (Content False and Type is <class 'bool'>).

3.2.2 socket_protocol.pure_json_protocol: Authentification required, but not processed/correctly processed.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (43)
Start-Time:	2020-12-25 15:03:46,063
Finished-Time:	2020-12-25 15:03:47,475
Time-Consumption	1.412s

Testsummary:

Info	Authentification with different secrets for request and response instance (pure_json_protocol).
Success	Return value of authentification is correct (Content False and Type is <class 'bool'>).
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Response Status (Authentification required) transfered via pure_json_protocol is correct (Content 2 and Type is <class 'int'>).
Success	Response Data (no data) transfered via pure_json_protocol is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

3.2.3 socket_protocol.pure_json_protocol: Checksum corumpation while sending.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (37)
Start-Time:	2020-12-25 15:03:43,436
Finished-Time:	2020-12-25 15:03:43,942
Time-Consumption	0.507s

Testsummary:

Info	Send data with wrong checksum by pure_json_protocol.
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Callback executed variable is correct (Content False and Type is <class 'bool'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

3.2.4 socket_protocol.pure_json_protocol: Incompatible Callback return value(s).

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (45)
Start-Time:	2020-12-25 15:03:47,478
Finished-Time:	2020-12-25 15:03:47,889
Time-Consumption	0.411s

Testsummary:

Info	Send and received data with incompatible callback (pure_json_protocol).
Success	Exception (TypeError) detection variable is correct (Content True and Type is <class 'bool'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).
Success	Exception (TypeError) detection variable is correct (Content True and Type is <class 'bool'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

3.2.5 socket_protocol.pure_json_protocol: No Callback at response instance for the request.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (42)
Start-Time:	2020-12-25 15:03:45,354
Finished-Time:	2020-12-25 15:03:46,062
Time-Consumption	0.708s

Testsummary:

Info	Send data, but no callback registered (pure_json_protocol).
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Response Status (Request has no callback. Data buffered.) transfered via pure_json_protocol is correct (Content 1 and Type is <class 'int'>).
Success	Response Data (no data) transfered via pure_json_protocol is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

3.2.6 socket_protocol.pure_json_protocol: Register a second Callback with the same service_id.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (32)
Start-Time:	2020-12-25 15:03:41,513
Finished-Time:	2020-12-25 15:03:42,223
Time-Consumption	0.710s

Testsummary:

Info	Send and received data by pure_json_protocol.
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

3.2.7 socket_protocol.pure_json_protocol: Send and receive check including authentification.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (28)
Start-Time:	2020-12-25 15:03:37,969
Finished-Time:	2020-12-25 15:03:39,383
Time-Consumption	1.414s

Testsummary:

Info	Send and received data by pure_json_protocol.
Success	Return value of authentification is correct (Content True and Type is <class 'bool'>).
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

3.2.8 socket_protocol.pure_json_protocol: Send and receive check.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (27)
Start-Time:	2020-12-25 15:03:37,259
Finished-Time:	2020-12-25 15:03:37,969
Time-Consumption	0.709s

Testsummary:

Info	Send and received data by pure_json_protocol.
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

3.2.9 socket_protocol.pure_json_protocol: Timeout measurement for authentification and send method.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (38)
Start-Time:	2020-12-25 15:03:43,943
Finished-Time:	2020-12-25 15:03:45,353
Time-Consumption	1.410s

Testsummary:

Success	Timeout for authentification is correct (Content 0.20126914978027344 in [0.2 ... 0.22000000000000003] and Type is <class 'float'>).
Success	Timeout for authentification is correct (Content 0.5021231174468994 in [0.5 ... 0.55] and Type is <class 'float'>).
Success	Timeout for send method is correct (Content 0.20092320442199707 in [0.2 ... 0.22000000000000003] and Type is <class 'float'>).
Success	Timeout for send method is correct (Content 0.5018301010131836 in [0.5 ... 0.55] and Type is <class 'float'>).

3.2.10 socket_protocol.pure_json_protocol: Wildcard Callback registration for data_id.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (31)
Start-Time:	2020-12-25 15:03:40,803
Finished-Time:	2020-12-25 15:03:41,512
Time-Consumption	0.709s

Testsummary:

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for .
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

3.2.11 socket_protocol.pure_json_protocol: Wildcard Callback registration for service_id and data_id.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (29)
Start-Time:	2020-12-25 15:03:39,384
Finished-Time:	2020-12-25 15:03:40,093
Time-Consumption	0.709s

Testsummary:

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for service_id and data_id.
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

3.2.12 socket_protocol.pure_json_protocol: Wildcard Callback registration for service_id.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (30)
Start-Time:	2020-12-25 15:03:40,093
Finished-Time:	2020-12-25 15:03:40,803
Time-Consumption	0.709s

Testsummary:

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for service_id.
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).
Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).
Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).
Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).
Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

3.2.13 socket_protocol.struct_json_protocol: Send and receive check (Twice for coverage of buffer initialisation).

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (33)
Start-Time:	2020-12-25 15:03:42,224
Finished-Time:	2020-12-25 15:03:43,435
Time-Consumption	1.212s

Testsummary:

Info	Send and received data by struct_json_protocol.
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Request Status (Okay) transfered via struct_json_protocol is correct (Content 0 and Type is <class 'int'>).
Success	Request Data transfered via struct_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).
Success	Response Status (Operation not permitted) transfered via struct_json_protocol is correct (Content 5 and Type is <class 'int'>).
Success	Response Data transfered via struct_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).
Success	Return Value (request instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

3.2.14 socket_protocol.struct_json_protocol: Send and receive check.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (26)
Start-Time:	2020-12-25 15:03:36,551
Finished-Time:	2020-12-25 15:03:37,259
Time-Consumption	0.708s

Testsummary:

Info	Send and received data by struct_json_protocol.
Success	Return value of send method is correct (Content True and Type is <class 'bool'>).
Success	Request Status (Okay) transfered via struct_json_protocol is correct (Content 0 and Type is <class 'int'>).
Success	Request Data transfered via struct_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).
Success	Response Status (Operation not permitted) transfered via struct_json_protocol is correct (Content 5 and Type is <class 'int'>).
Success	Response Data transfered via struct_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).
Success	Return Value (request instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).
Success	Return Value (response instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

3.2.15 socket_protocol: Client setting the channel name.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (50)
Start-Time:	2020-12-25 15:03:48,200
Finished-Time:	2020-12-25 15:03:48,510
Time-Consumption	0.310s

Testsummary:

Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content 'ut_client_set_channel_name' and Type is <class 'str'>).
Success	Channel name for client is correct (Content 'ut_client_set_channel_name' and Type is <class 'str'>).

3.2.16 socket_protocol: Server and Client setting different channel names.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (51)
Start-Time:	2020-12-25 15:03:48,511
Finished-Time:	2020-12-25 15:03:48,821
Time-Consumption	0.310s

Testsummary:

Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content 'ut_server_and_client_set_channel_name' and Type is <class 'str'>).
Success	Channel name for client is correct (Content 'ut_server_and_client_set_channel_name' and Type is <class 'str'>).

3.2.17 socket_protocol: Server and Client setting the same channel name.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (52)
Start-Time:	2020-12-25 15:03:48,821
Finished-Time:	2020-12-25 15:03:49,132
Time-Consumption	0.310s

Testsummary:

Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content 'ut_server_and_client_set_channel_name' and Type is <class 'str'>).
Success	Channel name for client is correct (Content 'ut_server_and_client_set_channel_name' and Type is <class 'str'>).

3.2.18 socket_protocol: Server setting the channel name.

Testresult

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

Testrun:	python 3.8.5 (final)
Caller:	/user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (49)
Start-Time:	2020-12-25 15:03:47,889
Finished-Time:	2020-12-25 15:03:48,199
Time-Consumption	0.310s

Testsummary:

Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content 'ut_server_set_channel_name' and Type is <class 'str'>).
Success	Channel name for client is correct (Content 'ut_server_set_channel_name' and Type is <class 'str'>).

A Trace for testrun with python 2.7.18 (final)

A.1 Tests with status Info (18)

A.1.1 socket_protocol.struct_json_protocol: Send and receive check.

Testresult

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

Info	Send and received data by struct_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
Send data: (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
Receive data (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
Send data: (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
Receive data (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Request Status (Okay) transfered via struct_json_protocol is correct (Content 0 and Type is <type 'int'>).

```
Result (Request Status (Okay) transfered via struct_json_protocol): 0 (<type 'int'>)
Expectation (Request Status (Okay) transfered via struct_json_protocol): result = 0 (<type
↪  'int'>)
```

Success	Request Data transfered via struct_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).

```
Result (Request Data transfered via struct_json_protocol): { u'test': u'test' } (<type
↪  'dict'>)
```

```
Expectation (Request Data transfered via struct_json_protocol): result = { u'test': u'test' }
↪  (<type 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via struct_json_protocol is correct (Content 5 and Type is <type 'int'>).

```
Result (Response Status (Operation not permitted) transfered via struct_json_protocol): 5
↪  (<type 'int'>)
```

```
Expectation (Response Status (Operation not permitted) transfered via struct_json_protocol):
↪  result = 5 (<type 'int'>)
```

Success	Response Data transfered via struct_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

```
Result (Response Data transfered via struct_json_protocol): [ 1, 3, u's' ] (<type 'list'>)
```

```
Expectation (Response Data transfered via struct_json_protocol): result = [ 1, 3, u's' ]
↪  (<type 'list'>)
```

Success	Return Value (request instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (request instance) for struct_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (request instance) for struct_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (response instance) for struct_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (response instance) for struct_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

A.1.2 socket_protocol.pure_json_protocol: Send and receive check.

Testresult

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

Info	Send and received data by pure_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
Send data: (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 24
Receive data (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 24
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
Send data: (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 85 0b b7 34
Receive data (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 85 0b b7 34
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<type 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<type
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { u'test': u'test' } (<type 'dict'>)
```

```
Expectation (Request Data transfered via pure_json_protocol): result = { u'test': u'test' }
↪  (<type 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<type 'int'>)
```

```
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<type 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, u's' ] (<type 'list'>)
```

```
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, u's' ] (<type
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

A.1.3 socket_protocol.pure_json_protocol: Send and receive check including authentification.

Testresult

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

Info	Send and received data by pure_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Requesting seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 1, data_id: 0, data: "None"
Send data: (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 2c 2d 2e 5d
Receive data (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 2c 2d 2e 5d
socket_protocol (server): RX <- status: 0, service_id: 1, data_id: 0, data: "None"
socket_protocol (server): Executing callback __authentificate_create_seed__ to process
↪  received data
socket_protocol (server): Got seed request, sending seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 2, data_id: 0, data:
↪  "'912a5df14734d4547b824eea2ac78565f9c47e224d504156b990bdb30fc10ecc'"
Send data: (124): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 32 2c 20 22 64 61 74 61 22 3a 20 22 39 31 32 61 35 64 66 31 34 37 33 34 64 34 35
↪  34 37 62 38 32 34 65 65 61 32 61 63 37 38 35 36 35 66 39 63 34 37 65 32 32 34 64 35 30 34
↪  31 35 36 62 39 39 30 62 64 62 33 30 66 63 31 30 65 63 63 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d 1d ec f7 06
Receive data (124): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 32 2c 20 22 64 61 74 61 22 3a 20 22 39 31 32 61 35 64 66 31 34 37 33 34 64 34
↪  35 34 37 62 38 32 34 65 65 61 32 61 63 37 38 35 36 35 66 39 63 34 37 65 32 32 34 64 35 30
↪  34 31 35 36 62 39 39 30 62 64 62 33 30 66 63 31 30 65 63 63 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d 1d ec f7 06
socket_protocol (server): RX <- status: 0, service_id: 2, data_id: 0, data:
↪  "u'912a5df14734d4547b824eea2ac78565f9c47e224d504156b990bdb30fc10ecc'"
socket_protocol (server): Executing callback __authentificate_create_key__ to process
↪  received data
socket_protocol (server): Got seed, sending key for authentification
socket_protocol (server): TX -> status: 0, service_id: 3, data_id: 0, data:
↪  "'c3edbfa339b6a4812ad8aadc6aaf8a6f113d9bf1e7761064ed2a04c31ea94e8bea2caa40bec44bcced56c15
↪  048934cb57a856465d5b0d47d47dbdfa3aec505c0'"
Send data: (188): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 33 2c 20 22 64 61 74 61 22 3a 20 22 63 33 65 64 62 66 61 33 33 39 62 36 61 34 38
↪  31 32 61 64 38 61 61 64 63 36 61 61 66 38 61 36 66 31 31 33 64 39 62 66 31 65 37 37 36 31
↪  30 36 34 65 64 32 61 30 34 63 33 31 65 61 39 34 65 38 62 65 61 32 63 61 61 34 30 62 65 63
↪  34 34 62 63 63 65 64 35 36 63 31 35 30 34 38 39 33 34 63 62 35 37 61 38 35 36 34 36 35 64
↪  35 62 30 64 34 37 64 34 37 64 62 64 66 61 33 61 65 63 35 30 35 63 30 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 30 7d d6 e5 e1 72
Receive data (188): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 33 2c 20 22 64 61 74 61 22 3a 20 22 63 33 65 64 62 66 61 33 33 39 62 36 61 34
↪  38 31 32 61 64 38 61 61 64 63 36 61 61 66 38 61 36 66 31 31 33 64 39 62 66 31 65 37 37 36
↪  31 30 36 34 65 64 32 61 30 34 63 33 31 65 61 39 34 65 38 62 65 61 32 63 61 61 34 30 62 65
↪  63 34 34 62 63 63 65 64 35 36 63 31 35 30 34 38 39 33 34 63 62 35 37 61 38 35 36 34 36 35
↪  64 35 62 30 64 34 37 64 34 37 64 62 64 66 61 33 61 65 63 35 30 35 63 30 22 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 30 7d d6 e5 e1 72
socket_protocol (server): RX <- status: 0, service_id: 3, data_id: 0, data:
    "u'c3edbfa339b6a4812ad8aadc6aaf8a6f113d9bf1e7761064ed2a04c31ea94e8bea2caa40bec44bcced56c1
```

```
Result (Return value of authentification): True (<type 'bool'>)
Expectation (Return value of authentification): result = True (<type 'bool'>)
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<type 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<type
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { u'test': u'test' } (<type 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { u'test': u'test' }
↪  (<type 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<type 'int'>)
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<type 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, u's' ] (<type 'list'>)
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, u's' ] (<type
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

A.1.4 socket_protocol.pure_json_protocol: Wildcard Callback registration for service_id and data_id.

Testresult

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

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for service_id and data_id.

