Unittest for `socket_protocol`

December 26, 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	883349fccf7a5238d7c8d6e4e98afa6d
Dependencies	
stringtools	3eac28a80770a728e1f521fadb92868d

1.2 Unittest Information

Unittest Information	
Version	9553e12e9d4099ace581b5decf091c01
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	12.617s

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	12.581s

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:	2020-12-26 10:11:41,226
Finished-Time:	2020-12-26 10:11:41,229
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:	2020-12-26 10:11:39,812
Finished-Time:	2020-12-26 10:11:41,226
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:	2020-12-26 10:11:37,177
Finished-Time:	2020-12-26 10:11:37,692
Time-Consumption	0.514s

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:	2020-12-26 10:11:41,229
Finished-Time:	2020-12-26 10:11:41,639
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 <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:	2020-12-26 10:11:39,102

Finished-Time:	2020-12-26 10:11:39,811
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:	2020-12-26 10:11:35,237
Finished-Time:	2020-12-26 10:11:35,949
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:	2020-12-26 10:11:31,682
Finished-Time:	2020-12-26 10:11:33,100
Time-Consumption	1.417s

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:	2020-12-26 10:11:30,970
Finished-Time:	2020-12-26 10:11:31,682
Time-Consumption	0.712s

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:	2020-12-26 10:11:37,692
Finished-Time:	2020-12-26 10:11:39,102
Time-Consumption	1.410s

Testsummary:

Success	Timeout for authentification is correct (Content 0.20080995559692383 in [0.2 ... 0.22000000000000003] and Type is <type 'float'>).
Success	Timeout for authentification is correct (Content 0.5016460418701172 in [0.5 ... 0.55] and Type is <type 'float'>).
Success	Timeout for send method is correct (Content 0.20120906829833984 in [0.2 ... 0.22000000000000003] and Type is <type 'float'>).
Success	Timeout for send method is correct (Content 0.5018620491027832 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:	2020-12-26 10:11:34,523
Finished-Time:	2020-12-26 10:11:35,237
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:	2020-12-26 10:11:33,100
Finished-Time:	2020-12-26 10:11:33,812
Time-Consumption	0.712s

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:	2020-12-26 10:11:33,812
Finished-Time:	2020-12-26 10:11:34,522
Time-Consumption	0.710s

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:	2020-12-26 10:11:35,950
Finished-Time:	2020-12-26 10:11:37,175
Time-Consumption	1.226s

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:	2020-12-26 10:11:30,261
Finished-Time:	2020-12-26 10:11:30,970
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:	2020-12-26 10:11:41,952
Finished-Time:	2020-12-26 10:11:42,264
Time-Consumption	0.312s
Testsummary:	
Info	Initiating communication including channel_name exchange.
Success	Channel name for server is correct (Content 'ut_client_set_channel_name' and Type is <type 'str'>).
Success	Channel name for client is correct (Content 'ut_client_set_channel_name' and Type is <type 'str'>).

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:	2020-12-26 10:11:42,265
Finished-Time:	2020-12-26 10:11:42,576
Time-Consumption	0.311s
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 <type 'str'>).
Success	Channel name for client is correct (Content 'ut_server_and_client_set_channel_name' and Type is <type 'str'>).

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:	2020-12-26 10:11:42,577
Finished-Time:	2020-12-26 10:11:42,889
Time-Consumption	0.312s

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

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:	2020-12-26 10:11:41,640
Finished-Time:	2020-12-26 10:11:41,951
Time-Consumption	0.311s

Testsummary:

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

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:	2020-12-26 10:11:54,310
Finished-Time:	2020-12-26 10:11:54,312

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:	2020-12-26 10:11:52,897
Finished-Time:	2020-12-26 10:11:54,310
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:	2020-12-26 10:11:50,269
Finished-Time:	2020-12-26 10:11:50,776
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:	2020-12-26 10:11:54,313
Finished-Time:	2020-12-26 10:11:54,724
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:	2020-12-26 10:11:52,188
Finished-Time:	2020-12-26 10:11:52,897
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 <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:	2020-12-26 10:11:48,346
Finished-Time:	2020-12-26 10:11:49,055
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:	2020-12-26 10:11:44,798
Finished-Time:	2020-12-26 10:11:46,211
Time-Consumption	1.413s

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:	2020-12-26 10:11:44,087
Finished-Time:	2020-12-26 10:11:44,797
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'>).

3.2.9 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:	2020-12-26 10:11:50,777
Finished-Time:	2020-12-26 10:11:52,187
Time-Consumption	1.411s

Testsummary:

Success	Timeout for authentification is correct (Content 0.20149755477905273 in [0.2 ... 0.22000000000000003] and Type is <class 'float'>).
Success	Timeout for authentification is correct (Content 0.5021364688873291 in [0.5 ... 0.55] and Type is <class 'float'>).
Success	Timeout for send method is correct (Content 0.20085549354553223 in [0.2 ... 0.22000000000000003] and Type is <class 'float'>).
Success	Timeout for send method is correct (Content 0.5018312931060791 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:	2020-12-26 10:11:47,632
Finished-Time:	2020-12-26 10:11:48,345
Time-Consumption	0.713s

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:	2020-12-26 10:11:46,212
Finished-Time:	2020-12-26 10:11:46,922
Time-Consumption	0.710s

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:	2020-12-26 10:11:46,922
Finished-Time:	2020-12-26 10:11:47,632
Time-Consumption	0.710s

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:	2020-12-26 10:11:49,056
Finished-Time:	2020-12-26 10:11:50,268
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:	2020-12-26 10:11:43,378
Finished-Time:	2020-12-26 10:11:44,087
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:	2020-12-26 10:11:55,036
Finished-Time:	2020-12-26 10:11:55,346
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:	2020-12-26 10:11:55,346
Finished-Time:	2020-12-26 10:11:55,657
Time-Consumption	0.311s

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:	2020-12-26 10:11:55,657
Finished-Time:	2020-12-26 10:11:55,967
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:	2020-12-26 10:11:54,725
Finished-Time:	2020-12-26 10:11:55,035
Time-Consumption	0.311s

