# Unittest for `socket_protocol`

January 6, 2021

# Contents

# 1 Test Information

## 1.1 Test Candidate Information

The Module socket_protocol is designed for point to point communication for client-server issues. For more Information read the sphinx documentation.

| **Library Information** | |
|---|---|
| Name | socket_protocol |
| State | Released |
| Supported Interpreters | python2, python3 |
| Version | fdca73452afabd6d5a54327817639dce |
| **Dependencies** | |
| stringtools | 09b4d1c41b828c8d1ccb723fa1fd79a9 |

## 1.2 Unittest Information

| **Unittest Information** | |
|---|---|
| Version | 937ff4d1e2cef1eec3e4f65a89281dc2 |
| 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 | 5.4.0-59-generic (#65-Ubuntu SMP Thu Dec 10 12:01:51 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 | **22** |
| Number of successfull tests | **22** |
| Number of possibly failed tests | **0** |
| Number of failed tests | **0** |
| Executionlevel | Full Test (all defined tests) |
| Time consumption | 13.640s |

## 2.2 Test-Statistic for testrun with python 3.8.5 (final)

| | |
|---|---|
| Number of tests | **22** |
| Number of successfull tests | **22** |
| Number of possibly failed tests | **0** |
| Number of failed tests | **0** |
| Executionlevel | Full Test (all defined tests) |
| Time consumption | 13.588s |

## 2.3 Coverage Statistic

| Module- or Filename | Line-Coverage | Branch-Coverage |
|---|---|---|
| socket_protocol | 100.0% | 100.0% |
| socket_protocol.__init__.py | 100.0% | |

# 3 Tested Requirements

## 3.1 Message Object

A Message Object shall hold the following information for transmission.

### 3.1.1 Status

**Description**
The Status shall hold some general information (in most cases it is used by the responder). Examples: Okay, Service or Data unknown, Operation not permitted, Authentification required, ...

**Reason for the implementation**
Give the possibility to transfer additional status information (e.g. to explain negative responses).

**Fitcriterion**
A Status is part of the Message Object and it is holding the Status information.

**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: | 2021-01-06 22:48:56,876 |
| Finished-Time: | 2021-01-06 22:48:56,877 |
| Time-Consumption | 0.001s |

**Testsummary:**

| | |
|---|---|
| **Info** | Creating empty message object: {'status': None, 'service_id': None, 'data': None, 'data_id': None} |
| **Success** | status is part of the message object is correct ('status' is in the list or dict). |
| **Info** | Creating a maximum message object: {'status': 'S', 'service_id': 'SID', 'data': 'D', 'data_id': 'DID'} |
| **Success** | status is part of the message object is correct ('status' is in the list or dict). |
| **Success** | Content in message object for status is correct (Content 'S' and Type is <type 'str'>). |

**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: | 2021-01-06 22:49:11,386 |
| Finished-Time: | 2021-01-06 22:49:11,390 |
| Time-Consumption | 0.004s |

**Testsummary:**

| | |
|---|---|
| **Info** | Creating empty message object: {'data': None, 'data_id': None, 'service_id': None, 'status': None} |
| **Success** | status is part of the message object is correct ('status' is in the list or dict). |
| **Info** | Creating a maximum message object: {'data': 'D', 'data_id': 'DID', 'service_id': 'SID', 'status': 'S'} |
| **Success** | status is part of the message object is correct ('status' is in the list or dict). |
| **Success** | Content in message object for status is correct (Content 'S' and Type is <class 'str'>). |

### 3.1.2 Service-ID

**Description**

The Service-ID shall hold information about the type of the request / corresponding response. Examples: read request, write request, read response, write response, ...

**Reason for the implementation**

Give the requestor the possibility to use different types (Services) for a transfer.

**Fitcriterion**

A Service-ID is part of the Message Object and it is holding the Service-ID information.

**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: | 2021-01-06 22:48:56,877 |
| Finished-Time: | 2021-01-06 22:48:56,878 |
| Time-Consumption | 0.001s |

**Testsummary:**

| | |
|---|---|
| **Info** | Creating empty message object: {'status': None, 'service_id': None, 'data': None, 'data_id': None} |
| **Success** | service_id is part of the message object is correct ('service_id' is in the list or dict). |
| **Info** | Creating a maximum message object: {'status': 'S', 'service_id': 'SID', 'data': 'D', 'data_id': 'DID'} |
| **Success** | service_id is part of the message object is correct ('service_id' is in the list or dict). |
| **Success** | Content in message object for service_id is correct (Content 'SID' and Type is <type 'str'>). |

**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: | 2021-01-06 22:49:11,390 |
| Finished-Time: | 2021-01-06 22:49:11,392 |

| | |
|---|---|
| Time-Consumption | 0.002s |

**Testsummary:**

| | |
|---|---|
| **Info** | Creating empty message object: {'data': None, 'data_id': None, 'service_id': None, 'status': None} |
| **Success** | service_id is part of the message object is correct ('service_id' is in the list or dict). |
| **Info** | Creating a maximum message object: {'data': 'D', 'data_id': 'DID', 'service_id': 'SID', 'status': 'S'} |
| **Success** | service_id is part of the message object is correct ('service_id' is in the list or dict). |
| **Success** | Content in message object for service_id is correct (Content 'SID' and Type is <class 'str'>). |

### 3.1.3 Data-ID

**Description**

The Data-ID shall hold information to differtiate the data for a specific Service.

**Reason for the implementation**

Give the possibility to transfer different information for each Service.

**Fitcriterion**

A Data-ID is part of the Message Object and it is holding the Data-ID information.

**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: | 2021-01-06 22:48:56,878 |
| Finished-Time: | 2021-01-06 22:48:56,878 |
| Time-Consumption | 0.001s |

**Testsummary:**

| | |
|---|---|
| **Info** | Creating empty message object: {'status': None, 'service_id': None, 'data': None, 'data_id': None} |
| **Success** | data_id is part of the message object is correct ('data_id' is in the list or dict). |
| **Info** | Creating a maximum message object: {'status': 'S', 'service_id': 'SID', 'data': 'D', 'data_id': 'DID'} |
| **Success** | data_id is part of the message object is correct ('data_id' is in the list or dict). |
| **Success** | Content in message object for data_id is correct (Content 'DID' and Type is <type 'str'>). |

**Testresult**

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

| | |
|---|---|
| Testrun: | python 3.8.5 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (28) |
| Start-Time: | 2021-01-06 22:49:11,392 |

| | |
|---|---|
| Finished-Time: | 2021-01-06 22:49:11,394 |
| Time-Consumption | 0.001s |

**Testsummary:**

| | |
|---|---|
| **Info** | Creating empty message object: {'data': None, 'data_id': None, 'service_id': None, 'status': None} |
| **Success** | data_id is part of the message object is correct ('data_id' is in the list or dict). |
| **Info** | Creating a maximum message object: {'data': 'D', 'data_id': 'DID', 'service_id': 'SID', 'status': 'S'} |
| **Success** | data_id is part of the message object is correct ('data_id' is in the list or dict). |
| **Success** | Content in message object for data_id is correct (Content 'DID' and Type is <class 'str'>). |

### 3.1.4 Data

**Description**

The Data shall hold the data to be transfered. For the most requests not data is transmitted.

**Reason for the implementation**

Give the possibility to transfer Data.

**Fitcriterion**

Data is part of the Message Object and it is holding the Data information.