```
socket_protocol (server): Cleaning up receive-buffer
```

```
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

```
socket_protocol (server): Cleaning up receive-buffer
```

```
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

```
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 48879, data: "{u'test':
↪  u'test'}"
```

```
Send data: (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d e8 e0 82 59
```

```
Receive data (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d e8 e0 82 59
```

```
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 48879, data: "{u'test':
↪  u'test'}"
```

```
socket_protocol (server): Executing callback response_data_method to process received data
```

```
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 48879, data: "[1, 3,
↪  u's']"
```

```
Send data: (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d 0e 05 f1 49
```

```
Receive data (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d 0e 05 f1 49
```

```
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3,
↪  u's']"
```

```
socket_protocol (server): Received message has a peculiar status: Operation not permitted
```

```
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<type 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<type
↪  'int'>)
```

Success Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { u'test': u'test' } (<type 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { u'test': u'test' }
↪  (<type 'dict'>)
```

Success Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<type 'int'>)
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<type 'int'>)
```

Success Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, u's' ] (<type 'list'>)
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, u's' ] (<type
↪  'list'>)
```

Success Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 48879) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): result = None (<type 'NoneType'>)
```

Success Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 48879) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<type 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xbeef): result = None (<type 'NoneType'>)
```

A.1.5 socket_protocol.pure_json_protocol: Wildcard Callback registration for service_id.

Testresult

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

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for service_id.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 48879, data: "{u'test':
↪  u'test'}"
Send data: (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d e8 e0 82 59
Receive data (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d e8 e0 82 59
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 48879, data: "{u'test':
↪  u'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 48879, data: "[1, 3,
↪  u's']"
Send data: (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d 0e 05 f1 49
Receive data (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d 0e 05 f1 49
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3,
↪  u's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<type 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<type
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { u'test': u'test' } (<type 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { u'test': u'test' }
↪  (<type 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<type 'int'>)
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<type 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, u's' ] (<type 'list'>)
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, u's' ] (<type
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 48879) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 48879) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<type 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xbeef): result = None (<type 'NoneType'>)
```

A.1.6 socket_protocol.pure_json_protocol: Wildcard Callback registration for data_id.

Testresult

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

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for .

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 48879, data: "{u'test':
↪  u'test'}"
Send data: (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d e8 e0 82 59
Receive data (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d e8 e0 82 59
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 48879, data: "{u'test':
↪  u'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 48879, data: "[1, 3,
↪  u's']"
Send data: (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d 0e 05 f1 49
Receive data (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 7d 0e 05 f1 49
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3,
↪  u's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<type 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<type
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { u'test': u'test' } (<type 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { u'test': u'test' }
↪  (<type 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<type 'int'>)
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<type 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, u's' ] (<type 'list'>)
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, u's' ] (<type
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 48879) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 48879) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<type 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xbeef): result = None (<type 'NoneType'>)
```

A.1.7 socket_protocol.pure_json_protocol: Register a second Callback with the same service_id.

Testresult

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

Info	Send and received data by pure_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
Send data: (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 24
Receive data (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 24
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
socket_protocol (server): Executing callback response_data_method_2 to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
Send data: (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 85 0b b7 34
Receive data (74): 7b 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 2c 20 22 64
↪  61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 85 0b b7 34
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <type 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<type 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<type
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { u'test': u'test' } (<type 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { u'test': u'test' }
↪  (<type 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <type 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<type 'int'>)
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<type 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, u's' ] (<type 'list'>)
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, u's' ] (<type
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

A.1.8 socket_protocol.struct_json_protocol: Send and receive check (Twice for coverage of buffer initialisation).

Testresult

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

Info	Send and received data by struct_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
Send data: (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
Receive data (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
Send data: (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
Receive data (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
Send data: (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
Receive data (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
Send data: (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
Receive data (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3,
↪  u's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Request Status (Okay) transfered via struct_json_protocol is correct (Content 0 and Type is <type 'int'>).

```
Result (Request Status (Okay) transfered via struct_json_protocol): 0 (<type 'int'>)
Expectation (Request Status (Okay) transfered via struct_json_protocol): result = 0 (<type
↪  'int'>)
```

Success	Request Data transfered via struct_json_protocol is correct (Content {u'test': u'test'} and Type is <type 'dict'>).

```
Result (Request Data transfered via struct_json_protocol): { u'test': u'test' } (<type
↪  'dict'>)
Expectation (Request Data transfered via struct_json_protocol): result = { u'test': u'test' }
↪  (<type 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via struct_json_protocol is correct (Content 5 and Type is <type 'int'>).

```
Result (Response Status (Operation not permitted) transfered via struct_json_protocol): 5
↪  (<type 'int'>)
Expectation (Response Status (Operation not permitted) transfered via struct_json_protocol):
↪  result = 5 (<type 'int'>)
```

Success	Response Data transfered via struct_json_protocol is correct (Content [1, 3, u's'] and Type is <type 'list'>).

```
Result (Response Data transfered via struct_json_protocol): [ 1, 3, u's' ] (<type 'list'>)
Expectation (Response Data transfered via struct_json_protocol): result = [ 1, 3, u's' ]
↪  (<type 'list'>)
```

Success	Return Value (request instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for struct_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for struct_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (response instance) for struct_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (response instance) for struct_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

A.1.9 socket_protocol.pure_json_protocol: Checksum corumpation while sending.

Testresult

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

Info	Send data with wrong checksum by pure_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
```

```
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

```
socket_protocol (server): Cleaning up receive-buffer
```

```
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

```
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
```

```
Send data: (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 24
```

```
Receive data (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 25
```

```
socket_protocol (server): Received message has a wrong checksum and will be ignored: (79): 7b
↪  22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64 22 3a 20 31 30 2c
↪  20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 25.
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
```

```
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Callback executed variable is correct (Content False and Type is <type 'bool'>).

```
Result (Callback executed variable): False (<type 'bool'>)
```

```
Expectation (Callback executed variable): result = False (<type 'bool'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

A.1.10 socket_protocol.pure_json_protocol: Timeout measurement for authentification and send method.

Testresult

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

Success	Timeout for authentification is correct (Content 0.2013099193572998 in [0.2 ... 0.22000000000000003] and Type is <type 'float'>).

```
socket_protocol (server): Cleaning up receive-buffer
```

```
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

```
socket_protocol (server): Cleaning up receive-buffer
```

```
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

```
socket_protocol (server): Requesting seed for authentification
```

```
socket_protocol (server): TX -> status: 0, service_id: 1, data_id: 0, data: "None"
```

```
Send data: (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 2c 2d 2e 5d
```

```
Result (Timeout for authentification): 0.2013099193572998 (<type 'float'>)
```

```
Expectation (Timeout for authentification): 0.2 <= result <= 0.22000000000000003
```

Success	Timeout for authentification is correct (Content 0.502190113067627 in [0.5 ... 0.55] and Type is <type 'float'>).

```
socket_protocol (server): Requesting seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 1, data_id: 0, data: "None"
Send data: (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 2c 2d 2e 5d
Result (Timeout for authentification): 0.502190113067627 (<type 'float'>)
Expectation (Timeout for authentification): 0.5 <= result <= 0.55
```

Success	Timeout for send method is correct (Content 0.20104289054870605 in [0.2 ... 0.22000000000000003] and Type is <type 'float'>).

```
socket_protocol (server): TIMEOUT (0.2s): Requested data (service_id: 30; data_id: 0) not in
↪  buffer.
Result (Timeout for send method): 0.20104289054870605 (<type 'float'>)
Expectation (Timeout for send method): 0.2 <= result <= 0.22000000000000003
```

Success	Timeout for send method is correct (Content 0.501816987991333 in [0.5 ... 0.55] and Type is <type 'float'>).

```
socket_protocol (server): TIMEOUT (0.5s): Requested data (service_id: 30; data_id: 0) not in
↪  buffer.
Result (Timeout for send method): 0.501816987991333 (<type 'float'>)
Expectation (Timeout for send method): 0.5 <= result <= 0.55
```

A.1.11 socket_protocol.pure_json_protocol: No Callback at response instance for the request.

Testresult

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

Info	Send data, but no callback registered (pure_json_protocol).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
Send data: (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 24
Receive data (79): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22
↪  7d 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 7d 63 ee c4 24
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test':
↪  u'test'}"
socket_protocol (server): Received message with no registered callback. Sending negative
↪  response.
socket_protocol (server): TX -> status: 1, service_id: 11, data_id: 45054, data: "None"
Send data: (67): 7b 22 73 74 61 74 75 73 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 34 35 30 35 34 7d b1 70 10 64
Receive data (67): 7b 22 73 74 61 74 75 73 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 34 35 30 35 34 7d b1 70 10 64
socket_protocol (server): RX <- status: 1, service_id: 11, data_id: 45054, data: "None"
socket_protocol (server): Received message has a peculiar status: Request has no callback.
↪  Data buffered.
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Response Status (Request has no callback. Data buffered.) transfered via pure_json_protocol is correct (Content 1 and Type is <type 'int'>).

```
Result (Response Status (Request has no callback. Data buffered.) transfered via
↪  pure_json_protocol): 1 (<type 'int'>)
Expectation (Response Status (Request has no callback. Data buffered.) transfered via
↪  pure_json_protocol): result = 1 (<type 'int'>)
```