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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test': u'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, u's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test': u'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, u's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Requesting seed for authentification
SP 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
SP server: RX <- status: 0, service_id: 1, data_id: 0, data: "None"
SP server: Executing callback __authentificate_create_seed__ to process received data
SP server: Got seed request, sending seed for authentification
SP server: TX -> status: 0, service_id: 2, data_id: 0, data:
↪  "'3ac139c3e934e4a40e8c5c63c33d02d88d1501c55759efec96b558c20e16d073'"
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 33 61 63 31 33 39 63 33 65 39 33 34 65 34 61
↪  34 30 65 38 63 35 63 36 33 63 33 33 64 30 32 64 38 38 64 31 35 30 31 63 35 35 37 35 39 65
↪  66 65 63 39 36 62 35 35 38 63 32 30 65 31 36 64 30 37 33 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d c8 c7 76 19
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 33 61 63 31 33 39 63 33 65 39 33 34 65 34
↪  61 34 30 65 38 63 35 63 36 33 63 33 33 64 30 32 64 38 38 64 31 35 30 31 63 35 35 37 35 39
↪  65 66 65 63 39 36 62 35 35 38 63 32 30 65 31 36 64 30 37 33 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d c8 c7 76 19
SP server: RX <- status: 0, service_id: 2, data_id: 0, data:
↪  "u'3ac139c3e934e4a40e8c5c63c33d02d88d1501c55759efec96b558c20e16d073'"
SP server: Executing callback __authentificate_create_key__ to process received data
SP server: Got seed, sending key for authentification
SP server: TX -> status: 0, service_id: 3, data_id: 0, data:
↪  "'813c10a4956148bf9a50200f971dba676e7152370d799df78716059aa018129fc61ecc4dc7b4fa57b36015
↪  1a117553313003030b41b5ca929614f6e45a3b661b'"
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 38 31 33 63 31 30 61 34 39 35 36 31 34 38 62
↪  66 39 61 35 30 32 30 30 66 39 37 31 64 62 61 36 37 36 65 37 31 35 32 33 37 30 64 37 39 39
↪  64 66 37 38 37 31 36 30 35 39 61 61 30 31 38 31 32 39 66 63 36 31 65 63 63 34 64 63 37 62
↪  34 66 61 35 37 62 33 36 30 31 35 31 61 31 31 37 35 35 33 33 31 33 30 30 33 30 33 30 62 34
↪  31 62 35 63 61 39 32 39 36 31 34 66 36 65 34 35 61 33 62 36 36 31 62 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 30 7d c6 bc d0 67
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 38 31 33 63 31 30 61 34 39 35 36 31 34 38
↪  62 66 39 61 35 30 32 30 30 66 39 37 31 64 62 61 36 37 36 65 37 31 35 32 33 37 30 64 37 39
↪  39 64 66 37 38 37 31 36 30 35 39 61 61 30 31 38 31 32 39 66 63 36 31 65 63 63 34 64 63 37
↪  62 34 66 61 35 37 62 33 36 30 31 35 31 61 31 31 37 35 35 33 33 31 33 30 30 33 30 33 30 62
↪  34 31 62 35 63 61 39 32 39 36 31 34 66 36 65 34 35 61 33 62 36 36 31 62 22 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 30 7d c6 bc d0 67
SP server: RX <- status: 0, service_id: 3, data_id: 0, data:
↪  "u'813c10a4956148bf9a50200f971dba676e7152370d799df78716059aa018129fc61ecc4dc7b4fa57b3601
↪  51a117553313003030b41b5ca929614f6e45a3b661b'"
SP server: Executing callback __authentificate_check_key__ to process received data
```

```
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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
```

```
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

```
SP server: Cleaning up receive-buffer
```

```
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
```

```
SP 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
```

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

```
SP server: Executing callback response_data_method to process received data
```

```
SP 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
```

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

```
SP server: Received message has a peculiar status: Operation not permitted
```

```
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 48879, data: "{u'test': u'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3, u's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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 .

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 48879, data: "{u'test': u'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3, u's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test': u'test'}"
SP server: Executing callback response_data_method_2 to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, u's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test': u'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, u's']"
SP server: Received message has a peculiar status: Operation not permitted
SP server: Message data is stored in buffer and is now ready to be retrieved by receive
↪  method
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test': u'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, u's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP 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'>).

```
SP 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'>).

```
SP 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 0.20080995559692383 in [0.2 ... 0.22000000000000003] and Type is <type 'float'>).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Requesting seed for authentification
SP 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): 0.20080995559692383 (<type 'float'>)
Expectation (Timeout for authentification): 0.2 <= result <= 0.22000000000000003
```

Success	Timeout for authentification is correct (Content 0.5016460418701172 in [0.5 ... 0.55] and Type is <type 'float'>).

```
SP server: Requesting seed for authentification
SP 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): 0.5016460418701172 (<type 'float'>)
Expectation (Timeout for authentification): 0.5 <= result <= 0.55
```

Success	Timeout for send method is correct (Content 0.20120906829833984 in [0.2 ... 0.22000000000000003] and Type is <type 'float'>).

```
SP server: TIMEOUT (0.2s): Requested data (service_id: 30; data_id: 0) not in buffer.
Result (Timeout for send method): 0.20120906829833984 (<type 'float'>)
Expectation (Timeout for send method): 0.2 <= result <= 0.22000000000000003
```

Success	Timeout for send method is correct (Content 0.5018620491027832 in [0.5 ... 0.55] and Type is <type 'float'>).

```
SP server: TIMEOUT (0.5s): Requested data (service_id: 30; data_id: 0) not in buffer.
Result (Timeout for send method): 0.5018620491027832 (<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).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{u'test': u'test'}"
SP server: Received message with no registered callback. Sending negative response.
SP 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
SP server: RX <- status: 1, service_id: 11, data_id: 45054, data: "None"
SP server: Received message has a peculiar status: Request has no callback. Data buffered.
SP 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'>).