**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: | 2021-01-06 22:48:56,878 |
| Finished-Time: | 2021-01-06 22:48:56,879 |
| Time-Consumption | 0.001s |

**Testsummary:**

| | |
|---|---|
| **Info** | Creating empty message object: {'status': None, 'service_id': None, 'data': None, 'data_id': None} |
| **Success** | data is part of the message object is correct ('data' is in the list or dict). |
| **Info** | Creating a maximum message object: {'status': 'S', 'service_id': 'SID', 'data': 'D', 'data_id': 'DID'} |
| **Success** | data is part of the message object is correct ('data' is in the list or dict). |
| **Success** | Content in message object for data is correct (Content 'D' and Type is <type 'str'>). |

**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: | 2021-01-06 22:49:11,394 |
| Finished-Time: | 2021-01-06 22:49:11,395 |
| Time-Consumption | 0.001s |

**Testsummary:**

| | |
|---|---|
| **Info** | Creating empty message object: {'data': None, 'data_id': None, 'service_id': None, 'status': None} |
| **Success** | data is part of the message object is correct ('data' is in the list or dict). |
| **Info** | Creating a maximum message object: {'data': 'D', 'data_id': 'DID', 'service_id': 'SID', 'status': 'S'} |
| **Success** | data is part of the message object is correct ('data' is in the list or dict). |
| **Success** | Content in message object for data is correct (Content 'D' and Type is <class 'str'>). |

## 3.2 Communication

### 3.2.1 A full Message Object including the defined properties and data shall be transfered.

**Description**

Every Communication shall transfer a complete message with its content.

**Reason for the implementation**

See Reasons for every single information of the Message Object.

**Fitcriterion**

Send two different messages and compare the received message with each sent message.

**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 (33) |
| Start-Time: | 2021-01-06 22:48:56,879 |
| Finished-Time: | 2021-01-06 22:48:57,087 |
| Time-Consumption | 0.208s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content {u'status': 0, u'service_id': 17, u'data': u'msg1_data_to_be_transfered', u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on client side is correct (Content {u'status': 4, u'service_id': 17, u'data': u'msg2_data_to_be_transfered', u'data_id': 35} and Type is <class 'socket_protocol.data_storage'>). |

**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 (33) |
| Start-Time: | 2021-01-06 22:49:11,395 |
| Finished-Time: | 2021-01-06 22:49:11,606 |
| Time-Consumption | 0.211s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content {'data_id': 34, 'service_id': 17, 'status': 0, 'data': 'msg1_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on client side is correct (Content {'data_id': 35, 'service_id': 17, 'status': 4, 'data': 'msg2_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |

### 3.2.2 A checksumm shall ensure the correct transmition

**Description**

If the checksum does not fit to the checksum of the transferred data, the message will be ignored, because the complete content including the Service- and Data-ID is possibly corrupted.

**Reason for the implementation**

Ensure correct data transfer.

**Fitcriterion**

Corrupted message is not in the receive buffer after transmission.

**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 (34) |
| Start-Time: | 2021-01-06 22:48:57,088 |
| Finished-Time: | 2021-01-06 22:48:57,705 |
| Time-Consumption | 0.617s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Checksum Error → No message received by server is correct (Content None and Type is <type 'NoneType'>). |

| | |
|---|---|
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Checksum Error → No message received by client is correct (Content None and Type is <type 'NoneType'>). |

**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 (34) |
| Start-Time: | 2021-01-06 22:49:11,607 |
| Finished-Time: | 2021-01-06 22:49:12,219 |
| Time-Consumption | 0.612s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Checksum Error → No message received by server is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Checksum Error → No message received by client is correct (Content None and Type is <class 'NoneType'>). |

### 3.2.3 An authentification between server and client shall be possible including status feedback methods

**Description**

The Client shall have a method to initiate the authentification. In case that the server and the client do have identical secrets, the authentification shall be successfull.

**Reason for the implementation**

Message protection (e.g. for secure functions or data)

**Fitcriterion**

Check authentification method feedback (client) and authentification feedback (client and server), in case of differing and identical secrets.

**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 (35) |
| Start-Time: | 2021-01-06 22:48:57,706 |
| Finished-Time: | 2021-01-06 22:49:01,137 |

| | |
|---|---|
| Time-Consumption | 3.432s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | No secret set |
| **Info** | Performing Authentification |
| **Success** | Return Value of authentification method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Authentification state of server is correct (Content True and Type is <type 'bool'>). |
| **Success** | Authentification state of client is correct (Content True and Type is <type 'bool'>). |
| **Info** | Different secrets set |
| **Success** | Authentification state of server is correct (Content False and Type is <type 'bool'>). |
| **Success** | Authentification state of client is correct (Content False and Type is <type 'bool'>). |
| **Info** | Performing Authentification |
| **Success** | Return Value of authentification method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Authentification state of server is correct (Content False and Type is <type 'bool'>). |
| **Success** | Authentification state of client is correct (Content False and Type is <type 'bool'>). |
| **Info** | Identical secrets set |
| **Info** | Performing Authentification |
| **Success** | Return Value of authentification method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Authentification state of server is correct (Content True and Type is <type 'bool'>). |
| **Success** | Authentification state of client is correct (Content True and Type is <type 'bool'>). |
| **Info** | Corrupting the authentification mechanism |
| **Info** | Performing Authentification |
| **Success** | Return Value of authentification method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Authentification state of server is correct (Content False and Type is <type 'bool'>). |
| **Success** | Authentification state of client is correct (Content False and Type is <type 'bool'>). |

**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 (35) |
| Start-Time: | 2021-01-06 22:49:12,220 |
| Finished-Time: | 2021-01-06 22:49:15,648 |
| Time-Consumption | 3.429s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | No secret set |
| **Info** | Performing Authentification |
| **Success** | Return Value of authentification method is correct (Content False and Type is <class 'bool'>). |
| **Success** | Authentification state of server is correct (Content True and Type is <class 'bool'>). |
| **Success** | Authentification state of client is correct (Content True and Type is <class 'bool'>). |
| **Info** | Different secrets set |
| **Success** | Authentification state of server is correct (Content False and Type is <class 'bool'>). |
| **Success** | Authentification state of client is correct (Content False and Type is <class 'bool'>). |
| **Info** | Performing Authentification |
| **Success** | Return Value of authentification method is correct (Content False and Type is <class 'bool'>). |

| | |
|---|---|
| **Success** | Authentification state of server is correct (Content False and Type is <class 'bool'>). |
| **Success** | Authentification state of client is correct (Content False and Type is <class 'bool'>). |
| **Info** | Identical secrets set |
| **Info** | Performing Authentification |
| **Success** | Return Value of authentification method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Authentification state of server is correct (Content True and Type is <class 'bool'>). |
| **Success** | Authentification state of client is correct (Content True and Type is <class 'bool'>). |
| **Info** | Corrupting the authentification mechanism |
| **Info** | Performing Authentification |
| **Success** | Return Value of authentification method is correct (Content False and Type is <class 'bool'>). |
| **Success** | Authentification state of server is correct (Content False and Type is <class 'bool'>). |
| **Success** | Authentification state of client is correct (Content False and Type is <class 'bool'>). |

### 3.2.4 An automatic authentification shall available

**Description**
An authentification is executed by the client on every connect.

**Reason for the implementation**
Simplify handling for authentification.

**Fitcriterion**
Check authentification feedback (client and server) after connect has been triggered.

**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 (36) |
| Start-Time: | 2021-01-06 22:49:01,138 |
| Finished-Time: | 2021-01-06 22:49:01,852 |
| Time-Consumption | 0.714s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Identical secrets set and automatic authentification |
| **Success** | Authentification state of server is correct (Content False and Type is <type 'bool'>). |
| **Success** | Authentification state of client is correct (Content False and Type is <type 'bool'>). |
| **Info** | Server and Client connect callback triggered |
| **Success** | Authentification state of server is correct (Content True and Type is <type 'bool'>). |
| **Success** | Authentification state of client is correct (Content True and Type is <type 'bool'>). |

**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 (36) |
| Start-Time: | 2021-01-06 22:49:15,649 |
| Finished-Time: | 2021-01-06 22:49:16,362 |
| Time-Consumption | 0.713s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Identical secrets set and automatic authentification |
| **Success** | Authentification state of server is correct (Content False and Type is <class 'bool'>). |
| **Success** | Authentification state of client is correct (Content False and Type is <class 'bool'>). |
| **Info** | Server and Client connect callback triggered |
| **Success** | Authentification state of server is correct (Content True and Type is <class 'bool'>). |
| **Success** | Authentification state of client is correct (Content True and Type is <class 'bool'>). |

### 3.2.5 Communication (rx and tx) shall be disabled, if a secret is given but no authentification had been successfully performed.