Success	Response Data (no data) transfered via pure_json_protocol is correct (Content None and Type is <type 'NoneType'>).

```
Result (Response Data (no data) transfered via pure_json_protocol): None (<type 'NoneType'>)
```

```
Expectation (Response Data (no data) transfered via pure_json_protocol): result = None (<type
↪  'NoneType'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
```

```
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

A.1.12 socket_protocol.pure_json_protocol: Authentification required, but not processed/correctly processed.

Testresult

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

Info	Authentification with different secrets for request and response instance (pure_json_protocol).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Requesting seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 1, data_id: 0, data: "None"
Send data: (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 2c 2d 2e 5d
Receive data (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 2c 2d 2e 5d
socket_protocol (server): RX <- status: 0, service_id: 1, data_id: 0, data: "None"
socket_protocol (server): Executing callback __authentificate_create_seed__ to process
↪  received data
socket_protocol (server): Got seed request, sending seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 2, data_id: 0, data:
↪  "'c212ab22451532ff9781a3147283982d1a3d1776430fb3743fd31bd934162933'"
Send data: (124): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 32 2c 20 22 64 61 74 61 22 3a 20 22 63 32 31 32 61 62 32 32 34 35 31 35 33 32 66
↪  66 39 37 38 31 61 33 31 34 37 32 38 33 39 38 32 64 31 61 33 64 31 37 37 36 34 33 30 66 62
↪  33 37 34 33 66 64 33 31 62 64 39 33 34 31 36 32 39 33 33 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d 68 96 22 b0
Receive data (124): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 32 2c 20 22 64 61 74 61 22 3a 20 22 63 32 31 32 61 62 32 32 34 35 31 35 33 32
↪  66 66 39 37 38 31 61 33 31 34 37 32 38 33 39 38 32 64 31 61 33 64 31 37 37 36 34 33 30 66
↪  62 33 37 34 33 66 64 33 31 62 64 39 33 34 31 36 32 39 33 33 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d 68 96 22 b0
socket_protocol (server): RX <- status: 0, service_id: 2, data_id: 0, data:
↪  "u'c212ab22451532ff9781a3147283982d1a3d1776430fb3743fd31bd934162933'"
socket_protocol (server): Executing callback __authentificate_create_key__ to process
↪  received data
socket_protocol (server): Got seed, sending key for authentification
socket_protocol (server): TX -> status: 0, service_id: 3, data_id: 0, data:
↪  "'201d6224fcd5ca1c5c10f23d5ca000620b0953e3290f2b3eba5a0f67cfef4126e9b5cc8b8e5b7040b29bd79
↪  507e8071d0d2687146c78fa1a0b7625e6ecbfe6f2'"
Send data: (188): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 33 2c 20 22 64 61 74 61 22 3a 20 22 32 30 31 64 36 32 32 34 66 63 64 35 63 61 31
↪  63 35 63 31 30 66 32 33 64 35 63 61 30 30 30 36 32 30 62 30 39 35 33 65 33 32 39 30 66 32
↪  62 33 65 62 61 35 61 30 66 36 37 63 66 65 66 34 31 32 36 65 39 62 35 63 63 38 62 38 65 35
↪  62 37 30 34 30 62 32 39 62 64 37 39 35 30 37 65 38 30 37 31 64 30 64 32 36 38 37 31 34 36
↪  63 37 38 66 61 31 61 30 62 37 36 32 35 65 36 65 63 62 66 65 36 66 32 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 30 7d 5f 83 0f d9
Receive data (188): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 33 2c 20 22 64 61 74 61 22 3a 20 22 32 30 31 64 36 32 32 34 66 63 64 35 63 61
↪  31 63 35 63 31 30 66 32 33 64 35 63 61 30 30 30 36 32 30 62 30 39 35 33 65 33 32 39 30 66
↪  32 62 33 65 62 61 35 61 30 66 36 37 63 66 65 66 34 31 32 36 65 39 62 35 63 63 38 62 38 65
↪  35 62 37 30 34 30 62 32 39 62 64 37 39 35 30 37 65 38 30 37 31 64 30 64 32 36 38 37 31 34
↪  36 63 37 38 66 61 31 61 30 62 37 36 32 35 65 36 65 63 62 66 65 36 66 32 22 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 30 7d 5f 83 0f d9
socket_protocol (server): RX <- status: 0, service_id: 3, data_id: 0, data:
    "u'201d6224fcd5ca1c5c10f23d5ca000620b0953e3290f2b3eba5a0f67cfef4126e9b5cc8b8e5b7040b29bd7
```

```
Result (Return value of authentification): False (<type 'bool'>)
Expectation (Return value of authentification): result = False (<type 'bool'>)
```

Success	Return value of send method is correct (Content True and Type is <type 'bool'>).

```
Result (Return value of send method): True (<type 'bool'>)
Expectation (Return value of send method): result = True (<type 'bool'>)
```

Success	Response Status (Authentification required) transfered via pure_json_protocol is correct (Content 2 and Type is <type 'int'>).

```
Result (Response Status (Authentification required) transfered via pure_json_protocol): 2
↪  (<type 'int'>)
Expectation (Response Status (Authentification required) transfered via pure_json_protocol):
↪  result = 2 (<type 'int'>)
```

Success	Response Data (no data) transfered via pure_json_protocol is correct (Content None and Type is <type 'NoneType'>).

```
Result (Response Data (no data) transfered via pure_json_protocol): None (<type 'NoneType'>)
Expectation (Response Data (no data) transfered via pure_json_protocol): result = None (<type
↪  'NoneType'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

A.1.13 socket_protocol.pure_json_protocol: Authentification processed without secret.

Testresult

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

Info	Authentification with no secret definition (pure_json_protocol).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

Success	Return value of authentification is correct (Content False and Type is <type 'bool'>).

```
Result (Return value of authentification): False (<type 'bool'>)
Expectation (Return value of authentification): result = False (<type 'bool'>)
```

A.1.14 socket_protocol.pure_json_protocol: Incompatible Callback return value(s).

Testresult

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

Info	Send and received data with incompatible callback (pure_json_protocol).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "None"
Send data: (67): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 34 35 30 35 34 7d fc 3e bd 5f
Receive data (67): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 34 35 30 35 34 7d fc 3e bd 5f
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "None"
socket_protocol (server): Executing callback response_data_method_fail to process received
↪  data
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 48879, data: "None"
Send data: (67): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 34 38 38 37 39 7d 77 30 fb 22
Receive data (67): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 30 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 34 38 38 37 39 7d 77 30 fb 22
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 48879, data: "None"
socket_protocol (server): Received message with no registered callback. Sending negative
↪  response.
socket_protocol (server): TX -> status: 1, service_id: 11, data_id: 48879, data: "None"
Send data: (67): 7b 22 73 74 61 74 75 73 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 34 38 38 37 39 7d 3a 7e 56 19
Receive data (67): 7b 22 73 74 61 74 75 73 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 31 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 34 38 38 37 39 7d 3a 7e 56 19
socket_protocol (server): RX <- status: 1, service_id: 11, data_id: 48879, data: "None"
socket_protocol (server): Received message has a peculiar status: Request has no callback.
↪  Data buffered.
socket_protocol (server): Executing callback response_data_method_fail to process received
↪  data
```

Success	Exception (TypeError) detection variable is correct (Content True and Type is <type 'bool'>).

```
Result (Exception (TypeError) detection variable): True (<type 'bool'>)
Expectation (Exception (TypeError) detection variable): result = True (<type 'bool'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<type 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<type 'NoneType'>)
```

Success	Exception (TypeError) detection variable is correct (Content True and Type is <type 'bool'>).

```
Result (Exception (TypeError) detection variable): True (<type 'bool'>)
Expectation (Exception (TypeError) detection variable): result = True (<type 'bool'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 48879) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<type 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): result = None (<type 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <type 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 48879) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<type 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xbeef): result = None (<type 'NoneType'>)
```

A.1.15 socket_protocol: Server setting the channel name.

Testresult

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

Info	Initiating communication including channel_name exchange.

```
ut_server_set_channel_name (server): Cleaning up receive-buffer
ut_server_set_channel_name (server): Resetting authentification state to
↪  AUTH_STATE_UNKNOWN_CLIENT
Overwriting existing callback '__channel_name_response__' for service_id (6) and data_id (0)
↪  to 'ut_response_callback'!
socket_protocol (client): Cleaning up receive-buffer
socket_protocol (client): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
ut_server_set_channel_name (server): Cleaning up receive-buffer
ut_server_set_channel_name (server): TX -> status: 0, service_id: 5, data_id: 0, data:
↪  "u'ut_server_set_channel_name'"
Send data: (86): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 35 2c 20 22 64 61 74 61 22 3a 20 22 75 74 5f 73 65 72 76 65 72 5f 73 65 74 5f 63
↪  68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 30 7d 81 c2 44
↪  8c
socket_protocol (client): Cleaning up receive-buffer
Receive data (86): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 35 2c 20 22 64 61 74 61 22 3a 20 22 75 74 5f 73 65 72 76 65 72 5f 73 65 74 5f 63
↪  68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 30 7d 81 c2 44
↪  8c
socket_protocol (client): RX <- status: 0, service_id: 5, data_id: 0, data:
↪  "u'ut_server_set_channel_name'"
socket_protocol (client): Executing callback __channel_name_request__ to process received data
ut_server_set_channel_name (client): channel name is now u'ut_server_set_channel_name'
ut_server_set_channel_name (client): TX -> status: 0, service_id: 6, data_id: 0, data: "None"
Send data: (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 99 0f 87 65
Receive data (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 99 0f 87 65
ut_server_set_channel_name (server): RX <- status: 0, service_id: 6, data_id: 0, data: "None"
ut_server_set_channel_name (server): Executing callback ut_response_callback to process
↪  received data
```

Success Channel name for server is correct (Content u'ut_server_set_channel_name' and Type is <type 'unicode'>).

```
Result (Channel name for server): u'ut_server_set_channel_name' (<type 'unicode'>)
Expectation (Channel name for server): result = u'ut_server_set_channel_name' (<type
↪  'unicode'>)
```

Success Channel name for client is correct (Content u'ut_server_set_channel_name' and Type is <type 'unicode'>).

```
Result (Channel name for client): u'ut_server_set_channel_name' (<type 'unicode'>)
Expectation (Channel name for client): result = u'ut_server_set_channel_name' (<type
↪  'unicode'>)
```

A.1.16 socket_protocol: Client setting the channel name.

Testresult

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

Info	Initiating communication including channel_name exchange.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
Overwriting existing callback '__channel_name_response__' for service_id (6) and data_id (0)
↪  to 'ut_response_callback'!
ut_client_set_channel_name (client): Cleaning up receive-buffer
ut_client_set_channel_name (client): Resetting authentification state to
↪  AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): TX -> status: 0, service_id: 5, data_id: 0, data: "None"
Send data: (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 35 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d dd ae a2 7d
ut_client_set_channel_name (client): Cleaning up receive-buffer
Receive data (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 35 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d dd ae a2 7d
ut_client_set_channel_name (client): RX <- status: 0, service_id: 5, data_id: 0, data: "None"
ut_client_set_channel_name (client): Executing callback __channel_name_request__ to process
↪  received data
ut_client_set_channel_name (client): TX -> status: 0, service_id: 6, data_id: 0, data:
↪  "u'ut_client_set_channel_name'"
Send data: (86): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 22 75 74 5f 63 6c 69 65 6e 74 5f 73 65 74 5f 63
↪  68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 30 7d 71 b2 b8
↪  9d
Receive data (86): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 22 75 74 5f 63 6c 69 65 6e 74 5f 73 65 74 5f 63
↪  68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 30 7d 71 b2 b8
↪  9d
socket_protocol (server): RX <- status: 0, service_id: 6, data_id: 0, data:
↪  "u'ut_client_set_channel_name'"
socket_protocol (server): Executing callback ut_response_callback to process received data
ut_client_set_channel_name (server): channel name is now u'ut_client_set_channel_name'
```

Success	Channel name for server is correct (Content u'ut_client_set_channel_name' and Type is <type 'unicode'>).

```
Result (Channel name for server): u'ut_client_set_channel_name' (<type 'unicode'>)
```

```
Expectation (Channel name for server): result = u'ut_client_set_channel_name' (<type
↪  'unicode'>)
```

Success	Channel name for client is correct (Content u'ut_client_set_channel_name' and Type is <type 'unicode'>).

```
Result (Channel name for client): u'ut_client_set_channel_name' (<type 'unicode'>)
```

```
Expectation (Channel name for client): result = u'ut_client_set_channel_name' (<type
↪  'unicode'>)
```

A.1.17 socket_protocol: Server and Client setting different channel names.

Testresult

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

Info	Initiating communication including channel_name exchange.

```
ut_server_and_client_set_channel_name (server): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (server): Resetting authentification state to
↪ AUTH_STATE_UNKNOWN_CLIENT
Overwriting existing callback '__channel_name_response__' for service_id (6) and data_id (0)
↪ to 'ut_response_callback'!
foo (client): Cleaning up receive-buffer
foo (client): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
ut_server_and_client_set_channel_name (server): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (server): TX -> status: 0, service_id: 5, data_id: 0,
↪ data: "u'ut_server_and_client_set_channel_name'"
Send data: (97): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪ 22 3a 20 35 2c 20 22 64 61 74 61 22 3a 20 22 75 74 5f 73 65 72 76 65 72 5f 61 6e 64 5f 63
↪ 6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 2c 20 22 64 61 74 61
↪ 5f 69 64 22 3a 20 30 7d c6 3d f9 62
foo (client): Cleaning up receive-buffer
Receive data (97): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪ 22 3a 20 35 2c 20 22 64 61 74 61 22 3a 20 22 75 74 5f 73 65 72 76 65 72 5f 61 6e 64 5f 63
↪ 6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 2c 20 22 64 61 74 61
↪ 5f 69 64 22 3a 20 30 7d c6 3d f9 62
foo (client): RX <- status: 0, service_id: 5, data_id: 0, data:
↪ "u'ut_server_and_client_set_channel_name'"
foo (client): Executing callback __channel_name_request__ to process received data
ut_server_and_client_set_channel_name (client): overwriting user defined channel name from
↪ 'foo' to u'ut_server_and_client_set_channel_name'
ut_server_and_client_set_channel_name (client): TX -> status: 0, service_id: 6, data_id: 0,
↪ data: "None"
Send data: (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪ 22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪ 20 30 7d 99 0f 87 65
Receive data (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪ 22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪ 20 30 7d 99 0f 87 65
ut_server_and_client_set_channel_name (server): RX <- status: 0, service_id: 6, data_id: 0,
↪ data: "None"
ut_server_and_client_set_channel_name (server): Executing callback ut_response_callback to
↪ process received data
```

Success	Channel name for server is correct (Content u'ut_server_and_client_set_channel_name' and Type is <type 'unicode'>).