```
SP 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'>).

```
SP 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).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Requesting seed for authentification
SP 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
SP server: RX <- status: 0, service_id: 1, data_id: 0, data: "None"
SP server: Executing callback __authentificate_create_seed__ to process received data
SP server: Got seed request, sending seed for authentification
SP server: TX -> status: 0, service_id: 2, data_id: 0, data:
↪  "'3394f16e89fecd5e5e874d2c49cef180f654cf347db2fbe03998e2166d1f6f54'"
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 33 33 39 34 66 31 36 65 38 39 66 65 63 64 35
↪  65 35 65 38 37 34 64 32 63 34 39 63 65 66 31 38 30 66 36 35 34 63 66 33 34 37 64 62 32 66
↪  62 65 30 33 39 39 38 65 32 31 36 36 64 31 66 36 66 35 34 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d c1 c7 ea 60
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 33 33 39 34 66 31 36 65 38 39 66 65 63 64
↪  35 65 35 65 38 37 34 64 32 63 34 39 63 65 66 31 38 30 66 36 35 34 63 66 33 34 37 64 62 32
↪  66 62 65 30 33 39 39 38 65 32 31 36 36 64 31 66 36 66 35 34 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d c1 c7 ea 60
SP server: RX <- status: 0, service_id: 2, data_id: 0, data:
↪  "u'3394f16e89fecd5e5e874d2c49cef180f654cf347db2fbe03998e2166d1f6f54'"
SP server: Executing callback __authentificate_create_key__ to process received data
SP server: Got seed, sending key for authentification
SP server: TX -> status: 0, service_id: 3, data_id: 0, data:
↪  "'73d0ef4f8c1e50709490dad69d9517ff4e486ab60f5762ebf17f733e4f74dcda5fc2660ec58bdab64ffaa9
↪  312e50911033632670d5093e675d19158b2ca58330'"
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 37 33 64 30 65 66 34 66 38 63 31 65 35 30 37
↪  30 39 34 39 30 64 61 64 36 39 64 39 35 31 37 66 66 34 65 34 38 36 61 62 36 30 66 35 37 36
↪  32 65 62 66 31 37 66 37 33 33 65 34 66 37 34 64 63 64 61 35 66 63 32 36 36 30 65 63 35 38
↪  62 64 61 62 36 34 66 66 61 61 39 33 31 32 65 35 30 39 31 31 30 33 33 36 33 32 36 37 30 64
↪  35 30 39 33 65 36 37 35 64 31 39 31 35 38 62 32 63 61 35 38 33 33 30 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 30 7d ff c4 ce 07
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 37 33 64 30 65 66 34 66 38 63 31 65 35 30
↪  37 30 39 34 39 30 64 61 64 36 39 64 39 35 31 37 66 66 34 65 34 38 36 61 62 36 30 66 35 37
↪  36 32 65 62 66 31 37 66 37 33 33 65 34 66 37 34 64 63 64 61 35 66 63 32 36 36 30 65 63 35
↪  38 62 64 61 62 36 34 66 66 61 61 39 33 31 32 65 35 30 39 31 31 30 33 33 36 33 32 36 37 30
↪  64 35 30 39 33 65 36 37 35 64 31 39 31 35 38 62 32 63 61 35 38 33 33 30 22 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 30 7d ff c4 ce 07
SP server: RX <- status: 0, service_id: 3, data_id: 0, data:
↪  "u'73d0ef4f8c1e50709490dad69d9517ff4e486ab60f5762ebf17f733e4f74dcda5fc2660ec58bdab64ffaa
↪  9312e50911033632670d5093e675d19158b2ca58330'"
SP server: Executing callback __authentificate_check_key__ to process received data
```

```
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'>).

```
SP 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'>).

```
SP 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).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP 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).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "None"
SP server: Executing callback response_data_method_fail to process received data
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 48879, data: "None"
SP server: Received message with no registered callback. Sending negative response.
SP 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
SP server: RX <- status: 1, service_id: 11, data_id: 48879, data: "None"
SP server: Received message has a peculiar status: Request has no callback. Data buffered.
SP 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'>).

```
SP 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'>).

```
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP 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'!
SP client: Cleaning up receive-buffer
SP client: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: TX -> status: 0, service_id: 5, data_id: 0, data: "'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
SP 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
SP client: RX <- status: 0, service_id: 5, data_id: 0, data: "u'ut_server_set_channel_name'"
SP client: Executing callback __channel_name_request__ to process received data
SP client: channel name is now 'ut_server_set_channel_name'
SP 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
SP server: RX <- status: 0, service_id: 6, data_id: 0, data: "None"
SP 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 <type 'str'>).

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

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

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

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.

```
SP server: Cleaning up receive-buffer
SP 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'!
SP client: Cleaning up receive-buffer
SP client: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP 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
SP 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
SP client: RX <- status: 0, service_id: 5, data_id: 0, data: "None"
SP client: Executing callback __channel_name_request__ to process received data
SP client: TX -> status: 0, service_id: 6, data_id: 0, data: "'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
SP server: RX <- status: 0, service_id: 6, data_id: 0, data: "u'ut_client_set_channel_name'"
SP server: Executing callback ut_response_callback to process received data
SP 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 <type 'str'>).

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

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

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

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.

```
SP server: Cleaning up receive-buffer
SP 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'!
SP client: Cleaning up receive-buffer
SP client: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: TX -> status: 0, service_id: 5, data_id: 0, data:
↪  "'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
SP 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
SP client: RX <- status: 0, service_id: 5, data_id: 0, data:
↪  "u'ut_server_and_client_set_channel_name'"
SP client: Executing callback __channel_name_request__ to process received data
SP client: overwriting user defined channel name from 'foo' to
↪  u'ut_server_and_client_set_channel_name'
SP 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
SP server: RX <- status: 0, service_id: 6, data_id: 0, data: "None"
SP 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 <type 'str'>).

```
Result (Channel name for server): 'ut_server_and_client_set_channel_name' (<type 'str'>)
```

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

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

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

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

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.

```
SP server: Cleaning up receive-buffer
SP 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'!
SP client: Cleaning up receive-buffer
SP client: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: TX -> status: 0, service_id: 5, data_id: 0, data:
↪  "'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
SP 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
SP client: RX <- status: 0, service_id: 5, data_id: 0, data:
↪  "u'ut_server_and_client_set_channel_name'"
SP client: Executing callback __channel_name_request__ to process received data
SP 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
SP server: RX <- status: 0, service_id: 6, data_id: 0, data: "None"
SP 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 <type 'str'>).