**Description**

Communication (rx and tx) shall be disabled, if a secret is given. Except of a response for registered services, saying that a Authentification is required.

**Reason for the implementation**

Message protection (e.g. for secure functions or data)

**Fitcriterion**

RX and TX is not possible, till a successfull authentification has been performed.

**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: | 2021-01-06 22:49:01,852 |
| Finished-Time: | 2021-01-06 22:49:03,188 |
| Time-Consumption | 1.336s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Setting a Server secret and no Client secret |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content {u'status': 3, u'service_id': 31, u'data': None, u'data_id': 36} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Setting no Server secret but a Client secret |
| **Info** | Transfering a message server → client |

| | |
|---|---|
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Identical secrets set |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Performing Authentification |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content {u'status': 0, u'service_id': 17, u'data': u'msg1_data_to_be_transfered', u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on client side is correct (Content {u'status': 4, u'service_id': 17, u'data': u'msg2_data_to_be_transfered', u'data_id': 35} and Type is <class 'socket_protocol.data_storage'>). |

**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: | 2021-01-06 22:49:16,363 |
| Finished-Time: | 2021-01-06 22:49:17,697 |
| Time-Consumption | 1.334s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Setting a Server secret and no Client secret |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content {'data_id': 36, 'service_id': 31, 'status': 3, 'data': None} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Setting no Server secret but a Client secret |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Identical secrets set |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content False and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content False and Type is <class 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <class 'NoneType'>). |

| | |
|---|---|
| **Info** | Performing Authentification |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content {'data_id': 34, 'service_id': 17, 'status': 0, 'data': 'msg1_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on client side is correct (Content {'data_id': 35, 'service_id': 17, 'status': 4, 'data': 'msg2_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |

### 3.2.6 A whitelist for communication (rx and tx) shall be available to enable communication for unauthorised counterparts

**Description**

It shall be possible to add a specific message, identified by Service-ID and Data-ID, to a whitelist. All messages added to that whitelist shall be transmitted and received, if no authentification was successfull performed.

**Reason for the implementation**

Give the user the possibility to define messages which will not be protected behind the authentification mechanism.

**Fitcriterion**

Transmition and Reception will be enabled, after the message has been added to the whitelist.

**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: | 2021-01-06 22:49:03,189 |
| Finished-Time: | 2021-01-06 22:49:05,027 |
| Time-Consumption | 1.839s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Identical secrets set |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Added msg1 to client whitelist (sid=17, did=34) |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Transfering a message server → client |

| | |
|---|---|
| **Success** | Returnvalue of Server send Method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Added msg1 to server whitelist (sid=17, did=34) |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content {u'status': 0, u'service_id': 17, u'data': u'msg1_data_to_be_transfered', u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content False and Type is <type 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Added msg2 to client and server whitelist (sid=17, did=35) |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content {u'status': 0, u'service_id': 17, u'data': u'msg1_data_to_be_transfered', u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on client side is correct (Content {u'status': 4, u'service_id': 17, u'data': u'msg2_data_to_be_transfered', u'data_id': 35} and Type is <class 'socket_protocol.data_storage'>). |

**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: | 2021-01-06 22:49:17,698 |
| Finished-Time: | 2021-01-06 22:49:19,532 |
| Time-Consumption | 1.835s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Identical secrets set |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content False and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content False and Type is <class 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Added msg1 to client whitelist (sid=17, did=34) |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content False and Type is <class 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <class 'NoneType'>). |

| | |
|---|---|
| **Info** | Added msg1 to server whitelist (sid=17, did=34) |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content {'data_id': 34, 'service_id': 17, 'status': 0, 'data': 'msg1_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content False and Type is <class 'bool'>). |
| **Success** | Received message on client side is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Added msg2 to client and server whitelist (sid=17, did=35) |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content {'data_id': 34, 'service_id': 17, 'status': 0, 'data': 'msg1_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on client side is correct (Content {'data_id': 35, 'service_id': 17, 'status': 4, 'data': 'msg2_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |

### 3.2.7 Define a channel name for the server and client after connection is established

**Description**

After the connection is established, the client will initiate the channel name exchange. The channel name defined on the client side will be dominant.

**Reason for the implementation**

Structured logging by creating logger childs for each channel.

**Fitcriterion**

Perform a channel name exchange with no channel name definition, differing channel name definition and identical channel name definition. In all cases, the channel name of the client will be used. Perform two channel name exchanges with only one channel name definition. This definition will be used.

**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 (39) |
| Start-Time: | 2021-01-06 22:49:05,028 |
| Finished-Time: | 2021-01-06 22:49:08,068 |
| Time-Consumption | 3.040s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Setting no Channel name for server and client |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content None and Type is <type 'NoneType'>). |

| | |
|---|---|
| **Success** | Channel name of client is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Setting different Channel names for client and Server |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content 'client' and Type is <type 'str'>). |
| **Success** | Channel name of client is correct (Content 'client' and Type is <type 'str'>). |
| **Info** | Setting identical Channel names for client and server |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content 'unittest' and Type is <type 'str'>). |
| **Success** | Channel name of client is correct (Content 'unittest' and Type is <type 'str'>). |
| **Info** | Setting Channel name for client only |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content 'client' and Type is <type 'str'>). |
| **Success** | Channel name of client is correct (Content 'client' and Type is <type 'str'>). |
| **Info** | Setting Channel name for server only |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content 'server' and Type is <type 'str'>). |
| **Success** | Channel name of client is correct (Content 'server' and Type is <type 'str'>). |

**Testresult**

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

| | |
|---|---|
| Testrun: | python 3.8.5 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (39) |
| Start-Time: | 2021-01-06 22:49:19,533 |
| Finished-Time: | 2021-01-06 22:49:22,566 |
| Time-Consumption | 3.033s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Setting no Channel name for server and client |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content None and Type is <class 'NoneType'>). |
| **Success** | Channel name of client is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Setting different Channel names for client and Server |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content 'client' and Type is <class 'str'>). |
| **Success** | Channel name of client is correct (Content 'client' and Type is <class 'str'>). |
| **Info** | Setting identical Channel names for client and server |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content 'unittest' and Type is <class 'str'>). |
| **Success** | Channel name of client is correct (Content 'unittest' and Type is <class 'str'>). |
| **Info** | Setting Channel name for client only |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content 'client' and Type is <class 'str'>). |
| **Success** | Channel name of client is correct (Content 'client' and Type is <class 'str'>). |
| **Info** | Setting Channel name for server only |
| **Info** | Server and Client connect callback triggered |
| **Success** | Channel name of server is correct (Content 'server' and Type is <class 'str'>). |

| | |
|---|---|
| **Success** | Channel name of client is correct (Content 'server' and Type is <class 'str'>). |

### 3.2.8 The User shall be able to define a new service

**Description**
The service is defined by a Request Service-ID and a Response Service-ID.

**Reason for the implementation**
Definition of Request and Response SIDs.

**Fitcriterion**
Define a service and check, that the server will respond on the new Service-ID. The Status shall be "Request has no callback. Data buffered.", because no callback is registered for that request.

**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 (40) |
| Start-Time: | 2021-01-06 22:49:08,069 |
| Finished-Time: | 2021-01-06 22:49:08,685 |
| Time-Consumption | 0.616s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Transfering a message client → server → client |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Adding service to server instance for the transmit message |
| **Info** | Transfering a message client → server → client |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Received message on server side is correct (Content {u'status': 1, u'service_id': 18, u'data': None, u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>). |

**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 (40) |
| Start-Time: | 2021-01-06 22:49:22,567 |
| Finished-Time: | 2021-01-06 22:49:23,182 |
| Time-Consumption | 0.615s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Transfering a message client → server → client |

| | |
|---|---|
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Adding service to server instance for the transmit message |
| **Info** | Transfering a message client → server → client |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Received message on server side is correct (Content {'data_id': 34, 'service_id': 18, 'status': 1, 'data': None} and Type is <class 'socket_protocol.data_storage'>). |

### 3.2.9 Registration of already registered request Service-ID or response Service-ID shall not be possible

**Description**

An exception shall be raised, if a service registration with an existing request SID or response SID is performed.