```
Result (Channel name for server): u'ut_server_and_client_set_channel_name' (<type 'unicode'>)
Expectation (Channel name for server): result = u'ut_server_and_client_set_channel_name'
↪ (<type 'unicode'>)
```

Success	Channel name for client is correct (Content u'ut_server_and_client_set_channel_name' and Type is <type 'unicode'>).

```
Result (Channel name for client): u'ut_server_and_client_set_channel_name' (<type 'unicode'>)
```

```
Expectation (Channel name for client): result = u'ut_server_and_client_set_channel_name'
↪  (<type 'unicode'>)
```

A.1.18 socket_protocol: Server and Client setting the same channel name.

Testresult

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

Info	Initiating communication including channel_name exchange.

```
ut_server_and_client_set_channel_name (server): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (server): Resetting authentification state to
↪  AUTH_STATE_UNKNOWN_CLIENT
Overwriting existing callback '__channel_name_response__' for service_id (6) and data_id (0)
↪  to 'ut_response_callback'!
ut_server_and_client_set_channel_name (client): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (client): Resetting authentification state to
↪  AUTH_STATE_UNKNOWN_CLIENT
ut_server_and_client_set_channel_name (server): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (server): TX -> status: 0, service_id: 5, data_id: 0,
↪  data: "u'ut_server_and_client_set_channel_name'"
Send data: (97): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 35 2c 20 22 64 61 74 61 22 3a 20 22 75 74 5f 73 65 72 76 65 72 5f 61 6e 64 5f 63
↪  6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 2c 20 22 64 61 74 61
↪  5f 69 64 22 3a 20 30 7d c6 3d f9 62
ut_server_and_client_set_channel_name (client): Cleaning up receive-buffer
Receive data (97): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 35 2c 20 22 64 61 74 61 22 3a 20 22 75 74 5f 73 65 72 76 65 72 5f 61 6e 64 5f 63
↪  6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 2c 20 22 64 61 74 61
↪  5f 69 64 22 3a 20 30 7d c6 3d f9 62
ut_server_and_client_set_channel_name (client): RX <- status: 0, service_id: 5, data_id: 0,
↪  data: "u'ut_server_and_client_set_channel_name'"
ut_server_and_client_set_channel_name (client): Executing callback __channel_name_request__
↪  to process received data
ut_server_and_client_set_channel_name (client): TX -> status: 0, service_id: 6, data_id: 0,
↪  data: "None"
Send data: (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 99 0f 87 65
Receive data (62): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 6e 75 6c 6c 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 30 7d 99 0f 87 65
ut_server_and_client_set_channel_name (server): RX <- status: 0, service_id: 6, data_id: 0,
↪  data: "None"
ut_server_and_client_set_channel_name (server): Executing callback ut_response_callback to
↪  process received data
```

Success	Channel name for server is correct (Content u'ut_server_and_client_set_channel_name' and Type is <type 'unicode'>).

```
Result (Channel name for server): u'ut_server_and_client_set_channel_name' (<type 'unicode'>)
Expectation (Channel name for server): result = u'ut_server_and_client_set_channel_name'
↪  (<type 'unicode'>)
```

Success	Channel name for client is correct (Content u'ut_server_and_client_set_channel_name' and Type is <type 'unicode'>).

```
Result (Channel name for client): u'ut_server_and_client_set_channel_name' (<type 'unicode'>)
Expectation (Channel name for client): result = u'ut_server_and_client_set_channel_name'
↪  (<type 'unicode'>)
```

B Trace for testrun with python 3.8.5 (final)

B.1 Tests with status Info (18)

B.1.1 socket_protocol.struct_json_protocol: Send and receive check.

Testresult

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

Info	Send and received data by struct_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
Send data: (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
Receive data (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
Send data: (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
Receive data (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Request Status (Okay) transfered via struct_json_protocol is correct (Content 0 and Type is <class 'int'>).

```
Result (Request Status (Okay) transfered via struct_json_protocol): 0 (<class 'int'>)
Expectation (Request Status (Okay) transfered via struct_json_protocol): result = 0 (<class
↪  'int'>)
```

Success	Request Data transfered via struct_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

```
Result (Request Data transfered via struct_json_protocol): { 'test': 'test' } (<class 'dict'>)
Expectation (Request Data transfered via struct_json_protocol): result = { 'test': 'test' }
↪  (<class 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via struct_json_protocol is correct (Content 5 and Type is <class 'int'>).

```
Result (Response Status (Operation not permitted) transfered via struct_json_protocol): 5
↪  (<class 'int'>)
Expectation (Response Status (Operation not permitted) transfered via struct_json_protocol):
↪  result = 5 (<class 'int'>)
```

Success	Response Data transfered via struct_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).

```
Result (Response Data transfered via struct_json_protocol): [ 1, 3, 's' ] (<class 'list'>)
Expectation (Response Data transfered via struct_json_protocol): result = [ 1, 3, 's' ]
↪  (<class 'list'>)
```

Success	Return Value (request instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for struct_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for struct_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for struct_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for struct_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

B.1.2 socket_protocol.pure_json_protocol: Send and receive check.

Testresult

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

Info	Send and received data by pure_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
Send data: (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 89
Receive data (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 89
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
Send data: (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 60 f8 dc 89
Receive data (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 60 f8 dc 89
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<class 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<class
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { 'test': 'test' } (<class 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { 'test': 'test' }
↪  (<class 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<class 'int'>)
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<class 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, 's' ] (<class 'list'>)
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, 's' ] (<class
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

B.1.3 socket_protocol.pure_json_protocol: Send and receive check including authentification.

Testresult

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

Info	Send and received data by pure_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Requesting seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 1, data_id: 0, data: "None"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 9e 85 7b 8d
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 9e 85 7b 8d
socket_protocol (server): RX <- status: 0, service_id: 1, data_id: 0, data: "None"
socket_protocol (server): Executing callback __authentificate_create_seed__ to process
↪  received data
socket_protocol (server): Got seed request, sending seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 2, data_id: 0, data:
↪  "'009397399f4a930023d86cd08371ecf17f3c1879a494a3ef4731cf839d1b215f'"
Send data: (124): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 32 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 30
↪  30 39 33 39 37 33 39 39 66 34 61 39 33 30 30 32 33 64 38 36 63 64 30 38 33 37 31 65 63 66
↪  31 37 66 33 63 31 38 37 39 61 34 39 34 61 33 65 66 34 37 33 31 63 66 38 33 39 64 31 62 32
↪  31 35 66 22 7d 98 9d 40 14
Receive data (124): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 32 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  30 30 39 33 39 37 33 39 39 66 34 61 39 33 30 30 32 33 64 38 36 63 64 30 38 33 37 31 65 63
↪  66 31 37 66 33 63 31 38 37 39 61 34 39 34 61 33 65 66 34 37 33 31 63 66 38 33 39 64 31 62
↪  32 31 35 66 22 7d 98 9d 40 14
socket_protocol (server): RX <- status: 0, service_id: 2, data_id: 0, data:
↪  "'009397399f4a930023d86cd08371ecf17f3c1879a494a3ef4731cf839d1b215f'"
socket_protocol (server): Executing callback __authentificate_create_key__ to process
↪  received data
socket_protocol (server): Got seed, sending key for authentification
socket_protocol (server): TX -> status: 0, service_id: 3, data_id: 0, data:
↪  "'5f8c68aa5c740a0841eeb15d14a72b228a30fa2bed55b4a69f17f1f94e0fa7182d53233ce0d06722a5a6b95
↪  0f7d17984a1aa2808fd665c1655a8ad193d604054'"
Send data: (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 33 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 35
↪  66 38 63 36 38 61 61 35 63 37 34 30 61 30 38 34 31 65 65 62 31 35 64 31 34 61 37 32 62 32
↪  32 38 61 33 30 66 61 32 62 65 64 35 35 62 34 61 36 39 66 31 37 66 31 66 39 34 65 30 66 61
↪  37 31 38 32 64 35 33 32 33 33 63 65 30 64 30 36 37 32 32 61 35 61 36 62 39 35 30 66 37 64
↪  31 37 39 38 34 61 31 61 61 32 38 30 38 66 64 36 36 35 63 31 36 35 35 61 38 61 64 31 39 33
↪  64 36 30 34 30 35 34 22 7d a8 90 36 be
Receive data (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 33 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  35 66 38 63 36 38 61 61 35 63 37 34 30 61 30 38 34 31 65 65 62 31 35 64 31 34 61 37 32 62
↪  32 32 38 61 33 30 66 61 32 62 65 64 35 35 62 34 61 36 39 66 31 37 66 31 66 39 34 65 30 66
↪  61 37 31 38 32 64 35 33 32 33 33 63 65 30 64 30 36 37 32 32 61 35 61 36 62 39 35 30 66 37
↪  64 31 37 39 38 34 61 31 61 61 32 38 30 38 66 64 36 36 35 63 31 36 35 35 61 38 61 64 31 39
↪  33 64 36 30 34 30 35 34 22 7d a8 90 36 be
socket_protocol (server): RX <- status: 0, service_id: 3, data_id: 0, data:
↪  "'5f8c68aa5c740a0841eeb15d14a72b228a30fa2bed55b4a69f17f1f94e0fa7182d53233ce0d06722a5a6b95
```

```
Result (Return value of authentification): True (<class 'bool'>)
Expectation (Return value of authentification): result = True (<class 'bool'>)
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<class 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<class
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { 'test': 'test' } (<class 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { 'test': 'test' }
↪  (<class 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<class 'int'>)
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<class 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, 's' ] (<class 'list'>)
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, 's' ] (<class
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

B.1.4 socket_protocol.pure_json_protocol: Wildcard Callback registration for service_id and data_id.

Testresult

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

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for service_id and data_id.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 48879, data: "{'test':
↪  'test'}"
Send data: (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d cc a2 b9 e6
Receive data (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d cc a2 b9 e6
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 48879, data: "{'test':
↪  'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
Send data: (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 7d 1e 6e 1b
Receive data (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 7d 1e 6e 1b
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<class 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<class
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { 'test': 'test' } (<class 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { 'test': 'test' }
↪  (<class 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<class 'int'>)
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<class 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, 's' ] (<class 'list'>)
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, 's' ] (<class
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 48879) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 48879) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xbeef): result = None (<class 'NoneType'>)
```

B.1.5 socket_protocol.pure_json_protocol: Wildcard Callback registration for service_id.

Testresult

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

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for service_id.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 48879, data: "{'test':
↪  'test'}"
Send data: (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d cc a2 b9 e6
Receive data (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d cc a2 b9 e6
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 48879, data: "{'test':
↪  'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
Send data: (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 7d 1e 6e 1b
Receive data (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 7d 1e 6e 1b
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<class 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<class
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { 'test': 'test' } (<class 'dict'>)
```

```
Expectation (Request Data transfered via pure_json_protocol): result = { 'test': 'test' }
↪  (<class 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<class 'int'>)
```

```
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<class 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, 's' ] (<class 'list'>)
```

```
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, 's' ] (<class
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 48879) not
↪  in buffer.
```

```
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<class 'NoneType'>)
```

```
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 48879) not
↪  in buffer.
```

```
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<class 'NoneType'>)
```

```
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xbeef): result = None (<class 'NoneType'>)
```

B.1.6 socket_protocol.pure_json_protocol: Wildcard Callback registration for data_id.

Testresult

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

Info	Send and received data by pure_json_protocol. Wildcard callback registerd for .

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 48879, data: "{'test':
↪  'test'}"
Send data: (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d cc a2 b9 e6
Receive data (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d cc a2 b9 e6
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 48879, data: "{'test':
↪  'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
Send data: (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 7d 1e 6e 1b
Receive data (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 7d 1e 6e 1b
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<class 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<class
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { 'test': 'test' } (<class 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { 'test': 'test' }
↪  (<class 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<class 'int'>)
```

```
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<class 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, 's' ] (<class 'list'>)
```

```
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, 's' ] (<class
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 48879) not
↪  in buffer.
```

```
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<class 'NoneType'>)
```

```
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 48879) not
↪  in buffer.
```

```
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<class 'NoneType'>)
```

```
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xbeef): result = None (<class 'NoneType'>)
```

B.1.7 socket_protocol.pure_json_protocol: Register a second Callback with the same service_id.

Testresult

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

Info	Send and received data by pure_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
Send data: (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 89
Receive data (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 89
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
socket_protocol (server): Executing callback response_data_method_2 to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
Send data: (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 60 f8 dc 89
Receive data (74): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 35 2c 20 22 64 61 74 61
↪  22 3a 20 5b 31 2c 20 33 2c 20 22 73 22 5d 7d 60 f8 dc 89
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Request Status (Okay) transfered via pure_json_protocol is correct (Content 0 and Type is <class 'int'>).