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

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

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

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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test': 'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test': 'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Requesting seed for authentification
SP 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
SP server: RX <- status: 0, service_id: 1, data_id: 0, data: "None"
SP server: Executing callback __authentificate_create_seed__ to process received data
SP server: Got seed request, sending seed for authentification
SP server: TX -> status: 0, service_id: 2, data_id: 0, data:
↪  "'20c0a07412777f13f1ae039c95f678598ff4ac31cd106b762df43c9c183171d9'"
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 32
↪  30 63 30 61 30 37 34 31 32 37 37 37 66 31 33 66 31 61 65 30 33 39 63 39 35 66 36 37 38 35
↪  39 38 66 66 34 61 63 33 31 63 64 31 30 36 62 37 36 32 64 66 34 33 63 39 63 31 38 33 31 37
↪  31 64 39 22 7d d5 b1 e4 80
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
↪  32 30 63 30 61 30 37 34 31 32 37 37 37 66 31 33 66 31 61 65 30 33 39 63 39 35 66 36 37 38
↪  35 39 38 66 66 34 61 63 33 31 63 64 31 30 36 62 37 36 32 64 66 34 33 63 39 63 31 38 33 31
↪  37 31 64 39 22 7d d5 b1 e4 80
SP server: RX <- status: 0, service_id: 2, data_id: 0, data:
↪  "'20c0a07412777f13f1ae039c95f678598ff4ac31cd106b762df43c9c183171d9'"
SP server: Executing callback __authentificate_create_key__ to process received data
SP server: Got seed, sending key for authentification
SP server: TX -> status: 0, service_id: 3, data_id: 0, data:
↪  "'1561c9c70281318df82bf7e6deb57bd5be9eab1af3ce8e4d326381bfc092ba701a5af7ed0ae3c7a80176b6
↪  c4863366b1178435ae7c95160278fe39e2ef4b9c46'"
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
↪  35 36 31 63 39 63 37 30 32 38 31 33 31 38 64 66 38 32 62 66 37 65 36 64 65 62 35 37 62 64
↪  35 62 65 39 65 61 62 31 61 66 33 63 65 38 65 34 64 33 32 36 33 38 31 62 66 63 30 39 32 62
↪  61 37 30 31 61 35 61 66 37 65 64 30 61 65 33 63 37 61 38 30 31 37 36 62 36 63 34 38 36 33
↪  33 36 36 62 31 31 37 38 34 33 35 61 65 37 63 39 35 31 36 30 32 37 38 66 65 33 39 65 32 65
↪  66 34 62 39 63 34 36 22 7d 35 c5 3a 5e
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 35 36 31 63 39 63 37 30 32 38 31 33 31 38 64 66 38 32 62 66 37 65 36 64 65 62 35 37 62
↪  64 35 62 65 39 65 61 62 31 61 66 33 63 65 38 65 34 64 33 32 36 33 38 31 62 66 63 30 39 32
↪  62 61 37 30 31 61 35 61 66 37 65 64 30 61 65 33 63 37 61 38 30 31 37 36 62 36 63 34 38 36
↪  33 33 36 36 62 31 31 37 38 34 33 35 61 65 37 63 39 35 31 36 30 32 37 38 66 65 33 39 65 32
↪  65 66 34 62 39 63 34 36 22 7d 35 c5 3a 5e
SP server: RX <- status: 0, service_id: 3, data_id: 0, data:
↪  "'1561c9c70281318df82bf7e6deb57bd5be9eab1af3ce8e4d326381bfc092ba701a5af7ed0ae3c7a80176b6
↪  c4863366b1178435ae7c95160278fe39e2ef4b9c46'"
SP server: Executing callback __authentificate_check_key__ to process received data
```

```
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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 48879, data: "{'test': 'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 48879, data: "{'test': 'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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 .

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 48879, data: "{'test': 'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 48879, data: "[1, 3, 's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test': 'test'}"
SP server: Executing callback response_data_method_2 to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test': 'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
SP server: Received message has a peculiar status: Operation not permitted
SP server: Message data is stored in buffer and is now ready to be retrieved by receive
↪  method
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test': 'test'}"
SP server: Executing callback response_data_method to process received data
SP 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
SP server: RX <- status: 5, service_id: 11, data_id: 45054, data: "[1, 3, 's']"
SP server: Received message has a peculiar status: Operation not permitted
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP 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'>).

```
SP 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'>).

```
SP 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 0.20149755477905273 in [0.2 ... 0.22000000000000003] and Type is <class 'float'>).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Requesting seed for authentification
SP 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): 0.20149755477905273 (<class 'float'>)
Expectation (Timeout for authentification): 0.2 <= result <= 0.22000000000000003
```

Success	Timeout for authentification is correct (Content 0.5021364688873291 in [0.5 ... 0.55] and Type is <class 'float'>).

```
SP server: Requesting seed for authentification
SP 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): 0.5021364688873291 (<class 'float'>)
Expectation (Timeout for authentification): 0.5 <= result <= 0.55
```

Success	Timeout for send method is correct (Content 0.20085549354553223 in [0.2 ... 0.22000000000000003] and Type is <class 'float'>).

```
SP server: TIMEOUT (0.2s): Requested data (service_id: 30; data_id: 0) not in buffer.
Result (Timeout for send method): 0.20085549354553223 (<class 'float'>)
Expectation (Timeout for send method): 0.2 <= result <= 0.22000000000000003
```

Success	Timeout for send method is correct (Content 0.5018312931060791 in [0.5 ... 0.55] and Type is <class 'float'>).

```
SP server: TIMEOUT (0.5s): Requested data (service_id: 30; data_id: 0) not in buffer.
Result (Timeout for send method): 0.5018312931060791 (<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).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "{'test': 'test'}"
SP server: Received message with no registered callback. Sending negative response.
SP 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
SP server: RX <- status: 1, service_id: 11, data_id: 45054, data: "None"
SP server: Received message has a peculiar status: Request has no callback. Data buffered.
SP 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'>).

```
SP 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'>).