**Reason for the implementation**

Changing existing services will create strange situations with already registered callbacks.

**Fitcriterion**

Catch exception for registration of existing request and response SID.

**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 (41) |
| Start-Time: | 2021-01-06 22:49:08,685 |
| Finished-Time: | 2021-01-06 22:49:08,694 |
| Time-Consumption | 0.009s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Adding a service with an already registered request SID |
| **Success** | Expected Exception RequestSidExistsError was triggered |
| **Info** | Adding a service with an already registered response SID |
| **Success** | Expected Exception ResponseSidExistsError was triggered |

**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 (41) |
| Start-Time: | 2021-01-06 22:49:23,183 |
| Finished-Time: | 2021-01-06 22:49:23,191 |
| Time-Consumption | 0.009s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |

| | |
|---|---|
| **Info** | Adding a service with an already registered request SID |
| **Success** | Expected Exception RequestSidExistsError was triggered |
| **Info** | Adding a service with an already registered response SID |
| **Success** | Expected Exception ResponseSidExistsError was triggered |

## 3.3 Callbacks

### 3.3.1 It shall be possible to register a callback for a specific Service- and Data-ID

**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: | 2021-01-06 22:49:08,694 |
| Finished-Time: | 2021-01-06 22:49:09,007 |
| Time-Consumption | 0.313s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering a correct working Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Overwriting existing Callback using one with faulty return values |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 2, u'service_id': 11, u'data': None, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Removing the registered Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content None and Type is <type 'NoneType'>). |
| **Success** | Message received by client is correct (Content {u'status': 1, u'service_id': 11, u'data': None, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

**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: | 2021-01-06 22:49:23,192 |
| Finished-Time: | 2021-01-06 22:49:23,504 |
| Time-Consumption | 0.313s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering a correct working Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Overwriting existing Callback using one with faulty return values |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 2, 'data': None} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Removing the registered Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content None and Type is <class 'NoneType'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 1, 'data': None} and Type is <class 'socket_protocol.data_storage'>). |

### 3.3.2 It shall be possible to register a callback for a specific Service-ID and all Data-IDs

**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 (46) |
| Start-Time: | 2021-01-06 22:49:09,007 |
| Finished-Time: | 2021-01-06 22:49:09,120 |
| Time-Consumption | 0.112s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering a correct working Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

**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 (46) |
| Start-Time: | 2021-01-06 22:49:23,505 |
| Finished-Time: | 2021-01-06 22:49:23,615 |
| Time-Consumption | 0.110s |

| Testsummary: | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering a correct working Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>). |

### 3.3.3 It shall be possible to register a callback for a specific Data-IDs and all Service-IDs

**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 (47) |
| Start-Time: | 2021-01-06 22:49:09,121 |
| Finished-Time: | 2021-01-06 22:49:09,229 |
| Time-Consumption | 0.108s |

| Testsummary: | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering a correct working Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

**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 (47) |
| Start-Time: | 2021-01-06 22:49:23,615 |
| Finished-Time: | 2021-01-06 22:49:23,724 |
| Time-Consumption | 0.109s |

| Testsummary: | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering a correct working Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>). |

### 3.3.4 It shall be possible to register a callback for all incomming messages

**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 (48) |
| Start-Time: | 2021-01-06 22:49:09,229 |
| Finished-Time: | 2021-01-06 22:49:09,334 |
| Time-Consumption | 0.105s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering a correct working Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

**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 (48) |
| Start-Time: | 2021-01-06 22:49:23,725 |
| Finished-Time: | 2021-01-06 22:49:23,835 |
| Time-Consumption | 0.110s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering a correct working Callback |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>). |

### 3.3.5 Callback choice, if several callbacks are available (caused by wildcard callbacks)

**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 (49) |
| Start-Time: | 2021-01-06 22:49:09,334 |
| Finished-Time: | 2021-01-06 22:49:09,763 |
| Time-Consumption | 0.430s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering all kind of Callbacks |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Removing Callback for a specific Data- and Service-ID |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 6, u'service_id': 11, u'data': 34, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Removing Callback for a specific Service-ID and all Data-IDs |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 6, u'service_id': 11, u'data': 35, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Removing Callback for a specific Data-ID and all Serice-IDs |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 36, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

**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 (49) |
| Start-Time: | 2021-01-06 22:49:23,835 |
| Finished-Time: | 2021-01-06 22:49:24,251 |
| Time-Consumption | 0.416s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Registering all kind of Callbacks |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Removing Callback for a specific Data- and Service-ID |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |

| | |
|---|---|
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 6, 'data': 34} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Removing Callback for a specific Service-ID and all Data-IDs |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 6, 'data': 35} and Type is <class 'socket_protocol.data_storage'>). |
| **Info** | Removing Callback for a specific Data-ID and all Serice-IDs |
| **Info** | Transfering data |
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 36} and Type is <class 'socket_protocol.data_storage'>). |

## 3.4 Some additional Information and Passthrough Methods

### 3.4.1 Connection established information

**Testresult**

This test was passed with the state: **Success**. See also full trace in section A.1.19!

| | |
|---|---|
| Testrun: | python 2.7.18 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (53) |
| Start-Time: | 2021-01-06 22:49:09,764 |
| Finished-Time: | 2021-01-06 22:49:09,884 |
| Time-Consumption | 0.120s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Success** | Client connection status is correct (Content False and Type is <type 'bool'>). |
| **Success** | Server connection status is correct (Content False and Type is <type 'bool'>). |
| **Info** | Connecting Client |
| **Success** | Client connection status is correct (Content True and Type is <type 'bool'>). |
| **Success** | Server connection status is correct (Content False and Type is <type 'bool'>). |
| **Info** | Connecting Server |
| **Success** | Client connection status is correct (Content True and Type is <type 'bool'>). |
| **Success** | Server connection status is correct (Content True and Type is <type 'bool'>). |
| **Info** | Adding secrets to socket_protocol |
| **Success** | Client connection status is correct (Content False and Type is <type 'bool'>). |
| **Success** | Server connection status is correct (Content False and Type is <type 'bool'>). |
| **Info** | Doing authentification |
| **Success** | Client connection status is correct (Content True and Type is <type 'bool'>). |
| **Success** | Server connection status is correct (Content True and Type is <type 'bool'>). |

**Testresult**

This test was passed with the state: **Success**. See also full trace in section B.1.19!

| | |
|---|---|
| Testrun: | python 3.8.5 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (53) |
| Start-Time: | 2021-01-06 22:49:24,252 |
| Finished-Time: | 2021-01-06 22:49:24,363 |
| Time-Consumption | 0.112s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Success** | Client connection status is correct (Content False and Type is <class 'bool'>). |
| **Success** | Server connection status is correct (Content False and Type is <class 'bool'>). |
| **Info** | Connecting Client |
| **Success** | Client connection status is correct (Content True and Type is <class 'bool'>). |
| **Success** | Server connection status is correct (Content False and Type is <class 'bool'>). |
| **Info** | Connecting Server |
| **Success** | Client connection status is correct (Content True and Type is <class 'bool'>). |
| **Success** | Server connection status is correct (Content True and Type is <class 'bool'>). |
| **Info** | Adding secrets to socket_protocol |
| **Success** | Client connection status is correct (Content False and Type is <class 'bool'>). |
| **Success** | Server connection status is correct (Content False and Type is <class 'bool'>). |
| **Info** | Doing authentification |
| **Success** | Client connection status is correct (Content True and Type is <class 'bool'>). |
| **Success** | Server connection status is correct (Content True and Type is <class 'bool'>). |

### 3.4.2 Is connected information

**Testresult**

This test was passed with the state: **Success**. See also full trace in section A.1.20!