```
Result (Request Status (Okay) transfered via pure_json_protocol): 0 (<class 'int'>)
Expectation (Request Status (Okay) transfered via pure_json_protocol): result = 0 (<class
↪  'int'>)
```

Success	Request Data transfered via pure_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

```
Result (Request Data transfered via pure_json_protocol): { 'test': 'test' } (<class 'dict'>)
Expectation (Request Data transfered via pure_json_protocol): result = { 'test': 'test' }
↪  (<class 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via pure_json_protocol is correct (Content 5 and Type is <class 'int'>).

```
Result (Response Status (Operation not permitted) transfered via pure_json_protocol): 5
↪  (<class 'int'>)
```

```
Expectation (Response Status (Operation not permitted) transfered via pure_json_protocol):
↪  result = 5 (<class 'int'>)
```

Success	Response Data transfered via pure_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).

```
Result (Response Data transfered via pure_json_protocol): [ 1, 3, 's' ] (<class 'list'>)
```

```
Expectation (Response Data transfered via pure_json_protocol): result = [ 1, 3, 's' ] (<class
↪  'list'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
```

```
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
```

```
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

B.1.8 socket_protocol.struct_json_protocol: Send and receive check (Twice for coverage of buffer initialisation).

Testresult

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

Info	Send and received data by struct_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
Send data: (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
Receive data (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
Send data: (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
Receive data (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
Send data: (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
Receive data (29): 00 00 00 00 00 00 00 0a 00 00 af fe 7b 22 74 65 73 74 22 3a 20 22 74 65 73
↪  74 22 7d 47
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
socket_protocol (server): Executing callback response_data_method to process received data
socket_protocol (server): TX -> status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
Send data: (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
Receive data (24): 00 00 00 05 00 00 00 0b 00 00 af fe 5b 31 2c 20 33 2c 20 22 73 22 5d 28
socket_protocol (server): RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
socket_protocol (server): Received message has a peculiar status: Operation not permitted
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success Request Status (Okay) transfered via struct_json_protocol is correct (Content 0 and Type is <class 'int'>).

```
Result (Request Status (Okay) transfered via struct_json_protocol): 0 (<class 'int'>)
Expectation (Request Status (Okay) transfered via struct_json_protocol): result = 0 (<class
↪  'int'>)
```

Success	Request Data transfered via struct_json_protocol is correct (Content {'test': 'test'} and Type is <class 'dict'>).

```
Result (Request Data transfered via struct_json_protocol): { 'test': 'test' } (<class 'dict'>)
Expectation (Request Data transfered via struct_json_protocol): result = { 'test': 'test' }
↪  (<class 'dict'>)
```

Success	Response Status (Operation not permitted) transfered via struct_json_protocol is correct (Content 5 and Type is <class 'int'>).

```
Result (Response Status (Operation not permitted) transfered via struct_json_protocol): 5
↪  (<class 'int'>)
Expectation (Response Status (Operation not permitted) transfered via struct_json_protocol):
↪  result = 5 (<class 'int'>)
```

Success	Response Data transfered via struct_json_protocol is correct (Content [1, 3, 's'] and Type is <class 'list'>).

```
Result (Response Data transfered via struct_json_protocol): [ 1, 3, 's' ] (<class 'list'>)
Expectation (Response Data transfered via struct_json_protocol): result = [ 1, 3, 's' ]
↪  (<class 'list'>)
```

Success	Return Value (request instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for struct_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for struct_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for struct_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for struct_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for struct_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

B.1.9 socket_protocol.pure_json_protocol: Checksum corumpation while sending.

Testresult

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

Info	Send data with wrong checksum by pure_json_protocol.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
Send data: (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 89
Receive data (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 8a
socket_protocol (server): Received message has a wrong checksum and will be ignored: (79): 7b
↪  22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64 22
↪  3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 7b 22 74
↪  65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 8a.
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Callback executed variable is correct (Content False and Type is <class 'bool'>).

```
Result (Callback executed variable): False (<class 'bool'>)
Expectation (Callback executed variable): result = False (<class 'bool'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

B.1.10 socket_protocol.pure_json_protocol: Timeout measurement for authentification and send method.

Testresult

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

Success	Timeout for authentification is correct (Content 0.20126914978027344 in [0.2 ... 0.22000000000000003] and Type is <class 'float'>).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Requesting seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 1, data_id: 0, data: "None"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 9e 85 7b 8d
Result (Timeout for authentification): 0.20126914978027344 (<class 'float'>)
Expectation (Timeout for authentification): 0.2 <= result <= 0.22000000000000003
```

Success	Timeout for authentification is correct (Content 0.5021231174468994 in [0.5 ... 0.55] and Type is <class 'float'>).

```
socket_protocol (server): Requesting seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 1, data_id: 0, data: "None"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 9e 85 7b 8d
Result (Timeout for authentification): 0.5021231174468994 (<class 'float'>)
Expectation (Timeout for authentification): 0.5 <= result <= 0.55
```

Success	Timeout for send method is correct (Content 0.20092320442199707 in [0.2 ... 0.22000000000000003] and Type is <class 'float'>).

```
socket_protocol (server): TIMEOUT (0.2s): Requested data (service_id: 30; data_id: 0) not in
↪  buffer.
Result (Timeout for send method): 0.20092320442199707 (<class 'float'>)
Expectation (Timeout for send method): 0.2 <= result <= 0.22000000000000003
```

Success	Timeout for send method is correct (Content 0.5018301010131836 in [0.5 ... 0.55] and Type is <class 'float'>).

```
socket_protocol (server): TIMEOUT (0.5s): Requested data (service_id: 30; data_id: 0) not in
↪  buffer.
Result (Timeout for send method): 0.5018301010131836 (<class 'float'>)
Expectation (Timeout for send method): 0.5 <= result <= 0.55
```

B.1.11 socket_protocol.pure_json_protocol: No Callback at response instance for the request.

Testresult

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

Info Send data, but no callback registered (pure_json_protocol).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
Send data: (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 89
Receive data (79): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 7b 22 74 65 73 74 22 3a 20 22 74 65 73 74 22 7d 7d 82 1c 8b 89
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test':
↪  'test'}"
socket_protocol (server): Received message with no registered callback. Sending negative
↪  response.
socket_protocol (server): TX -> status: 1, service_id: 11, data_id: 45054, data: "None"
Send data: (67): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 31 2c 20 22 64 61 74 61
↪  22 3a 20 6e 75 6c 6c 7d 24 86 3f 75
Receive data (67): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 31 2c 20 22 64 61 74 61
↪  22 3a 20 6e 75 6c 6c 7d 24 86 3f 75
socket_protocol (server): RX <- status: 1, service_id: 11, data_id: 45054, data: "None"
socket_protocol (server): Received message has a peculiar status: Request has no callback.
↪  Data buffered.
socket_protocol (server): Message data is stored in buffer and is now ready to be retrieved
↪  by receive method
```

Success Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
```

```
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Response Status (Request has no callback. Data buffered.) transfered via pure_json_protocol is correct (Content 1 and Type is <class 'int'>).

```
Result (Response Status (Request has no callback. Data buffered.) transfered via
↪  pure_json_protocol): 1 (<class 'int'>)
```

```
Expectation (Response Status (Request has no callback. Data buffered.) transfered via
↪  pure_json_protocol): result = 1 (<class 'int'>)
```

Success	Response Data (no data) transfered via pure_json_protocol is correct (Content None and Type is <class 'NoneType'>).

```
Result (Response Data (no data) transfered via pure_json_protocol): None (<class 'NoneType'>)
```

```
Expectation (Response Data (no data) transfered via pure_json_protocol): result = None
↪  (<class 'NoneType'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
```

```
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
```

```
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
```

```
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

B.1.12 socket_protocol.pure_json_protocol: Authentification required, but not processed/correctly processed.

Testresult

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

Info	Authentification with different secrets for request and response instance (pure_json_protocol).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Requesting seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 1, data_id: 0, data: "None"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 9e 85 7b 8d
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 9e 85 7b 8d
socket_protocol (server): RX <- status: 0, service_id: 1, data_id: 0, data: "None"
socket_protocol (server): Executing callback __authentificate_create_seed__ to process
↪  received data
socket_protocol (server): Got seed request, sending seed for authentification
socket_protocol (server): TX -> status: 0, service_id: 2, data_id: 0, data:
↪  "'c40ab7346f50e5b60ae8910a9ec440251cf4dcc13c6ed66d766926bb96d02b92'"
Send data: (124): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 32 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 63
↪  34 30 61 62 37 33 34 36 66 35 30 65 35 62 36 30 61 65 38 39 31 30 61 39 65 63 34 34 30 32
↪  35 31 63 66 34 64 63 63 31 33 63 36 65 64 36 36 64 37 36 36 39 32 36 62 62 39 36 64 30 32
↪  62 39 32 22 7d 96 af 40 61
Receive data (124): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 32 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  63 34 30 61 62 37 33 34 36 66 35 30 65 35 62 36 30 61 65 38 39 31 30 61 39 65 63 34 34 30
↪  32 35 31 63 66 34 64 63 63 31 33 63 36 65 64 36 36 64 37 36 36 39 32 36 62 62 39 36 64 30
↪  32 62 39 32 22 7d 96 af 40 61
socket_protocol (server): RX <- status: 0, service_id: 2, data_id: 0, data:
↪  "'c40ab7346f50e5b60ae8910a9ec440251cf4dcc13c6ed66d766926bb96d02b92'"
socket_protocol (server): Executing callback __authentificate_create_key__ to process
↪  received data
socket_protocol (server): Got seed, sending key for authentification
socket_protocol (server): TX -> status: 0, service_id: 3, data_id: 0, data:
↪  "'11d996b5b67233e7fca7b67415b64c59ddb91b794647d88b8fb6164e796bbfcaacadcb03e26fb1343585413
↪  357f3381d2e84a663942998b980bee671baa18b2f'"
Send data: (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 33 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 31
↪  31 64 39 39 36 62 35 62 36 37 32 33 33 65 37 66 63 61 37 62 36 37 34 31 35 62 36 34 63 35
↪  39 64 64 62 39 31 62 37 39 34 36 34 37 64 38 38 62 38 66 62 36 31 36 34 65 37 39 36 62 62
↪  66 63 61 61 63 61 64 63 62 30 33 65 32 36 66 62 31 33 34 33 35 38 35 34 31 33 33 35 37 66
↪  33 33 38 31 64 32 65 38 34 61 36 36 33 39 34 32 39 39 38 62 39 38 30 62 65 65 36 37 31 62
↪  61 61 31 38 62 32 66 22 7d 96 c4 04 92
Receive data (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 33 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  31 31 64 39 39 36 62 35 62 36 37 32 33 33 65 37 66 63 61 37 62 36 37 34 31 35 62 36 34 63
↪  35 39 64 64 62 39 31 62 37 39 34 36 34 37 64 38 38 62 38 66 62 36 31 36 34 65 37 39 36 62
↪  62 66 63 61 61 63 61 64 63 62 30 33 65 32 36 66 62 31 33 34 33 35 38 35 34 31 33 33 35 37
↪  66 33 33 38 31 64 32 65 38 34 61 36 36 33 39 34 32 39 39 38 62 39 38 30 62 65 65 36 37 31
↪  62 61 61 31 38 62 32 66 22 7d 96 c4 04 92
socket_protocol (server): RX <- status: 0, service_id: 3, data_id: 0, data:
↪  "'11d996b5b67233e7fca7b67415b64c59ddb91b794647d88b8fb6164e796bbfcaacadcb03e26fb1343585413
```

```
Result (Return value of authentification): False (<class 'bool'>)
Expectation (Return value of authentification): result = False (<class 'bool'>)
```

Success	Return value of send method is correct (Content True and Type is <class 'bool'>).

```
Result (Return value of send method): True (<class 'bool'>)
Expectation (Return value of send method): result = True (<class 'bool'>)
```

Success	Response Status (Authentification required) transfered via pure_json_protocol is correct (Content 2 and Type is <class 'int'>).

```
Result (Response Status (Authentification required) transfered via pure_json_protocol): 2
↪  (<class 'int'>)
Expectation (Response Status (Authentification required) transfered via pure_json_protocol):
↪  result = 2 (<class 'int'>)
```

Success	Response Data (no data) transfered via pure_json_protocol is correct (Content None and Type is <class 'NoneType'>).

```
Result (Response Data (no data) transfered via pure_json_protocol): None (<class 'NoneType'>)
Expectation (Response Data (no data) transfered via pure_json_protocol): result = None
↪  (<class 'NoneType'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

B.1.13 socket_protocol.pure_json_protocol: Authentification processed without secret.

Testresult

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

Info	Authentification with no secret definition (pure_json_protocol).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

Success	Return value of authentification is correct (Content False and Type is <class 'bool'>).