```
SP 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).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Requesting seed for authentification
SP 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
SP server: RX <- status: 0, service_id: 1, data_id: 0, data: "None"
SP server: Executing callback __authentificate_create_seed__ to process received data
SP server: Got seed request, sending seed for authentification
SP server: TX -> status: 0, service_id: 2, data_id: 0, data:
↪  "'3435b2d76e662ed74d4c53b5f9fc840b079a402602fd1813c078d5f8a3f22992'"
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 33
↪  34 33 35 62 32 64 37 36 65 36 36 32 65 64 37 34 64 34 63 35 33 62 35 66 39 66 63 38 34 30
↪  62 30 37 39 61 34 30 32 36 30 32 66 64 31 38 31 33 63 30 37 38 64 35 66 38 61 33 66 32 32
↪  39 39 32 22 7d 05 ef eb 79
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
↪  33 34 33 35 62 32 64 37 36 65 36 36 32 65 64 37 34 64 34 63 35 33 62 35 66 39 66 63 38 34
↪  30 62 30 37 39 61 34 30 32 36 30 32 66 64 31 38 31 33 63 30 37 38 64 35 66 38 61 33 66 32
↪  32 39 39 32 22 7d 05 ef eb 79
SP server: RX <- status: 0, service_id: 2, data_id: 0, data:
↪  "'3435b2d76e662ed74d4c53b5f9fc840b079a402602fd1813c078d5f8a3f22992'"
SP server: Executing callback __authentificate_create_key__ to process received data
SP server: Got seed, sending key for authentification
SP server: TX -> status: 0, service_id: 3, data_id: 0, data:
↪  "'989e9f888834c797745607d3f35a2355b3dbca31e05ec51e5147f54b32959ee93a06276b54b7fc8cff09f3
↪  72fc4eb356a960ecbc43298a83c46eeab1d2e939d5'"
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 39
↪  38 39 65 39 66 38 38 38 38 33 34 63 37 39 37 37 34 35 36 30 37 64 33 66 33 35 61 32 33 35
↪  35 62 33 64 62 63 61 33 31 65 30 35 65 63 35 31 65 35 31 34 37 66 35 34 62 33 32 39 35 39
↪  65 65 39 33 61 30 36 32 37 36 62 35 34 62 37 66 63 38 63 66 66 30 39 66 33 37 32 66 63 34
↪  65 62 33 35 36 61 39 36 30 65 63 62 63 34 33 32 39 38 61 38 33 63 34 36 65 65 61 62 31 64
↪  32 65 39 33 39 64 35 22 7d 4c 92 b1 60
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
↪  39 38 39 65 39 66 38 38 38 38 33 34 63 37 39 37 37 34 35 36 30 37 64 33 66 33 35 61 32 33
↪  35 35 62 33 64 62 63 61 33 31 65 30 35 65 63 35 31 65 35 31 34 37 66 35 34 62 33 32 39 35
↪  39 65 65 39 33 61 30 36 32 37 36 62 35 34 62 37 66 63 38 63 66 66 30 39 66 33 37 32 66 63
↪  34 65 62 33 35 36 61 39 36 30 65 63 62 63 34 33 32 39 38 61 38 33 63 34 36 65 65 61 62 31
↪  64 32 65 39 33 39 64 35 22 7d 4c 92 b1 60
SP server: RX <- status: 0, service_id: 3, data_id: 0, data:
↪  "'989e9f888834c797745607d3f35a2355b3dbca31e05ec51e5147f54b32959ee93a06276b54b7fc8cff09f3
↪  72fc4eb356a960ecbc43298a83c46eeab1d2e939d5'"
SP server: Executing callback __authentificate_check_key__ to process received data
```

```
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'>).

```
SP 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'>).

```
SP 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).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP 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).

```
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP server: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 45054, data: "None"
SP server: Executing callback response_data_method_fail to process received data
SP 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
SP server: RX <- status: 0, service_id: 10, data_id: 48879, data: "None"
SP server: Received message with no registered callback. Sending negative response.
SP 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
SP server: RX <- status: 1, service_id: 11, data_id: 48879, data: "None"
SP server: Received message has a peculiar status: Request has no callback. Data buffered.
SP 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'>).

```
SP 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'>).

```
SP 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'>).

```
SP 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'>).

```
SP 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.

```
SP server: Cleaning up receive-buffer
SP 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'!
SP client: Cleaning up receive-buffer
SP client: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP 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
SP 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
SP client: RX <- status: 0, service_id: 5, data_id: 0, data: "'ut_server_set_channel_name'"
SP client: Executing callback __channel_name_request__ to process received data
SP client: channel name is now 'ut_server_set_channel_name'
SP 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
SP server: RX <- status: 0, service_id: 6, data_id: 0, data: "None"
SP 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.

```
SP server: Cleaning up receive-buffer
SP 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'!
SP client: Cleaning up receive-buffer
SP client: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP 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
SP 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
SP client: RX <- status: 0, service_id: 5, data_id: 0, data: "None"
SP client: Executing callback __channel_name_request__ to process received data
SP 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
SP server: RX <- status: 0, service_id: 6, data_id: 0, data: "'ut_client_set_channel_name'"
SP server: Executing callback ut_response_callback to process received data
SP 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.

```
SP server: Cleaning up receive-buffer
SP 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'!
SP client: Cleaning up receive-buffer
SP client: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP 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
SP 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
SP client: RX <- status: 0, service_id: 5, data_id: 0, data:
↪  "'ut_server_and_client_set_channel_name'"
SP client: Executing callback __channel_name_request__ to process received data
SP client: overwriting user defined channel name from 'foo' to
↪  'ut_server_and_client_set_channel_name'
SP 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
SP server: RX <- status: 0, service_id: 6, data_id: 0, data: "None"
SP 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.

```
SP server: Cleaning up receive-buffer
SP 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'!
SP client: Cleaning up receive-buffer
SP client: Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT
SP server: Cleaning up receive-buffer
SP 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
SP 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
SP client: RX <- status: 0, service_id: 5, data_id: 0, data:
↪  "'ut_server_and_client_set_channel_name'"
SP client: Executing callback __channel_name_request__ to process received data
SP 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
SP server: RX <- status: 0, service_id: 6, data_id: 0, data: "None"
SP 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):
    DEFAULT_CHANNEL_NAME = 'all_others'

    def __init__(self, channel_name):
        self.init_channel_name(channel_name)
        dict.__init__(self)

    def init_channel_name(self, channel_name):
        if channel_name is None:
            self.logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__ + '.' + self.DEFAULT_CHANNEL_NAME)
        else:
            self.logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__ + '.' + channel_name)

    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):
            self.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'