| | |
|---|---|
| Testrun: | python 2.7.18 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (54) |
| Start-Time: | 2021-01-06 22:49:09,885 |
| Finished-Time: | 2021-01-06 22:49:09,903 |
| Time-Consumption | 0.019s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Success** | Client Communication instance connection status is correct (Content False and Type is <type 'bool'>). |
| **Success** | Server Communication instance connection status is correct (Content False and Type is <type 'bool'>). |
| **Info** | Connecting Client |
| **Success** | Client Communication instance connection status is correct (Content True and Type is <type 'bool'>). |
| **Success** | Server Communication instance connection status is correct (Content False and Type is <type 'bool'>). |
| **Info** | Connecting Server |
| **Success** | Client Communication instance connection status is correct (Content True and Type is <type 'bool'>). |

| | |
|---|---|
| **Success** | Server Communication instance connection status is correct (Content True and Type is <type 'bool'>). |

**Testresult**

This test was passed with the state: **Success**. See also full trace in section B.1.20!

| | |
|---|---|
| Testrun: | python 3.8.5 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (54) |
| Start-Time: | 2021-01-06 22:49:24,364 |
| Finished-Time: | 2021-01-06 22:49:24,372 |
| Time-Consumption | 0.008s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Success** | Client Communication instance connection status is correct (Content False and Type is <class 'bool'>). |
| **Success** | Server Communication instance connection status is correct (Content False and Type is <class 'bool'>). |
| **Info** | Connecting Client |
| **Success** | Client Communication instance connection status is correct (Content True and Type is <class 'bool'>). |
| **Success** | Server Communication instance connection status is correct (Content False and Type is <class 'bool'>). |
| **Info** | Connecting Server |
| **Success** | Client Communication instance connection status is correct (Content True and Type is <class 'bool'>). |
| **Success** | Server Communication instance connection status is correct (Content True and Type is <class 'bool'>). |

### 3.4.3 Reconnect Method

**Testresult**

This test was passed with the state: **Success**. See also full trace in section A.1.21!

| | |
|---|---|
| Testrun: | python 2.7.18 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (55) |
| Start-Time: | 2021-01-06 22:49:09,903 |
| Finished-Time: | 2021-01-06 22:49:09,912 |
| Time-Consumption | 0.008s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Success** | Reconnect executed marker is correct (Content False and Type is <type 'bool'>). |
| **Success** | Reconnect executed marker is correct (Content False and Type is <type 'bool'>). |
| **Info** | Executing reconnect for Server |
| **Success** | Reconnect executed marker is correct (Content True and Type is <type 'bool'>). |
| **Success** | Reconnect executed marker is correct (Content False and Type is <type 'bool'>). |
| **Info** | Executing reconnect for Client |

| | |
|---|---|
| **Success** | Reconnect executed marker is correct (Content True and Type is <type 'bool'>). |
| **Success** | Reconnect executed marker is correct (Content True and Type is <type 'bool'>). |

**Testresult**

This test was passed with the state: **Success**. See also full trace in section B.1.21!

| | |
|---|---|
| Testrun: | python 3.8.5 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (55) |
| Start-Time: | 2021-01-06 22:49:24,372 |
| Finished-Time: | 2021-01-06 22:49:24,376 |
| Time-Consumption | 0.003s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Success** | Reconnect executed marker is correct (Content False and Type is <class 'bool'>). |
| **Success** | Reconnect executed marker is correct (Content False and Type is <class 'bool'>). |
| **Info** | Executing reconnect for Server |
| **Success** | Reconnect executed marker is correct (Content True and Type is <class 'bool'>). |
| **Success** | Reconnect executed marker is correct (Content False and Type is <class 'bool'>). |
| **Info** | Executing reconnect for Client |
| **Success** | Reconnect executed marker is correct (Content True and Type is <class 'bool'>). |
| **Success** | Reconnect executed marker is correct (Content True and Type is <class 'bool'>). |

## 3.5 Depreceated struct protocol

### 3.5.1 A full Message Object including the defined properties and data shall be transfered.

**Description**

Every Communication shall transfer a complete message with its content.

**Reason for the implementation**

See Reasons for every single information of the Message Object.

**Fitcriterion**

Send two different messages and compare the received message with each sent message.

**Testresult**

This test was passed with the state: **Success**. See also full trace in section A.1.22!

| | |
|---|---|
| Testrun: | python 2.7.18 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (59) |
| Start-Time: | 2021-01-06 22:49:09,912 |
| Finished-Time: | 2021-01-06 22:49:10,525 |
| Time-Consumption | 0.613s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Checksum Error → No message received by server is correct (Content None and Type is <type 'NoneType'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>). |
| **Success** | Checksum Error → No message received by client is correct (Content None and Type is <type 'NoneType'>). |

**Testresult**

This test was passed with the state: **Success**. See also full trace in section B.1.22!

| | |
|---|---|
| Testrun: | python 3.8.5 (final) |
| Caller: | /user_data/data/dirk/prj/unittest/socket_protocol/unittest/src/tests/__init__.py (59) |
| Start-Time: | 2021-01-06 22:49:24,376 |
| Finished-Time: | 2021-01-06 22:49:24,983 |
| Time-Consumption | 0.607s |

**Testsummary:**

| | |
|---|---|
| **Info** | Setting up communication |
| **Info** | Transfering a message client → server |
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Checksum Error → No message received by server is correct (Content None and Type is <class 'NoneType'>). |
| **Info** | Transfering a message server → client |
| **Success** | Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>). |
| **Success** | Checksum Error → No message received by client is correct (Content None and Type is <class 'NoneType'>). |

# A Trace for testrun with python 2.7.18 (final)

## A.1 Tests with status Info (22)

### A.1.1 Status

**Description**

The Status shall hold some general information (in most cases it is used by the responder). Examples: Okay, Service or Data unknown, Operation not permitted, Authentification required, ...

**Reason for the implementation**

Give the possibility to transfer additional status information (e.g. to explain negative responses).

**Fitcriterion**

A Status is part of the Message Object and it is holding the Status information.

**Testresult**

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

| | |
|---|---|
| **Info** | Creating empty message object: {'status': None, 'service_id': None, 'data': None, 'data_id': None} |

| | |
|---|---|
| **Success** | status is part of the message object is correct ('status' is in the list or dict). |

```
Result (status is part of the message object): {'status': None, 'service_id': None, 'data':
↪  None, 'data_id': None} (<class 'socket_protocol.data_storage'>)
Expectation (status is part of the message object): 'status' in result
```

| | |
|---|---|
| **Info** | Creating a maximum message object: {'status': 'S', 'service_id': 'SID', 'data': 'D', 'data_id': 'DID'} |

| | |
|---|---|
| **Success** | status is part of the message object is correct ('status' is in the list or dict). |

```
Result (status is part of the message object): {'status': 'S', 'service_id': 'SID', 'data':
↪  'D', 'data_id': 'DID'} (<class 'socket_protocol.data_storage'>)
Expectation (status is part of the message object): 'status' in result
```

| | |
|---|---|
| **Success** | Content in message object for status is correct (Content 'S' and Type is <type 'str'>). |

```
Result (Content in message object for status): 'S' (<type 'str'>)
Expectation (Content in message object for status): result = 'S' (<type 'str'>)
```

### A.1.2 Service-ID

**Description**

The Service-ID shall hold information about the type of the request / corresponding response. Examples: read request, write request, read response, write response, ...

**Reason for the implementation**

Give the requestor the possibility to use different types (Services) for a transfer.

**Fitcriterion**

A Service-ID is part of the Message Object and it is holding the Service-ID information.