```
Result (Return value of authentification): False (<class 'bool'>)
Expectation (Return value of authentification): result = False (<class 'bool'>)
```

B.1.14 socket_protocol.pure_json_protocol: Incompatible Callback return value(s).

Testresult

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

Info	Send and received data with incompatible callback (pure_json_protocol).

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 45054, data: "None"
Send data: (67): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 6e 75 6c 6c 7d e9 e9 77 c0
Receive data (67): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 35 30 35 34 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 6e 75 6c 6c 7d e9 e9 77 c0
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 45054, data: "None"
socket_protocol (server): Executing callback response_data_method_fail to process received
↪  data
socket_protocol (server): TX -> status: 0, service_id: 10, data_id: 48879, data: "None"
Send data: (67): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 6e 75 6c 6c 7d da 63 1a 84
Receive data (67): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61
↪  22 3a 20 6e 75 6c 6c 7d da 63 1a 84
socket_protocol (server): RX <- status: 0, service_id: 10, data_id: 48879, data: "None"
socket_protocol (server): Received message with no registered callback. Sending negative
↪  response.
socket_protocol (server): TX -> status: 1, service_id: 11, data_id: 48879, data: "None"
Send data: (67): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 31 2c 20 22 64 61 74 61
↪  22 3a 20 6e 75 6c 6c 7d 17 0c 52 31
Receive data (67): 7b 22 64 61 74 61 5f 69 64 22 3a 20 34 38 38 37 39 2c 20 22 73 65 72 76 69
↪  63 65 5f 69 64 22 3a 20 31 31 2c 20 22 73 74 61 74 75 73 22 3a 20 31 2c 20 22 64 61 74 61
↪  22 3a 20 6e 75 6c 6c 7d 17 0c 52 31
socket_protocol (server): RX <- status: 1, service_id: 11, data_id: 48879, data: "None"
socket_protocol (server): Received message has a peculiar status: Request has no callback.
↪  Data buffered.
socket_protocol (server): Executing callback response_data_method_fail to process received
↪  data
```

Success	Exception (TypeError) detection variable is correct (Content True and Type is <class 'bool'>).

```
Result (Exception (TypeError) detection variable): True (<class 'bool'>)
Expectation (Exception (TypeError) detection variable): result = True (<class 'bool'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 45054) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xaffe is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 45054) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xaffe): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xaffe): result = None (<class 'NoneType'>)
```

Success	Exception (TypeError) detection variable is correct (Content True and Type is <class 'bool'>).

```
Result (Exception (TypeError) detection variable): True (<class 'bool'>)
Expectation (Exception (TypeError) detection variable): result = True (<class 'bool'>)
```

Success	Return Value (request instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 11; data_id: 48879) not
↪  in buffer.
Result (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<class 'NoneType'>)
Expectation (Return Value (request instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): result = None (<class 'NoneType'>)
```

Success	Return Value (response instance) for pure_json_protocol.receive with data data_id 0xbeef is correct (Content None and Type is <class 'NoneType'>).

```
socket_protocol (server): TIMEOUT (0.1s): Requested data (service_id: 10; data_id: 48879) not
↪  in buffer.
Result (Return Value (response instance) for pure_json_protocol.receive with data data_id
↪  0xbeef): None (<class 'NoneType'>)
Expectation (Return Value (response instance) for pure_json_protocol.receive with data
↪  data_id 0xbeef): result = None (<class 'NoneType'>)
```

B.1.15 socket_protocol: Server setting the channel name.

Testresult

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

Info	Initiating communication including channel_name exchange.

```
ut_server_set_channel_name (server): Cleaning up receive-buffer
ut_server_set_channel_name (server): Resetting authentification state to
↪  AUTH_STATE_UNKNOWN_CLIENT
Overwriting existing callback '__channel_name_response__' for service_id (6) and data_id (0)
↪  to 'ut_response_callback'!
socket_protocol (client): Cleaning up receive-buffer
socket_protocol (client): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
ut_server_set_channel_name (server): Cleaning up receive-buffer
ut_server_set_channel_name (server): TX -> status: 0, service_id: 5, data_id: 0, data:
↪  "'ut_server_set_channel_name'"
Send data: (86): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 35 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 75
↪  74 5f 73 65 72 76 65 72 5f 73 65 74 5f 63 68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 7d 6e a8 f6
↪  56
socket_protocol (client): Cleaning up receive-buffer
Receive data (86): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 35 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 75
↪  74 5f 73 65 72 76 65 72 5f 73 65 74 5f 63 68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 7d 6e a8 f6
↪  56
socket_protocol (client): RX <- status: 0, service_id: 5, data_id: 0, data:
↪  "'ut_server_set_channel_name'"
socket_protocol (client): Executing callback __channel_name_request__ to process received data
ut_server_set_channel_name (client): channel name is now 'ut_server_set_channel_name'
ut_server_set_channel_name (client): TX -> status: 0, service_id: 6, data_id: 0, data: "None"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 36 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 6c e3 72 30
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 36 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 6c e3 72 30
ut_server_set_channel_name (server): RX <- status: 0, service_id: 6, data_id: 0, data: "None"
ut_server_set_channel_name (server): Executing callback ut_response_callback to process
↪  received data
```

Success Channel name for server is correct (Content 'ut_server_set_channel_name' and Type is <class 'str'>).

```
Result (Channel name for server): 'ut_server_set_channel_name' (<class 'str'>)
Expectation (Channel name for server): result = 'ut_server_set_channel_name' (<class 'str'>)
```

Success Channel name for client is correct (Content 'ut_server_set_channel_name' and Type is <class 'str'>).

```
Result (Channel name for client): 'ut_server_set_channel_name' (<class 'str'>)
Expectation (Channel name for client): result = 'ut_server_set_channel_name' (<class 'str'>)
```

B.1.16 socket_protocol: Client setting the channel name.

Testresult

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

Info	Initiating communication including channel_name exchange.

```
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
Overwriting existing callback '__channel_name_response__' for service_id (6) and data_id (0)
↪  to 'ut_response_callback'!
ut_client_set_channel_name (client): Cleaning up receive-buffer
ut_client_set_channel_name (client): Resetting authentification state to
↪  AUTH_STATE_UNKNOWN_CLIENT
socket_protocol (server): Cleaning up receive-buffer
socket_protocol (server): TX -> status: 0, service_id: 5, data_id: 0, data: "None"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 35 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d c9 eb 19 5c
ut_client_set_channel_name (client): Cleaning up receive-buffer
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 35 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d c9 eb 19 5c
ut_client_set_channel_name (client): RX <- status: 0, service_id: 5, data_id: 0, data: "None"
ut_client_set_channel_name (client): Executing callback __channel_name_request__ to process
↪  received data
ut_client_set_channel_name (client): TX -> status: 0, service_id: 6, data_id: 0, data:
↪  "'ut_client_set_channel_name'"
Send data: (86): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 36 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 75
↪  74 5f 63 6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 7d db 90 55
↪  74
Receive data (86): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 36 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 75
↪  74 5f 63 6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65 6c 5f 6e 61 6d 65 22 7d db 90 55
↪  74
socket_protocol (server): RX <- status: 0, service_id: 6, data_id: 0, data:
↪  "'ut_client_set_channel_name'"
socket_protocol (server): Executing callback ut_response_callback to process received data
ut_client_set_channel_name (server): channel name is now 'ut_client_set_channel_name'
```

Success	Channel name for server is correct (Content 'ut_client_set_channel_name' and Type is <class 'str'>).

```
Result (Channel name for server): 'ut_client_set_channel_name' (<class 'str'>)
Expectation (Channel name for server): result = 'ut_client_set_channel_name' (<class 'str'>)
```

Success	Channel name for client is correct (Content 'ut_client_set_channel_name' and Type is <class 'str'>).

```
Result (Channel name for client): 'ut_client_set_channel_name' (<class 'str'>)
Expectation (Channel name for client): result = 'ut_client_set_channel_name' (<class 'str'>)
```

B.1.17 socket_protocol: Server and Client setting different channel names.

Testresult

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

Info	Initiating communication including channel_name exchange.

```
ut_server_and_client_set_channel_name (server): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (server): Resetting authentification state to
↪ AUTH_STATE_UNKNOWN_CLIENT
Overwriting existing callback '__channel_name_response__' for service_id (6) and data_id (0)
↪ to 'ut_response_callback'!
foo (client): Cleaning up receive-buffer
foo (client): Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
ut_server_and_client_set_channel_name (server): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (server): TX -> status: 0, service_id: 5, data_id: 0,
↪ data: "'ut_server_and_client_set_channel_name'"
Send data: (97): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪ 64 22 3a 20 35 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 75
↪ 74 5f 73 65 72 76 65 72 5f 61 6e 64 5f 63 6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65
↪ 6c 5f 6e 61 6d 65 22 7d fc 87 b8 63
foo (client): Cleaning up receive-buffer
Receive data (97): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪ 64 22 3a 20 35 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 75
↪ 74 5f 73 65 72 76 65 72 5f 61 6e 64 5f 63 6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65
↪ 6c 5f 6e 61 6d 65 22 7d fc 87 b8 63
foo (client): RX <- status: 0, service_id: 5, data_id: 0, data:
↪ "'ut_server_and_client_set_channel_name'"
foo (client): Executing callback __channel_name_request__ to process received data
ut_server_and_client_set_channel_name (client): overwriting user defined channel name from
↪ 'foo' to 'ut_server_and_client_set_channel_name'
ut_server_and_client_set_channel_name (client): TX -> status: 0, service_id: 6, data_id: 0,
↪ data: "None"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪ 64 22 3a 20 36 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪ 6c 6c 7d 6c e3 72 30
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪ 64 22 3a 20 36 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪ 6c 6c 7d 6c e3 72 30
ut_server_and_client_set_channel_name (server): RX <- status: 0, service_id: 6, data_id: 0,
↪ data: "None"
ut_server_and_client_set_channel_name (server): Executing callback ut_response_callback to
↪ process received data
```

Success	Channel name for server is correct (Content 'ut_server_and_client_set_channel_name' and Type is <class 'str'>).

```
Result (Channel name for server): 'ut_server_and_client_set_channel_name' (<class 'str'>)
Expectation (Channel name for server): result = 'ut_server_and_client_set_channel_name'
↪ (<class 'str'>)
```

Success	Channel name for client is correct (Content 'ut_server_and_client_set_channel_name' and Type is <class 'str'>).

```
Result (Channel name for client): 'ut_server_and_client_set_channel_name' (<class 'str'>)
```

```
Expectation (Channel name for client): result = 'ut_server_and_client_set_channel_name'
↪  (<class 'str'>)
```

B.1.18 socket_protocol: Server and Client setting the same channel name.

Testresult

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

Info	Initiating communication including channel_name exchange.

```
ut_server_and_client_set_channel_name (server): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (server): Resetting authentification state to
↪  AUTH_STATE_UNKNOWN_CLIENT
Overwriting existing callback '__channel_name_response__' for service_id (6) and data_id (0)
↪  to 'ut_response_callback'!
ut_server_and_client_set_channel_name (client): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (client): Resetting authentification state to
↪  AUTH_STATE_UNKNOWN_CLIENT
ut_server_and_client_set_channel_name (server): Cleaning up receive-buffer
ut_server_and_client_set_channel_name (server): TX -> status: 0, service_id: 5, data_id: 0,
↪  data: "'ut_server_and_client_set_channel_name'"
Send data: (97): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 35 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 75
↪  74 5f 73 65 72 76 65 72 5f 61 6e 64 5f 63 6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65
↪  6c 5f 6e 61 6d 65 22 7d fc 87 b8 63
ut_server_and_client_set_channel_name (client): Cleaning up receive-buffer
Receive data (97): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 35 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 75
↪  74 5f 73 65 72 76 65 72 5f 61 6e 64 5f 63 6c 69 65 6e 74 5f 73 65 74 5f 63 68 61 6e 6e 65
↪  6c 5f 6e 61 6d 65 22 7d fc 87 b8 63
ut_server_and_client_set_channel_name (client): RX <- status: 0, service_id: 5, data_id: 0,
↪  data: "'ut_server_and_client_set_channel_name'"
ut_server_and_client_set_channel_name (client): Executing callback __channel_name_request__
↪  to process received data
ut_server_and_client_set_channel_name (client): TX -> status: 0, service_id: 6, data_id: 0,
↪  data: "None"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 36 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 6c e3 72 30
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 36 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 6e 75
↪  6c 6c 7d 6c e3 72 30
ut_server_and_client_set_channel_name (server): RX <- status: 0, service_id: 6, data_id: 0,
↪  data: "None"
ut_server_and_client_set_channel_name (server): Executing callback ut_response_callback to
↪  process received data
```

Success	Channel name for server is correct (Content 'ut_server_and_client_set_channel_name' and Type is <class 'str'>).