```

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

```
160
161     SID_RESPONSE_DICT = {SID_AUTH_SEED_REQUEST: SID_AUTH_KEY_REQUEST,
162                          SID_AUTH_KEY_REQUEST: SID_AUTH_KEY_CHECK_REQUEST,
163                          SID_AUTH_KEY_CHECK_REQUEST: SID_AUTH_KEY_CHECK_RESPONSE,
164                          SID_CHANNEL_NAME_REQUEST: SID_CHANNEL_NAME_RESPONSE,
165                          SID_READ_REQUEST: SID_READ_RESPONSE,
166                          SID_WRITE_REQUEST: SID_WRITE_RESPONSE,
167                          SID_EXECUTE_REQUEST: SID_EXECUTE_RESPONSE}
168
169     SID_AUTH_LIST = [
170         SID_AUTH_SEED_REQUEST,
171         SID_AUTH_KEY_REQUEST,
172         SID_AUTH_KEY_CHECK_REQUEST,
173         SID_AUTH_KEY_CHECK_RESPONSE,
174         SID_CHANNEL_NAME_REQUEST,
175         SID_CHANNEL_NAME_RESPONSE
176     ]
177
178     STATUS_OKAY = 0
179     STATUS_BUFFERING_UNHANDLED_REQUEST = 1
180     STATUS_AUTH_REQUIRED = 2
181     STATUS_SERVICE_OR_DATA_UNKNOWN = 3
182     STATUS_CHECKSUM_ERROR = 4
183     STATUS_OPERATION_NOT_PERMITTED = 5
184     STATUS_NAMES = {STATUS_OKAY: 'Okay',
185                     STATUS_BUFFERING_UNHANDLED_REQUEST: 'Request has no callback. Data buffered.'
    ,
186                     STATUS_AUTH_REQUIRED: 'Authentification required',
187                     STATUS_SERVICE_OR_DATA_UNKNOWN: 'Service or Data unknown',
188                     STATUS_CHECKSUM_ERROR: 'Checksum Error',
189                     STATUS_OPERATION_NOT_PERMITTED: 'Operation not permitted'}
190
191     AUTH_STATE_UNKNOWN_CLIENT = 0
192     AUTH_STATE_SEED_REQUESTED = 1
193     AUTH_STATE_SEED_TRANSFERRED = 2
194     AUTH_STATE_KEY_TRANSFERRED = 3
195     AUTH_STATE_TRUSTED_CLIENT = 4
196     AUTH_STATUS_NAMES = {AUTH_STATE_UNKNOWN_CLIENT: 'Unknown Client',
197                          AUTH_STATE_SEED_REQUESTED: 'Seed was requested',
198                          AUTH_STATE_SEED_TRANSFERRED: 'Seed has been sent',
199                          AUTH_STATE_KEY_TRANSFERRED: 'Key has been sent',
200                          AUTH_STATE_TRUSTED_CLIENT: 'Trusted Client'}
201
202     def __init__(self, comm_instance, secret=None, auto_auth=False, channel_name=None):
203         self.__comm_inst__ = comm_instance
204         self.__secret__ = secret
205         self.__auto_auth__ = auto_auth
206         #
207         self.__callbacks__ = callback_storage(channel_name)
208         self.__init_channel_name__(channel_name)
209         #
210         self.__clean_receive_buffer__()
211         self.__callbacks__.add(self.SID_AUTH_SEED_REQUEST, 0, self.__authentificate_create_seed__
    )
212         self.__callbacks__.add(self.SID_AUTH_KEY_REQUEST, 0, self.__authentificate_create_key__)
213         self.__callbacks__.add(self.SID_AUTH_KEY_CHECK_REQUEST, 0, self.
    __authentificate_check_key__)
214         self.__callbacks__.add(self.SID_AUTH_KEY_CHECK_RESPONSE, 0, self.
    __authentificate_process_feedback__)
215         self.__callbacks__.add(self.SID_CHANNEL_NAME_REQUEST, 0, self.__channel_name_request__)
216         self.__callbacks__.add(self.SID_CHANNEL_NAME_RESPONSE, 0, self.__channel_name_response__)
217         self.__authentification_state_reset__()
```

```
218         self.__seed__ = None
219         self.__comm_inst__.register_callback(self.__data_available_callback__)
220         self.__comm_inst__.register_connect_callback(self.__connection_established__)
221         self.__comm_inst__.register_disconnect_callback(self.__authentification_state_reset__)
222 
223     def __init_channel_name__(self, channel_name):
224         self.__comm_inst__.init_channel_name(channel_name)
225         self.__callbacks__.init_channel_name(channel_name)
226         if channel_name is None:
227             self.logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__ + '.' + self.
    DEFAULT_CHANNEL_NAME)
228         else:
229             self.logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__ + '.' +
    channel_name)
230 
231     @property
232     def __channel_name__(self):
233         cn = self.logger.name.split('.')[-1]
234         if cn != self.DEFAULT_CHANNEL_NAME:
235             return cn
236 
237     def __log_prefix__(self):
238         return ' SP client:' if self.__comm_inst__.IS_CLIENT else ' SP server:'
239 
240     def connected(self):
241         return self.__comm_inst__.is_connected()
242 
243     def connection_established(self):
244         return self.connected() and (self.__secret__ is None or self.check_authentification_state
    ())
245 
246     def reconnect(self):
247         return self.__comm_inst__.reconnect()
248 
249     def __connection_established__(self):
250         self.__clean_receive_buffer__()
251         if not self.__comm_inst__.IS_CLIENT:
252             self.send(self.SID_CHANNEL_NAME_REQUEST, 0, self.__channel_name__)
253         if self.__auto_auth__ and self.__comm_inst__.IS_CLIENT and self.__secret__ is not None:
254             self.authentificate()
255 
256     def __channel_name_request__(self, msg):
257         data = msg.get_data()
258         if data is None:
259             return self.STATUS_OKAY, self.__channel_name__
260         else:
261             prev_channel_name = self.__channel_name__
262             self.__init_channel_name__(data)
263             if prev_channel_name is not None and prev_channel_name != data:
264                 self.logger.warning('%s overwriting user defined channel name from %s to %s',
    self.__log_prefix__(), repr(prev_channel_name), repr(data))
265             elif prev_channel_name is None:
266                 self.logger.info('%s channel name is now %s', self.__log_prefix__(), repr(self.
    __channel_name__))
267             return self.STATUS_OKAY, None
268 
269     def __channel_name_response__(self, msg):
270         data = msg.get_data()
271         if self.__channel_name__ is None and data is not None:
272             self.__init_channel_name__(data)
273             self.logger.info('%s channel name is now %s', self.__log_prefix__(), repr(self.
    __channel_name__))
274         return self.STATUS_OKAY, None
```

```
275
276     def __authentification_state_reset__(self):
277         self.logger.info("%s Resetting authentification state to AUTH_STATE_UNKNOWN_CLIENT", self
    .__log_prefix__())
278         self.__authentification_state__ = self.AUTH_STATE_UNKNOWN_CLIENT
279
280     def __analyse_frame__(self, frame):
281         status, service_id, data_id = struct.unpack('>III', frame[0:12])
282         if sys.version_info >= (3, 0):
283             data = json.loads(frame[12:-1].decode('utf-8'))
284         else:
285             data = json.loads(frame[12:-1])
286         return self.__mk_msg__(status, service_id, data_id, data)
287
288     def __build_frame__(self, service_id, data_id, data, status=STATUS_OKAY):
289         frame = struct.pack('>III', status, service_id, data_id)
290         if sys.version_info >= (3, 0):
291             frame += bytes(json.dumps(data), 'utf-8')