**Testresult**

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

| | |
|---|---|
| **Info** | Creating empty message object: {'status': None, 'service_id': None, 'data': None, 'data_id': None} |

| | |
|---|---|
| **Success** | service_id is part of the message object is correct ('service_id' is in the list or dict). |

```
Result (service_id is part of the message object): {'status': None, 'service_id': None,
↪  'data': None, 'data_id': None} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (service_id is part of the message object): 'service_id' in result
```

| | |
|---|---|
| **Info** | Creating a maximum message object: {'status': 'S', 'service_id': 'SID', 'data': 'D', 'data_id': 'DID'} |

| | |
|---|---|
| **Success** | service_id is part of the message object is correct ('service_id' is in the list or dict). |

```
Result (service_id is part of the message object): {'status': 'S', 'service_id': 'SID',
↪  'data': 'D', 'data_id': 'DID'} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (service_id is part of the message object): 'service_id' in result
```

| | |
|---|---|
| **Success** | Content in message object for service_id is correct (Content 'SID' and Type is <type 'str'>). |

```
Result (Content in message object for service_id): 'SID' (<type 'str'>)
```

```
Expectation (Content in message object for service_id): result = 'SID' (<type 'str'>)
```

### A.1.3 Data-ID

**Description**

The Data-ID shall hold information to differtiate the data for a specific Service.

**Reason for the implementation**

Give the possibility to transfer different information for each Service.

**Fitcriterion**

A Data-ID is part of the Message Object and it is holding the Data-ID information.

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

| | |
|---|---|
| **Info** | Creating empty message object: {'status': None, 'service_id': None, 'data': None, 'data_id': None} |

| | |
|---|---|
| **Success** | data_id is part of the message object is correct ('data_id' is in the list or dict). |

```
Result (data_id is part of the message object): {'status': None, 'service_id': None, 'data':
↪  None, 'data_id': None} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (data_id is part of the message object): 'data_id' in result
```

| | |
|---|---|
| **Info** | Creating a maximum message object: {'status': 'S', 'service_id': 'SID', 'data': 'D', 'data_id': 'DID'} |

| | |
|---|---|
| **Success** | data_id is part of the message object is correct ('data_id' is in the list or dict). |

```
Result (data_id is part of the message object): {'status': 'S', 'service_id': 'SID', 'data':
↪  'D', 'data_id': 'DID'} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (data_id is part of the message object): 'data_id' in result
```

| | |
|---|---|
| **Success** | Content in message object for data_id is correct (Content 'DID' and Type is <type 'str'>). |

```
Result (Content in message object for data_id): 'DID' (<type 'str'>)
```

```
Expectation (Content in message object for data_id): result = 'DID' (<type 'str'>)
```

### A.1.4 Data

**Description**
The Data shall hold the data to be transfered. For the most requests not data is transmitted.

**Reason for the implementation**
Give the possibility to transfer Data.

**Fitcriterion**
Data is part of the Message Object and it is holding the Data information.

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

| | |
|---|---|
| **Info** | Creating empty message object: {'status': None, 'service_id': None, 'data': None, 'data_id': None} |

| | |
|---|---|
| **Success** | data is part of the message object is correct ('data' is in the list or dict). |

```
Result (data is part of the message object): {'status': None, 'service_id': None, 'data':
↪  None, 'data_id': None} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (data is part of the message object): 'data' in result
```

| **Info** | Creating a maximum message object: {'status': 'S', 'service_id': 'SID', 'data': 'D', 'data_id': 'DID'} |
|---|---|

| **Success** | data is part of the message object is correct ('data' is in the list or dict). |
|---|---|

```
Result (data is part of the message object): {'status': 'S', 'service_id': 'SID', 'data':
↪  'D', 'data_id': 'DID'} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (data is part of the message object): 'data' in result
```

| **Success** | Content in message object for data is correct (Content 'D' and Type is <type 'str'>). |
|---|---|

```
Result (Content in message object for data): 'D' (<type 'str'>)
```

```
Expectation (Content in message object for data): result = 'D' (<type 'str'>)
```

### A.1.5 A full Message Object including the defined properties and data shall be transfered.

**Description**

Every Communication shall transfer a complete message with its content.

**Reason for the implementation**

See Reasons for every single information of the Message Object.

**Fitcriterion**

Send two different messages and compare the received message with each sent message.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
Receive data (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
SP server: RX <- service: 17, data_id: 34, status: okay, data: "u'msg1_data_to_be_transfered'"
```

```
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Received message on server side is correct (Content {u'status': 0, u'service_id': 17, u'data': u'msg1_data_to_be_transfered', u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {u'status': 0, u'service_id': 17, u'data':
↪  u'msg1_data_to_be_transfered', u'data_id': 34} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on server side): result = {'status': 0, 'service_id': 17,
↪  'data': 'msg1_data_to_be_transfered', 'data_id': 34} (<class
↪  'socket_protocol.data_storage'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
```

```
Receive data (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
```

```
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "u'msg2_data_to_be_transfered'"
```

```
SP client: RX <- Message has a peculiar status: status: service or data unknown
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Server send Method): result = True (<type 'bool'>)
```

| | |
|---|---|
| **Success** | Received message on client side is correct (Content {u'status': 4, u'service_id': 17, u'data': u'msg2_data_to_be_transfered', u'data_id': 35} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Received message on client side): {u'status': 4, u'service_id': 17, u'data':
↪  u'msg2_data_to_be_transfered', u'data_id': 35} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on client side): result = {'status': 4, 'service_id': 17,
↪  'data': 'msg2_data_to_be_transfered', 'data_id': 35} (<class
↪  'socket_protocol.data_storage'>)
```

### A.1.6 A checksumm shall ensure the correct transmition

**Description**

If the checksum does not fit to the checksum of the transferred data, the message will be ignored, because the complete content including the Service- and Data-ID is possibly corrupted.

**Reason for the implementation**

Ensure correct data transfer.

**Fitcriterion**

Corrupted message is not in the receive buffer after transmission.

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
Receive data (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9c
```

```
SP server: RX <- Received message has a wrong checksum. Message will be ignored.
```

```
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Checksum Error → No message received by server is correct (Content None and Type is <type 'NoneType'>).

```
Result (Checksum Error -> No message received by server): None (<type 'NoneType'>)
```

```
Expectation (Checksum Error -> No message received by server): result = None (<type
↪  'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
```

```
Receive data (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
```

```
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "u'msg2_data_to_be_transfered'"
```

```
SP client: RX <- Message has a peculiar status: status: service or data unknown
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

```
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): True (<type 'bool'>)
Expectation (Returnvalue of Server send Method): result = True (<type 'bool'>)
```

| | |
|---|---|
| **Success** | Checksum Error → No message received by client is correct (Content None and Type is <type 'NoneType'>). |

```
Result (Checksum Error -> No message received by client): None (<type 'NoneType'>)
Expectation (Checksum Error -> No message received by client): result = None (<type
↪  'NoneType'>)
```

### A.1.7 An authentification between server and client shall be possible including status feedback methods

**Description**

The Client shall have a method to initiate the authentification. In case that the server and the client do have identical secrets, the authentification shall be successfull.

**Reason for the implementation**

Message protection (e.g. for secure functions or data)

**Fitcriterion**

Check authentification method feedback (client) and authentification feedback (client and server), in case of differing and identical secrets.

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

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

**Success** Authentification state of server is correct (Content True and Type is <type 'bool'>).

```
Result (Authentification state of server): True (<type 'bool'>)
Expectation (Authentification state of server): result = True (<type 'bool'>)
```

**Success** Authentification state of client is correct (Content True and Type is <type 'bool'>).

```
Result (Authentification state of client): True (<type 'bool'>)
Expectation (Authentification state of client): result = True (<type 'bool'>)
```

**Info** Different secrets set

**Success** Authentification state of server is correct (Content False and Type is <type 'bool'>).

```
Result (Authentification state of server): False (<type 'bool'>)
Expectation (Authentification state of server): result = False (<type 'bool'>)
```

**Success** Authentification state of client is correct (Content False and Type is <type 'bool'>).