```
Result (Channel name for server): 'ut_server_and_client_set_channel_name' (<class 'str'>)
Expectation (Channel name for server): result = 'ut_server_and_client_set_channel_name'
↪  (<class 'str'>)
```

Success	Channel name for client is correct (Content 'ut_server_and_client_set_channel_name' and Type is <class 'str'>).

```
Result (Channel name for client): 'ut_server_and_client_set_channel_name' (<class 'str'>)
```

```
Expectation (Channel name for client): result = 'ut_server_and_client_set_channel_name'
↪  (<class 'str'>)
```

C Test-Coverage

C.1 socket_protocol

The line coverage for socket_protocol was 98.7%
The branch coverage for socket_protocol was 98.7%

C.1.1 socket_protocol.__init__.py

The line coverage for socket_protocol.__init__.py was 98.7%
The branch coverage for socket_protocol.__init__.py was 98.7%

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
"""
socket_protocol (Socket Protocol)
=================================

**Author:**

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

**Description:**

    This Module supports point to point communication for client-server issues.

**Submodules:**

* :class:`socket_protocol.struct_json_protocol`
* :class:`socket_protocol.pure_json_protocol`

**Unittest:**

        See also the :download:`unittest <../../socket_protocol/_testresults_/unittest.pdf>`
    documentation.
"""
__DEPENDENCIES__ = ['stringtools']

import stringtools

import binascii
import hashlib
import json
import logging
import os
import struct
import sys
import time


```

```
try:
    from config import APP_NAME as ROOT_LOGGER_NAME
except ImportError:
    ROOT_LOGGER_NAME = 'root'
logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__)


__DESCRIPTION__ = """The Module {\\tt %s} is designed to pack and unpack data for serial
    transportation.
For more Information read the sphinx documentation.""" % __name__.replace('_', '\_')
"""The Module Description"""
__INTERPRETER__ = (2, 3)
"""The Tested Interpreter-Versions"""


class callback_storage(dict):
    def __init__(self):
        dict.__init__(self)

    def get(self, service_id, data_id):
        if service_id is not None and data_id is not None:
            try:
                return self[service_id][data_id]
            except KeyError:
                pass  # nothing to append
        if data_id is not None:
            try:
                return self[None][data_id]
            except KeyError:
                pass  # nothing to append
        if service_id is not None:
            try:
                return self[service_id][None]
            except KeyError:
                pass  # nothing to append
        try:
            return self[None][None]
        except KeyError:
            pass  # nothing to append
        return (None, None, None)

    def add(self, service_id, data_id, callback, *args, **kwargs):
        cb_data = self.get(service_id, data_id)
        if cb_data != (None, None, None):
            logger.warning("Overwriting existing callback %s for service_id (%s) and data_id (%s)
     to %s!", repr(cb_data[0].__name__), repr(service_id), repr(data_id), repr(callback.__name__
    ))
        if service_id not in self:
            self[service_id] = {}
        self[service_id][data_id] = (callback, args, kwargs)


class data_storage(dict):
    KEY_STATUS = 'status'
    KEY_SERVICE_ID = 'service_id'
    KEY_DATA_ID = 'data_id'
    KEY_DATA = 'data'

    def __init__(self, *args, **kwargs):
        dict.__init__(self, *args, **kwargs)