292             frame += self.__calc_chksum__(frame)
293         else:
294             frame += json.dumps(data)
295             frame += self.__calc_chksum__(frame)
296         return frame
297
298     def __calc_chksum__(self, raw_data):
299         chksum = 0
300         for b in raw_data:
301             if sys.version_info >= (3, 0):
302                 chksum ^= b
303             else:
304                 chksum ^= ord(b)
305         if sys.version_info >= (3, 0):
306             return bytes([chksum])
307         else:
308             return chr(chksum)
309
310     def __check_frame_checksum__(self, frame):
311         return self.__calc_chksum__(frame[:-1]) == frame[-1:]
312
313     def __data_available_callback__(self, comm_inst):
314         frame = comm_inst.receive()
315         if not self.__check_frame_checksum__(frame):
316             self.logger.warning("%s Received message has a wrong checksum and will be ignored: %s
    .", self.__log_prefix__(), stringtools.hexlify(frame))
317         else:
318             msg = self.__analyse_frame__(frame)
319             self.logger.info(
320                 '%s RX <- status: %s, service_id: %s, data_id: %s, data: "%s"',
321                 self.__log_prefix__(),
322                 repr(msg.get_status()),
323                 repr(msg.get_service_id()),
324                 repr(msg.get_data_id()),
325                 repr(msg.get_data())
326             )
327             callback, args, kwargs = self.__callbacks__.get(msg.get_service_id(), msg.get_data_id
    ())
328             if msg.get_service_id() in self.SID_RESPONSE_DICT.keys():
329                 #
330                 # REQUEST RECEIVED
331                 #
```

```
332                 if self.__secret__ is not None and not self.check_authentification_state() and msg.get_service_id() not in self.SID_AUTH_LIST:
333                     status = self.STATUS_AUTH_REQUIRED
334                     data = None
335                     self.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!'))
336                 elif callback is None:
337                     self.logger.warning("%s Received message with no registered callback. Sending negative response.", self.__log_prefix__())
338                     status = self.STATUS_BUFFERING_UNHANDLED_REQUEST
339                     data = None
340                 else:
341                     try:
342                         self.logger.debug("%s Executing callback %s to process received data", self.__log_prefix__(), callback.__name__)
343                         status, data = callback(msg, *args, **kwargs)
344                     except TypeError:
345                         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())))
346                 self.send(self.SID_RESPONSE_DICT[msg.get_service_id()], msg.get_data_id(), data, status=status)
347             else:
348                 #
349                 # RESPONSE RECEIVED
350                 #
351                 if msg.get_status() not in [self.STATUS_OKAY]:
352                     self.logger.warning("%s Received message has a peculiar status: %s", self.__log_prefix__(), self.STATUS_NAMES.get(msg.get_status(), 'Unknown status response!'))
353                 if callback is None:
354                     status = self.STATUS_OKAY
355                     data = None
356                     self.__buffer_received_data__(msg)
357                 else:
358                     try:
359                         self.logger.debug("%s Executing callback %s to process received data", self.__log_prefix__(), callback.__name__)
360                         status, data = callback(msg, *args, **kwargs)
361                     except TypeError:
362                         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())))
363 
364     def __buffer_received_data__(self, msg):
365         if not msg.get_service_id() in self.__msg_buffer__:
366             self.__msg_buffer__[msg.get_service_id()] = {}
367         if not msg.get_data_id() in self.__msg_buffer__[msg.get_service_id()]:
368             self.__msg_buffer__[msg.get_service_id()][msg.get_data_id()] = []
369         self.__msg_buffer__[msg.get_service_id()][msg.get_data_id()].append(msg)
370         self.logger.debug("%s Message data is stored in buffer and is now ready to be retrieved by receive method", self.__log_prefix__())
371 
372     def __clean_receive_buffer__(self):
373         self.logger.debug("%s Cleaning up receive-buffer", self.__log_prefix__())
374         self.__msg_buffer__ = {}
375 
376     def receive(self, service_id, data_id, timeout=1):
377         data = None
378         cnt = 0
379         while data is None and cnt < timeout * 10:
```

```
380             try:
381                 data = self.__msg_buffer__.get(service_id, {}).get(data_id, []).pop(0)
382             except IndexError:
383                 data = None
384             cnt += 1
385             time.sleep(0.1)
386         if data is None and cnt >= timeout * 10:
387             self.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))
388         return data
389 
390     def __mk_msg__(self, status, service_id, data_id, data):
391         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})
392 
393     def send(self, service_id, data_id, data, status=STATUS_OKAY, timeout=2, log_lvl=logging.INFO
    ):
394         """
395         :param service_id: The Service-ID for the message. See class definitions starting with ``
    SERVICE_``.
396         :type service_id: int
397         :param data_id: The Data-ID for the message.
398         :type data_id: int
399         :param data: The data to be transfered. The data needs to be json compatible.
400         :type data: str
401         :param status: The Status for the message. All requests should have ``STATUS_OKAY``.
402         :type status: int
403         :param timeout: The timeout for sending data (e.g. time to establish new connection).
404         :type timeout: float
405         :param rx_log_lvl: The log level to log outgoing TX-data
406         :type rx_log_lvl: int
407         :return: True if data had been sent, otherwise False.
408         :rtype: bool
409 
410         This methods sends out a message with the given content.
411         """
412         self.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))
413         return self.__comm_inst__.send(self.__build_frame__(service_id, data_id, data, status),
    timeout=timeout, log_lvl=logging.DEBUG)
414 
415     def register_callback(self, service_id, data_id, callback, *args, **kwargs):
416         """
417         :param service_id: The Service-ID for the message. See class definitions starting with ``
    SID_``.
418         :type service_id: int
419         :param data_id: The Data-ID for the message.
420         :type data_id: int
421         :returns: True, if registration was successfull; False, if registration failed (e.g.
    existance of a callback for this configuration)
422         :rtype: bool
423 