```
Result (Authentification state of client): False (<type 'bool'>)
Expectation (Authentification state of client): result = False (<type 'bool'>)
```

**Info** Performing Authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 30 7d 10 4d cd 55
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 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 30 7d 10 4d cd 55
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'39cbad68d7688a6eacb746081099bdb15938c8706a3244970581f82683958b48'"
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 31 2c 20 22 64 61 74 61 22 3a 20 22 33 39 63 62 61 64 36 38 64 37 36 38 38 61 36
↪  65 61 63 62 37 34 36 30 38 31 30 39 39 62 64 62 31 35 39 33 38 63 38 37 30 36 61 33 32 34
↪  34 39 37 30 35 38 31 66 38 32 36 38 33 39 35 38 62 34 38 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d b4 3e 73 ab
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 31 2c 20 22 64 61 74 61 22 3a 20 22 33 39 63 62 61 64 36 38 64 37 36 38 38 61
↪  36 65 61 63 62 37 34 36 30 38 31 30 39 39 62 64 62 31 35 39 33 38 63 38 37 30 36 61 33 32
↪  34 34 39 37 30 35 38 31 66 38 32 36 38 33 39 35 38 62 34 38 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d b4 3e 73 ab
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "u'39cbad68d7688a6eacb746081099bdb15938c8706a3244970581f82683958b48'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'d971ebb309d9c96503fa6b2138f7e30b29009b14dbe4069d34b3f6cd9b1633a9954b13a93594c9b99c97053
↪  e4f4d644bda7f2a0b958b331112063c8bd2ac030d'"
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 30 2c 20 22 64 61 74 61 22 3a 20 22 64 39 37 31 65 62 62 33 30 39 64 39 63 39 36
↪  35 30 33 66 61 36 62 32 31 33 38 66 37 65 33 30 62 32 39 30 30 39 62 31 34 64 62 65 34 30
↪  36 39 64 33 34 62 33 66 36 63 64 39 62 31 36 33 33 61 39 39 35 34 62 31 33 61 39 33 35 39
↪  34 63 39 62 39 39 63 39 37 30 35 33 65 34 66 34 64 36 34 34 62 64 61 37 66 32 61 30 62 39
↪  35 38 62 33 33 31 31 31 32 30 36 33 63 38 62 64 32 61 63 30 33 30 64 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 31 7d 87 d8 31 d6
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 30 2c 20 22 64 61 74 61 22 3a 20 22 64 39 37 31 65 62 62 33 30 39 64 39 63 39
↪  36 35 30 33 66 61 36 62 32 31 33 38 66 37 65 33 30 62 32 39 30 30 39 62 31 34 64 62 65 34
↪  30 36 39 64 33 34 62 33 66 36 63 64 39 62 31 36 33 33 61 39 39 35 34 62 31 33 61 39 33 35
↪  39 34 63 39 62 39 39 63 39 37 30 35 33 65 34 66 34 64 36 34 34 62 64 61 37 66 32 61 30 62
↪  39 35 38 62 33 33 31 31 31 32 30 36 33 63 38 62 64 32 61 63 30 33 30 64 22 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 31 7d 87 d8 31 d6
SP server: RX <- service: authentification request, data_id: key, status: okay, data:
↪  "u'd971ebb309d9c96503fa6b2138f7e30b29009b14dbe4069d34b3f6cd9b1633a9954b13a93594c9b99c9705
↪  3e4f4d644bda7f2a0b958b331112063c8bd2ac030d'"
SP server: RX <- Executing callback __authentificate_check_key__ to process received data
SP server: TX <- service: authentification response, data_id: key, status: okay, data: "False"
Send data: (63): 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 66 61 6c 73 65 2c 20 22 64 61 74 61 5f 69 64 22
↪  3a 20 31 7d a1 48 27 7d
Receive data (63): 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 66 61 6c 73 65 2c 20 22 64 61 74 61 5f 69 64 22
```

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

**Success** Authentification state of server is correct (Content False and Type is <type 'bool'>).

```
Result (Authentification state of server): False (<type 'bool'>)
Expectation (Authentification state of server): result = False (<type 'bool'>)
```

**Success** Authentification state of client is correct (Content False and Type is <type 'bool'>).

```
Result (Authentification state of client): False (<type 'bool'>)
Expectation (Authentification state of client): result = False (<type 'bool'>)
```

**Info** Identical secrets set

**Info** Performing Authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 30 7d 10 4d cd 55
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 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 30 7d 10 4d cd 55
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'304b61ac154ab9c4d5b495331036bacc3d1ba73b58231cdfa65ab76672ef5a88'"
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 31 2c 20 22 64 61 74 61 22 3a 20 22 33 30 34 62 36 31 61 63 31 35 34 61 62 39 63
↪  34 64 35 62 34 39 35 33 33 31 30 33 36 62 61 63 63 33 64 31 62 61 37 33 62 35 38 32 33 31
↪  63 64 66 61 36 35 61 62 37 36 36 37 32 65 66 35 61 38 38 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d e2 11 2a ad
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 31 2c 20 22 64 61 74 61 22 3a 20 22 33 30 34 62 36 31 61 63 31 35 34 61 62 39
↪  63 34 64 35 62 34 39 35 33 33 31 30 33 36 62 61 63 63 33 64 31 62 61 37 33 62 35 38 32 33
↪  31 63 64 66 61 36 35 61 62 37 36 36 37 32 65 66 35 61 38 38 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d e2 11 2a ad
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "u'304b61ac154ab9c4d5b495331036bacc3d1ba73b58231cdfa65ab76672ef5a88'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'c8245de1de6057d04359964bcbae62100c634d66ee38b12cbe44af4e223ab644cf6847fce9be0fc780d8731
↪  89fb1bb5d97bc6699aa1d221829cdc9512289d25a'"
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 30 2c 20 22 64 61 74 61 22 3a 20 22 63 38 32 34 35 64 65 31 64 65 36 30 35 37 64
↪  30 34 33 35 39 39 36 34 62 63 62 61 65 36 32 31 30 30 63 36 33 34 64 36 36 65 65 33 38 62
↪  31 32 63 62 65 34 34 61 66 34 65 32 32 33 61 62 36 34 34 63 66 36 38 34 37 66 63 65 39 62
↪  65 30 66 63 37 38 30 64 38 37 33 31 38 39 66 62 31 62 62 35 64 39 37 62 63 36 36 39 39 61
↪  61 31 64 32 32 31 38 32 39 63 64 63 39 35 31 32 32 38 39 64 32 35 61 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 31 7d 03 29 6d 58
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 30 2c 20 22 64 61 74 61 22 3a 20 22 63 38 32 34 35 64 65 31 64 65 36 30 35 37
↪  64 30 34 33 35 39 39 36 34 62 63 62 61 65 36 32 31 30 30 63 36 33 34 64 36 36 65 65 33 38
↪  62 31 32 63 62 65 34 34 61 66 34 65 32 32 33 61 62 36 34 34 63 66 36 38 34 37 66 63 65 39
↪  62 65 30 66 63 37 38 30 64 38 37 33 31 38 39 66 62 31 62 62 35 64 39 37 62 63 36 36 39 39
↪  61 61 31 64 32 32 31 38 32 39 63 64 63 39 35 31 32 32 38 39 64 32 35 61 22 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 31 7d 03 29 6d 58
SP server: RX <- service: authentification request, data_id: key, status: okay, data:
↪  "u'c8245de1de6057d04359964bcbae62100c634d66ee38b12cbe44af4e223ab644cf6847fce9be0fc780d873
↪  189fb1bb5d97bc6699aa1d221829cdc9512289d25a'"
SP server: RX <- Executing callback __authentificate_check_key__ to process received data
SP server: TX <- service: authentification response, data_id: key, status: okay, data: "True"
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 74 72 75 65 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 31 7d 11 d3 26 78
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 74 72 75 65 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 31 7d 11 d3 26 78
```

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

**Success** Authentification state of server is correct (Content True and Type is <type 'bool'>).

```
Result (Authentification state of server): True (<type 'bool'>)
Expectation (Authentification state of server): result = True (<type 'bool'>)
```

**Success** Authentification state of client is correct (Content True and Type is <type 'bool'>).