```

```
97      def get_status(self, default=None):
98          return self.get(self.KEY_STATUS, default)
99
100     def get_service_id(self, default=None):
101         return self.get(self.KEY_SERVICE_ID, default)
102
103     def get_data_id(self, default=None):
104         return self.get(self.KEY_DATA_ID, default)
105
106     def get_data(self, default=None):
107         return self.get(self.KEY_DATA, default)
108
109
110 class struct_json_protocol(object):
111     """
112     :param comm_instance: a communication instance supportin at least these functions: :func:`
        register_callback`, :func:`register_disconnect_callback`, :func:`send`.
113     :type comm_instance: instance
114     :param secret: A secret (e.g. created by ``binascii.hexlify(os.urandom(24))``).
115     :type secret: str
116
117     This communication protocol supports to transfer a Service-ID, Data-ID and Data. The
        transmitted data is shorter than :class:`pure_json_protocol`.
118
119     .. note::
120         This class is here for compatibility reasons. Usage of :class:`pure_json_protocol` is
        recommended.
121
122     **Example:**
123
124     Server:
125
126     .. literalinclude:: ../../socket_protocol/_examples_/
        socket_protocol__struct_json_protocol_server.py
127
128     .. literalinclude:: ../../socket_protocol/_examples_/
        socket_protocol__struct_json_protocol_server.log
129
130
131     Client:
132
133     .. literalinclude:: ../../socket_protocol/_examples_/
        socket_protocol__struct_json_protocol_client.py
134
135     .. literalinclude:: ../../socket_protocol/_examples_/
        socket_protocol__struct_json_protocol_client.log
136     """
137     SID_AUTH_SEED_REQUEST = 1
138     SID_AUTH_KEY_REQUEST = 2
139     SID_AUTH_KEY_CHECK_REQUEST = 3
140     SID_AUTH_KEY_CHECK_RESPONSE = 4
141     SID_CHANNEL_NAME_REQUEST = 5
142     SID_CHANNEL_NAME_RESPONSE = 6
143     SID_READ_REQUEST = 10
144     SID_READ_RESPONSE = 11
145     SID_WRITE_REQUEST = 20
146     SID_WRITE_RESPONSE = 21
147     SID_EXECUTE_REQUEST = 30
148     SID_EXECUTE_RESPONSE = 31
149
150     SID_RESPONSE_DICT = {SID_AUTH_SEED_REQUEST: SID_AUTH_KEY_REQUEST,
```

```
151                         SID_AUTH_KEY_REQUEST: SID_AUTH_KEY_CHECK_REQUEST,
152                         SID_AUTH_KEY_CHECK_REQUEST: SID_AUTH_KEY_CHECK_RESPONSE,
153                         SID_CHANNEL_NAME_REQUEST: SID_CHANNEL_NAME_RESPONSE,
154                         SID_READ_REQUEST: SID_READ_RESPONSE,
155                         SID_WRITE_REQUEST: SID_WRITE_RESPONSE,
156                         SID_EXECUTE_REQUEST: SID_EXECUTE_RESPONSE}
157
158     SID_AUTH_LIST = [SID_AUTH_SEED_REQUEST, SID_AUTH_KEY_REQUEST, SID_AUTH_KEY_CHECK_REQUEST,
    SID_AUTH_KEY_CHECK_RESPONSE]
159
160     STATUS_OKAY = 0
161     STATUS_BUFFERING_UNHANDLED_REQUEST = 1
162     STATUS_AUTH_REQUIRED = 2
163     STATUS_SERVICE_OR_DATA_UNKNOWN = 3
164     STATUS_CHECKSUM_ERROR = 4
165     STATUS_OPERATION_NOT_PERMITTED = 5
166     STATUS_NAMES = {STATUS_OKAY: 'Okay',
167                     STATUS_BUFFERING_UNHANDLED_REQUEST: 'Request has no callback. Data buffered.'
    ,
168                     STATUS_AUTH_REQUIRED: 'Authentification required',
169                     STATUS_SERVICE_OR_DATA_UNKNOWN: 'Service or Data unknown',
170                     STATUS_CHECKSUM_ERROR: 'Checksum Error',
171                     STATUS_OPERATION_NOT_PERMITTED: 'Operation not permitted'}
172
173     AUTH_STATE_UNKNOWN_CLIENT = 0
174     AUTH_STATE_SEED_REQUESTED = 1
175     AUTH_STATE_SEED_TRANSFERRED = 2
176     AUTH_STATE_KEY_TRANSFERRED = 3
177     AUTH_STATE_TRUSTED_CLIENT = 4
178     AUTH_STATUS_NAMES = {AUTH_STATE_UNKNOWN_CLIENT: 'Unknown Client',
179                          AUTH_STATE_SEED_REQUESTED: 'Seed was requested',
180                          AUTH_STATE_SEED_TRANSFERRED: 'Seed has been sent',
181                          AUTH_STATE_KEY_TRANSFERRED: 'Key has been sent',
182                          AUTH_STATE_TRUSTED_CLIENT: 'Trusted Client'}
183
184     def __init__(self, comm_instance, secret=None, auto_auth=False, channel_name=None):
185         self.__comm_inst__ = comm_instance
186         self.__secret__ = secret
187         self.__auto_auth__ = auto_auth
188         self.__channel_name__ = channel_name
189         #
190         self.__clean_receive_buffer__()
191         self.__callbacks__ = callback_storage()
192         self.__callbacks__.add(self.SID_AUTH_SEED_REQUEST, 0, self.__authentificate_create_seed__
    )
193         self.__callbacks__.add(self.SID_AUTH_KEY_REQUEST, 0, self.__authentificate_create_key__)
194         self.__callbacks__.add(self.SID_AUTH_KEY_CHECK_REQUEST, 0, self.
    __authentificate_check_key__)
195         self.__callbacks__.add(self.SID_AUTH_KEY_CHECK_RESPONSE, 0, self.
    __authentificate_process_feedback__)
196         self.__callbacks__.add(self.SID_CHANNEL_NAME_REQUEST, 0, self.__channel_name_request__)
197         self.__callbacks__.add(self.SID_CHANNEL_NAME_RESPONSE, 0, self.__channel_name_response__)
198         self.__authentification_state_reset__()
199         self.__seed__ = None
200         self.__comm_inst__.register_callback(self.__data_available_callback__)
201         self.__comm_inst__.register_connect_callback(self.__connection_established__)
202         self.__comm_inst__.register_disconnect_callback(self.__authentification_state_reset__)
203
204     def __log_prefix__(self):
205         postfix = ' (client)' if self.__comm_inst__.IS_CLIENT else ' (server)'
206         if self.__channel_name__ is None:
207             return __name__ + postfix + ':'
```

```
208         else:
209             return self.__channel_name__ + postfix + ':'
210 
211     def connected(self):
212         return self.__comm_inst__.is_connected()
213 
214     def connection_established(self):
215         return self.connected() and (self.__secret__ is None or self.check_authentification_state
    ())
216 
217     def reconnect(self):
218         return self.__comm_inst__.reconnect()
219 
220     def __connection_established__(self):
221         self.__clean_receive_buffer__()
222         if not self.__comm_inst__.IS_CLIENT:
223             self.send(self.SID_CHANNEL_NAME_REQUEST, 0, self.__channel_name__)
224         if self.__auto_auth__ and self.__comm_inst__.IS_CLIENT and self.__secret__ is not None:
225             self.authentificate()
226 
227     def __channel_name_request__(self, msg):
228         data = msg.get_data()
229         if data is None:
230             return self.STATUS_OKAY, self.__channel_name__
231         else:
232             prev_channel_name = self.__channel_name__
233             self.__channel_name__ = data
234             if prev_channel_name is not None and prev_channel_name != data:
235                 logger.warning('%s overwriting user defined channel name from %s to %s', self.
    __log_prefix__(), repr(prev_channel_name), repr(data))
236             elif prev_channel_name is None:
237                 logger.info('%s channel name is now %s', self.__log_prefix__(), repr(self.
    __channel_name__))
238             return self.STATUS_OKAY, None
239 
240     def __channel_name_response__(self, msg):
241         data = msg.get_data()
242         if self.__channel_name__ is None and data is not None:
243             self.__channel_name__ = data
244             logger.info('%s channel name is now %s', self.__log_prefix__(), repr(self.
    __channel_name__))
245         return self.STATUS_OKAY, None
246 
247     def __authentification_state_reset__(self):
248         logger.info("%s Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT", self.
    __log_prefix__())
249         self.__authentification_state__ = self.AUTH_STATE_UNKNOWN_CLIENT
250 
251     def __analyse_frame__(self, frame):
252         status, service_id, data_id = struct.unpack('>III', frame[0:12])
253         if sys.version_info >= (3, 0):
254             data = json.loads(frame[12:-1].decode('utf-8'))
255         else:
256             data = json.loads(frame[12:-1])
257         return self.__mk_msg__(status, service_id, data_id, data)
258 
259     def __build_frame__(self, service_id, data_id, data, status=STATUS_OKAY):
260         frame = struct.pack('>III', status, service_id, data_id)
261         if sys.version_info >= (3, 0):
262             frame += bytes(json.dumps(data), 'utf-8')
263             frame += self.__calc_chksum__(frame)
264         else:
```

```
265             frame += json.dumps(data)
266             frame += self.__calc_chksum__(frame)
267         return frame
268 
269     def __calc_chksum__(self, raw_data):
270         chksum = 0
271         for b in raw_data:
272             if sys.version_info >= (3, 0):
273                 chksum ^= b
274             else:
275                 chksum ^= ord(b)
276         if sys.version_info >= (3, 0):
277             return bytes([chksum])
278         else:
279             return chr(chksum)
280 
281     def __check_frame_checksum__(self, frame):
282         return self.__calc_chksum__(frame[:-1]) == frame[-1:]
283 
284     def __data_available_callback__(self, comm_inst):
285         frame = comm_inst.receive()
286         if not self.__check_frame_checksum__(frame):
287             logger.warning("%s Received message has a wrong checksum and will be ignored: %s.",
    self.__log_prefix__(), stringtools.hexlify(frame))
288         else:
289             msg = self.__analyse_frame__(frame)
290             logger.info(
291                 '%s RX <- status: %s, service_id: %s, data_id: %s, data: "%s"',
292                 self.__log_prefix__(),
293                 repr(msg.get_status()),
294                 repr(msg.get_service_id()),
295                 repr(msg.get_data_id()),
296                 repr(msg.get_data())
297             )
298             callback, args, kwargs = self.__callbacks__.get(msg.get_service_id(), msg.get_data_id
    ())
299             if msg.get_service_id() in self.SID_RESPONSE_DICT.keys():
300                 #
301                 # REQUEST RECEIVED
302                 #
303                 if self.__secret__ is not None and not self.check_authentification_state() and
    msg.get_service_id() not in self.SID_AUTH_LIST:
304                     status = self.STATUS_AUTH_REQUIRED
305                     data = None
306                     logger.warning("%s Received message needs authentification: %s. Sending
    negative response.", self.__log_prefix__(), self.AUTH_STATUS_NAMES.get(self.
    __authentification_state__, 'Unknown authentification status!'))
307                 elif callback is None:
308                     logger.warning("%s Received message with no registered callback. Sending
    negative response.", self.__log_prefix__())
309                     status = self.STATUS_BUFFERING_UNHANDLED_REQUEST
310                     data = None
311                 else:
312                     try:
313                         logger.debug("%s Executing callback %s to process received data", self.
    __log_prefix__(), callback.__name__)
314                         status, data = callback(msg, *args, **kwargs)
315                     except TypeError:
316                         raise TypeError('Check return value of callback function {callback_name}
    for service_id  {service_id} and data_id {data_id}'.format(callback_name=callback.__name__,
    service_id=repr(msg.get_service_id()), data_id=repr(msg.get_data_id())))
317                 self.send(self.SID_RESPONSE_DICT[msg.get_service_id()], msg.get_data_id(), data,
    status=status)
```

```
318             else:
319                 #
320                 # RESPONSE RECEIVED
321                 #
322                 if msg.get_status() not in [self.STATUS_OKAY]:
323                     logger.warning("%s Received message has a peculiar status: %s", self.
    __log_prefix__(), self.STATUS_NAMES.get(msg.get_status(), 'Unknown status response!'))
324                 if callback is None:
325                     status = self.STATUS_OKAY
326                     data = None
327                     self.__buffer_received_data__(msg)
328                 else:
329                     try:
330                         logger.debug("%s Executing callback %s to process received data", self.
    __log_prefix__(), callback.__name__)
331                         status, data = callback(msg, *args, **kwargs)
332                     except TypeError:
333                         raise TypeError('Check return value of callback function {callback_name}
    for service_id  {service_id} and data_id {data_id}'.format(callback_name=callback.__name__,
    service_id=repr(msg.get_service_id()), data_id=repr(msg.get_data_id())))
334 
335     def __buffer_received_data__(self, msg):
336         if not msg.get_service_id() in self.__msg_buffer__:
337             self.__msg_buffer__[msg.get_service_id()] = {}
338         if not msg.get_data_id() in self.__msg_buffer__[msg.get_service_id()]:
339             self.__msg_buffer__[msg.get_service_id()][msg.get_data_id()] = []
340         self.__msg_buffer__[msg.get_service_id()][msg.get_data_id()].append(msg)
341         logger.debug("%s Message data is stored in buffer and is now ready to be retrieved by
    receive method", self.__log_prefix__())
342 
343     def __clean_receive_buffer__(self):
344         logger.debug("%s Cleaning up receive-buffer", self.__log_prefix__())
345         self.__msg_buffer__ = {}
346 
347     def receive(self, service_id, data_id, timeout=1):
348         data = None
349         cnt = 0
350         while data is None and cnt < timeout * 10:
351             try:
352                 data = self.__msg_buffer__.get(service_id, {}).get(data_id, []).pop(0)
353             except IndexError:
354                 data = None
355             cnt += 1
356             time.sleep(0.1)
357         if data is None and cnt >= timeout * 10:
358             logger.warning('%s TIMEOUT (%ss): Requested data (service_id: %s; data_id: %s) not in
     buffer.', self.__log_prefix__(), repr(timeout), repr(service_id), repr(data_id))
359         return data
360 
361     def __mk_msg__(self, status, service_id, data_id, data):
362         return data_storage({data_storage.KEY_DATA_ID: data_id, data_storage.KEY_SERVICE_ID:
    service_id, data_storage.KEY_STATUS: status, data_storage.KEY_DATA: data})
363 
364     def send(self, service_id, data_id, data, status=STATUS_OKAY, timeout=2, log_lvl=logging.INFO
    ):
365         """
366         :param service_id: The Service-ID for the message. See class definitions starting with ``
    SERVICE_``.
367         :type service_id: int
368         :param data_id: The Data-ID for the message.
369         :type data_id: int
370         :param data: The data to be transfered. The data needs to be json compatible.
```

```
371         :type data: str
372         :param status: The Status for the message. All requests should have ``STATUS_OKAY``.
373         :type status: int
374         :param timeout: The timeout for sending data (e.g. time to establish new connection).
375         :type timeout: float
376         :param rx_log_lvl: The log level to log outgoing TX-data
377         :type rx_log_lvl: int
378         :return: True if data had been sent, otherwise False.
379         :rtype: bool
380 
381         This methods sends out a message with the given content.
382         """
383         logger.log(log_lvl, '%s TX -> status: %d, service_id: %d, data_id: %d, data: "%s"', self.__log_prefix__(), status, service_id, data_id, repr(data))
384         return self.__comm_inst__.send(self.__build_frame__(service_id, data_id, data, status), timeout=timeout, log_lvl=logging.DEBUG)
385 
386     def register_callback(self, service_id, data_id, callback, *args, **kwargs):
387         """
388         :param service_id: The Service-ID for the message. See class definitions starting with ``SID_``.
389         :type service_id: int
390         :param data_id: The Data-ID for the message.
391         :type data_id: int
392         :returns: True, if registration was successfull; False, if registration failed (e.g. existance of a callback for this configuration)
393         :rtype: bool
394 
395         This method registers a callback for the given parameters. Givin ``None`` means, that all Service-IDs or all Data-IDs are used.
396         If a message hitting these parameters has been received, the callback will be executed.
397 
398         .. note:: The :func:`callback` is priorised in the following order:
399 
400             * Callbacks with defined Service-ID and Data-ID.
401             * Callbacks with a defined Data-ID.
402             * Callbacks with a defined Service-ID.
403             * Unspecific Callbacks
404 
405         .. note:: The :func:`callback` is executed with these arguments:
406 
407             :param msg: A :class:`dict` containing all message information.
408             :returns: status (see class definition starting with ``STATUS_``), response_data (JSON compatible object)
409         """
410         self.__callbacks__.add(service_id, data_id, callback, *args, **kwargs)
411 
412     def authentificate(self, timeout=2):
413         """
414         :param timeout: The timeout for the authentification (requesting seed, sending key and getting authentification_feedback).
415         :type timeout: float
416         :returns: True, if authentification was successfull; False, if not.
417         :rtype: bool
418 
419         This method authetificates the client at the server.
420 
421         .. note:: An authentification will only processed, if a secret had been given on initialisation.
422 
423         .. note:: Client and Server needs to use the same secret.
424         """
```

```
425         if self.__secret__ is not None:
426             self.__authentification_state__ = self.AUTH_STATE_SEED_REQUESTED
427             logger.info("%s Requesting seed for authentification", self.__log_prefix__())
428             self.send(self.SID_AUTH_SEED_REQUEST, 0, None)
429             cnt = 0
430             while cnt < timeout * 10:
431                 time.sleep(0.1)
432                 if self.__authentification_state__ == self.AUTH_STATE_TRUSTED_CLIENT:
433                     return True
434                 elif self.__authentification_state__ == self.AUTH_STATE_UNKNOWN_CLIENT:
435                     break
436                 cnt += 1
437         return False
438 
439     def check_authentification_state(self):
440         """
441         :return: True, if authentification state is okay, otherwise False
442         :rtype: bool
443         """
444         return self.__authentification_state__ == self.AUTH_STATE_TRUSTED_CLIENT
445 
446     def __authentificate_salt_and_hash__(self, seed):
447         if sys.version_info >= (3, 0):
448             return hashlib.sha512(bytes(seed, 'utf-8') + self.__secret__).hexdigest()
449         else:
450             return hashlib.sha512(seed.encode('utf-8') + self.__secret__.encode('utf-8')).
    hexdigest()
451 
452     def __authentificate_create_seed__(self, msg):
453         logger.info("%s Got seed request, sending seed for authentification", self.__log_prefix__
    ())
454         self.__authentification_state__ = self.AUTH_STATE_SEED_TRANSFERRED
455         if sys.version_info >= (3, 0):
456             self.__seed__ = binascii.hexlify(os.urandom(32)).decode('utf-8')
457         else:
458             self.__seed__ = binascii.hexlify(os.urandom(32))
459         return self.STATUS_OKAY, self.__seed__
460 
461     def __authentificate_create_key__(self, msg):
462         logger.info("%s Got seed, sending key for authentification", self.__log_prefix__())
463         self.__authentification_state__ = self.AUTH_STATE_KEY_TRANSFERRED
464         seed = msg.get_data()
465         key = self.__authentificate_salt_and_hash__(seed)
466         return self.STATUS_OKAY, key
467 
468     def __authentificate_check_key__(self, msg):
469         key = msg.get_data()
470         if key == self.__authentificate_salt_and_hash__(self.__seed__):
471             self.__authentification_state__ = self.AUTH_STATE_TRUSTED_CLIENT
472             logger.info("%s Got correct key, sending positive authentification feedback", self.
    __log_prefix__())
473             return self.STATUS_OKAY, True
474         else:
475             self.__authentification_state__ = self.AUTH_STATE_UNKNOWN_CLIENT
476             logger.info("%s Got incorrect key, sending negative authentification feedback", self.
    __log_prefix__())
477             return self.STATUS_OKAY, False
478 
479     def __authentificate_process_feedback__(self, msg):
480         feedback = msg.get_data()
481         if feedback:
482             self.__authentification_state__ = self.AUTH_STATE_TRUSTED_CLIENT
483             logger.info("%s Got positive authentification feedback", self.__log_prefix__())
```

```
484         else:
485             self.__authentification_state__ = self.AUTH_STATE_UNKNOWN_CLIENT
486             logger.warning("%s Got negative authentification feedback", self.__log_prefix__())
487         return self.STATUS_OKAY, None
488 
489 
490 class pure_json_protocol(struct_json_protocol):
491     """
492     :param comm_instance: a communication instance supportin at least these functions: :func:`
        register_callback`, :func:`register_disconnect_callback`, :func:`send`.
493     :type comm_instance: instance
494     :param secret: A secret (e.g. created by ``binascii.hexlify(os.urandom(24))``).
495     :type secret: str
496 
497     This communication protocol supports to transfer a Service-ID, Data-ID and Data.
498 
499     **Example:**
500 
501     Server:
502 
503     .. literalinclude:: ../../socket_protocol/_examples_/
        socket_protocol__pure_json_protocol_server.py
504 
505     .. literalinclude:: ../../socket_protocol/_examples_/
        socket_protocol__pure_json_protocol_server.log
506 
507 
508     Client:
509 
510     .. literalinclude:: ../../socket_protocol/_examples_/
        socket_protocol__pure_json_protocol_client.py
511 
512     .. literalinclude:: ../../socket_protocol/_examples_/
        socket_protocol__pure_json_protocol_client.log
513     """
514     def __init__(self, *args, **kwargs):
515         struct_json_protocol.__init__(self, *args, **kwargs)
516 
517     def __build_frame__(self, service_id, data_id, data, status=struct_json_protocol.STATUS_OKAY)
        :
518         data_frame = json.dumps(self.__mk_msg__(status, service_id, data_id, data))
519         if sys.version_info >= (3, 0):
520             data_frame = bytes(data_frame, 'utf-8')
521         checksum = self.__calc_chksum__(data_frame)
522         return data_frame + checksum
523 
524     def __analyse_frame__(self, frame):
525         if sys.version_info >= (3, 0):
526             return data_storage(json.loads(frame[:-4].decode('utf-8')))
527         else:
528             return data_storage(json.loads(frame[:-4]))
529 
530     def __calc_chksum__(self, raw_data):
531         return struct.pack('>I', binascii.crc32(raw_data) & 0xffffffff)
532 
533     def __check_frame_checksum__(self, frame):
534         return self.__calc_chksum__(frame[:-4]) == frame[-4:]
```