424         This method registers a callback for the given parameters. Givin ``None`` means, that all
     Service-IDs or all Data-IDs are used.
425         If a message hitting these parameters has been received, the callback will be executed.
426 
427         .. note:: The :func:`callback` is priorised in the following order:
428 
429             * Callbacks with defined Service-ID and Data-ID.
430             * Callbacks with a defined Data-ID.
431             * Callbacks with a defined Service-ID.
```

```
432             * Unspecific Callbacks
433
434         .. note:: The :func:`callback` is executed with these arguments:
435
436             :param msg: A :class:`dict` containing all message information.
437             :returns: status (see class definition starting with ``STATUS_``), response_data (
    JSON compatible object)
438         """
439         self.__callbacks__.add(service_id, data_id, callback, *args, **kwargs)
440
441     def authentificate(self, timeout=2):
442         """
443         :param timeout: The timeout for the authentification (requesting seed, sending key and
    getting authentification_feedback).
444         :type timeout: float
445         :returns: True, if authentification was successfull; False, if not.
446         :rtype: bool
447
448         This method authetificates the client at the server.
449
450         .. note:: An authentification will only processed, if a secret had been given on
    initialisation.
451
452         .. note:: Client and Server needs to use the same secret.
453         """
454         if self.__secret__ is not None:
455             self.__authentification_state__ = self.AUTH_STATE_SEED_REQUESTED
456             self.logger.info("%s Requesting seed for authentification", self.__log_prefix__())
457             self.send(self.SID_AUTH_SEED_REQUEST, 0, None)
458             cnt = 0
459             while cnt < timeout * 10:
460                 time.sleep(0.1)
461                 if self.__authentification_state__ == self.AUTH_STATE_TRUSTED_CLIENT:
462                     return True
463                 elif self.__authentification_state__ == self.AUTH_STATE_UNKNOWN_CLIENT:
464                     break
465                 cnt += 1
466         return False
467
468     def check_authentification_state(self):
469         """
470         :return: True, if authentification state is okay, otherwise False
471         :rtype: bool
472         """
473         return self.__authentification_state__ == self.AUTH_STATE_TRUSTED_CLIENT
474
475     def __authentificate_salt_and_hash__(self, seed):
476         if sys.version_info >= (3, 0):
477             return hashlib.sha512(bytes(seed, 'utf-8') + self.__secret__).hexdigest()
478         else:
479             return hashlib.sha512(seed.encode('utf-8') + self.__secret__.encode('utf-8')).
    hexdigest()
480
481     def __authentificate_create_seed__(self, msg):
482         self.logger.info("%s Got seed request, sending seed for authentification", self.
    __log_prefix__())
483         self.__authentification_state__ = self.AUTH_STATE_SEED_TRANSFERRED
484         if sys.version_info >= (3, 0):
485             self.__seed__ = binascii.hexlify(os.urandom(32)).decode('utf-8')
486         else:
487             self.__seed__ = binascii.hexlify(os.urandom(32))
488         return self.STATUS_OKAY, self.__seed__
```

```
489
490     def __authentificate_create_key__(self, msg):
491         self.logger.info("%s Got seed, sending key for authentification", self.__log_prefix__())
492         self.__authentification_state__ = self.AUTH_STATE_KEY_TRANSFERRED
493         seed = msg.get_data()
494         key = self.__authentificate_salt_and_hash__(seed)
495         return self.STATUS_OKAY, key
496
497     def __authentificate_check_key__(self, msg):
498         key = msg.get_data()
499         if key == self.__authentificate_salt_and_hash__(self.__seed__):
500             self.__authentification_state__ = self.AUTH_STATE_TRUSTED_CLIENT
501             self.logger.info("%s Got correct key, sending positive authentification feedback",
    self.__log_prefix__())
502             return self.STATUS_OKAY, True
503         else:
504             self.__authentification_state__ = self.AUTH_STATE_UNKNOWN_CLIENT
505             self.logger.info("%s Got incorrect key, sending negative authentification feedback",
    self.__log_prefix__())
506             return self.STATUS_OKAY, False
507
508     def __authentificate_process_feedback__(self, msg):
509         feedback = msg.get_data()
510         if feedback:
511             self.__authentification_state__ = self.AUTH_STATE_TRUSTED_CLIENT
512             self.logger.info("%s Got positive authentification feedback", self.__log_prefix__())
513         else:
514             self.__authentification_state__ = self.AUTH_STATE_UNKNOWN_CLIENT
515             self.logger.warning("%s Got negative authentification feedback", self.__log_prefix__
    ())
516         return self.STATUS_OKAY, None
517
518
519 class pure_json_protocol(struct_json_protocol):
520     """
521     :param comm_instance: a communication instance supportin at least these functions: :func:`
    register_callback`, :func:`register_disconnect_callback`, :func:`send`.
522     :type comm_instance: instance
523     :param secret: A secret (e.g. created by ``binascii.hexlify(os.urandom(24))``).
524     :type secret: str
525
526     This communication protocol supports to transfer a Service-ID, Data-ID and Data.
527
528     **Example:**
529
530     Server:
531
532     .. literalinclude:: ../../socket_protocol/_examples_/
    socket_protocol__pure_json_protocol_server.py
533
534     .. literalinclude:: ../../socket_protocol/_examples_/
    socket_protocol__pure_json_protocol_server.log
535
536
537     Client:
538
539     .. literalinclude:: ../../socket_protocol/_examples_/
    socket_protocol__pure_json_protocol_client.py
540
541     .. literalinclude:: ../../socket_protocol/_examples_/
    socket_protocol__pure_json_protocol_client.log
542     """
```

```
543     def __init__(self, *args, **kwargs):
544         struct_json_protocol.__init__(self, *args, **kwargs)
545 
546     def __build_frame__(self, service_id, data_id, data, status=struct_json_protocol.STATUS_OKAY)
        :
547         data_frame = json.dumps(self.__mk_msg__(status, service_id, data_id, data))
548         if sys.version_info >= (3, 0):
549             data_frame = bytes(data_frame, 'utf-8')
550         checksum = self.__calc_chksum__(data_frame)
551         return data_frame + checksum
552 
553     def __analyse_frame__(self, frame):
554         if sys.version_info >= (3, 0):
555             return data_storage(json.loads(frame[:-4].decode('utf-8')))
556         else:
557             return data_storage(json.loads(frame[:-4]))
558 
559     def __calc_chksum__(self, raw_data):
560         return struct.pack('>I', binascii.crc32(raw_data) & 0xffffffff)
561 
562     def __check_frame_checksum__(self, frame):
563         return self.__calc_chksum__(frame[:-4]) == frame[-4:]
```