```
Result (Authentification state of client): True (<type 'bool'>)
Expectation (Authentification state of client): result = True (<type 'bool'>)
```

**Info** Corrupting the authentification mechanism

```
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
```

**Info** Performing Authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 30 7d 10 4d cd 55
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 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 30 7d 10 4d cd 55
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: Executing callback __authentificate_create_seed__ to process received data
```

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

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

**Success** Authentification state of server is correct (Content False and Type is <type 'bool'>).

```
Result (Authentification state of server): False (<type 'bool'>)
Expectation (Authentification state of server): result = False (<type 'bool'>)
```

**Success** Authentification state of client is correct (Content False and Type is <type 'bool'>).

```
Result (Authentification state of client): False (<type 'bool'>)
Expectation (Authentification state of client): result = False (<type 'bool'>)
```

### A.1.8 An automatic authentification shall available

**Description**
An authentification is executed by the client on every connect.

**Reason for the implementation**
Simplify handling for authentification.

**Fitcriterion**
Check authentification feedback (client and server) after connect has been triggered.

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
Result (Authentification state of server): False (<type 'bool'>)
Expectation (Authentification state of server): result = False (<type 'bool'>)
```

**Success** Authentification state of client is correct (Content False and Type is <type 'bool'>).

```
Result (Authentification state of client): False (<type 'bool'>)
Expectation (Authentification state of client): result = False (<type 'bool'>)
```

**Info** Server and Client connect callback triggered

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 28 3b d3 54
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 38 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 28 3b d3 54
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 14 5b 30 5c
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 39 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 14 5b 30 5c
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 30 7d 10 4d cd 55
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 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 30 7d 10 4d cd 55
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'1d385cd82db5453445fc2fdf5970407c7420bbc060e4e225b09f079189dfa42c'"
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 31 2c 20 22 64 61 74 61 22 3a 20 22 31 64 33 38 35 63 64 38 32 64 62 35 34 35 33
↪  34 34 35 66 63 32 66 64 66 35 39 37 30 34 30 37 63 37 34 32 30 62 62 63 30 36 30 65 34 65
↪  32 32 35 62 30 39 66 30 37 39 31 38 39 64 66 61 34 32 63 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d f0 6a a1 06
Receive data (124): 7b 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 31 2c 20 22 64 61 74 61 22 3a 20 22 31 64 33 38 35 63 64 38 32 64 62 35 34 35
↪  33 34 34 35 66 63 32 66 64 66 35 39 37 30 34 30 37 63 37 34 32 30 62 62 63 30 36 30 65 34
↪  65 32 32 35 62 30 39 66 30 37 39 31 38 39 64 66 61 34 32 63 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d f0 6a a1 06
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "u'1d385cd82db5453445fc2fdf5970407c7420bbc060e4e225b09f079189dfa42c'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'bb3c1734a939e72078fa19ef1ee787c91935c91d9dab373dfcbcf8847e96e1ab8622e77d2f9c5921cba1fbe
↪  aab0b36b93b4d45d6ac2998c7516aa6bdbfdd99db'"
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 30 2c 20 22 64 61 74 61 22 3a 20 22 62 62 33 63 31 37 33 34 61 39 33 39 65 37 32
↪  30 37 38 66 61 31 39 65 66 31 65 65 37 38 37 63 39 31 39 33 35 63 39 31 64 39 64 61 62 33
↪  37 33 64 66 63 62 63 66 38 38 34 37 65 39 36 65 31 61 62 38 36 32 32 65 37 37 64 32 66 39
↪  63 35 39 32 31 63 62 61 31 66 62 65 61 61 62 30 62 33 36 62 39 33 62 34 64 34 35 64 36 61
```

```
Result (Authentification state of server): True (<type 'bool'>)
Expectation (Authentification state of server): result = True (<type 'bool'>)
```

| **Success** | Authentification state of client is correct (Content True and Type is <type 'bool'>). |
|---|---|

```
Result (Authentification state of client): True (<type 'bool'>)
Expectation (Authentification state of client): result = True (<type 'bool'>)
```

### A.1.9 Communication (rx and tx) shall be disabled, if a secret is given but no authentification had been successfully performed.

**Description**

Communication (rx and tx) shall be disabled, if a secret is given. Except of a response for registered services, saying that a Authentification is required.

**Reason for the implementation**

Message protection (e.g. for secure functions or data)

**Fitcriterion**

RX and TX is not possible, till a successfull authentification has been performed.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: execute request, data_id: 36, status: okay, data:
↪  "'msg3_data_to_be_transfered'"
```

```
Send data: (88): 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 30 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 33 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 36 7d 18
↪  82 9a 08
```

```
Receive data (88): 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 30 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 33 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 36 7d 18
↪  82 9a 08
```

```
SP server: RX <- Authentification is required. Just sending negative response.
```

```
SP server: TX <- service: execute response, data_id: 36, status: authentification required,
↪  data: "None"
```

```
Send data: (64): 7b 22 73 74 61 74 75 73 22 3a 20 33 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 33 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 33 36 7d 5e 04 41 f5
```

```
Receive data (64): 7b 22 73 74 61 74 75 73 22 3a 20 33 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 33 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 33 36 7d 5e 04 41 f5
```

```
SP client: RX <- service: execute response, data_id: 36, status: authentification required,
↪  data: "None"
```

```
SP client: RX <- Message has a peculiar status: status: authentification required
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Received message on server side is correct (Content {u'status': 3, u'service_id': 31, u'data': None, u'data_id': 36} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {u'status': 3, u'service_id': 31, u'data': None,
↪  u'data_id': 36} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on server side): result = {'status': 3, 'service_id': 31,
↪  'data': None, 'data_id': 36} (<class 'socket_protocol.data_storage'>)
```

**Info** Setting no Server secret but a Client secret

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
```

```
Receive data (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
```

```
SP client: RX <- Authentification is required. Message will be ignored.
```

```
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Server send Method): result = True (<type 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on client side): None (<type 'NoneType'>)
```

```
Expectation (Received message on client side): result = None (<type 'NoneType'>)
```

**Info** Identical secrets set

**Info** Transfering a message client → server

```
SP client: TX -> Authentification is required. Message service: 17, data_id: 34, status:
↪  okay, data: 'msg1_data_to_be_transfered' will be ignored.
```

```
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content False and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): False (<type 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = False (<type 'bool'>)
```

**Success** Received message on server side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on server side): None (<type 'NoneType'>)
```

```
Expectation (Received message on server side): result = None (<type 'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX -> Authentification is required. Message service: 17, data_id: 35, status:
↪  service or data unknown, data: 'msg2_data_to_be_transfered' will be ignored.
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content False and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): False (<type 'bool'>)
Expectation (Returnvalue of Server send Method): result = False (<type 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on client side): None (<type 'NoneType'>)
Expectation (Received message on client side): result = None (<type 'NoneType'>)
```

**Info** Performing Authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 30 7d 10 4d cd 55
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 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 30 7d 10 4d cd 55
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'17eacd65a271a55f498a6e5b9805dcb79ff62f0f38038b34323b28daf74acc23'"
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 31 2c 20 22 64 61 74 61 22 3a 20 22 31 37 65 61 63 64 36 35 61 32 37 31 61 35 35
↪  66 34 39 38 61 36 65 35 62 39 38 30 35 64 63 62 37 39 66 66 36 32 66 30 66 33 38 30 33 38
↪  62 33 34 33 32 33 62 32 38 64 61 66 37 34 61 63 63 32 33 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d fa 0a 5a 6e
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 31 2c 20 22 64 61 74 61 22 3a 20 22 31 37 65 61 63 64 36 35 61 32 37 31 61 35
↪  35 66 34 39 38 61 36 65 35 62 39 38 30 35 64 63 62 37 39 66 66 36 32 66 30 66 33 38 30 33
↪  38 62 33 34 33 32 33 62 32 38 64 61 66 37 34 61 63 63 32 33 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d fa 0a 5a 6e
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "u'17eacd65a271a55f498a6e5b9805dcb79ff62f0f38038b34323b28daf74acc23'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'9892b2515138924258bb896c0f5d83e5bf5360bda4e875dbcd82fbceb72d5732bd97fc1b936c38aeb9d3aa7
↪  e10506a3f0ad1004f34b345bde35203fe165ef4ff'"
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 30 2c 20 22 64 61 74 61 22 3a 20 22 39 38 39 32 62 32 35 31 35 31 33 38 39 32 34
↪  32 35 38 62 62 38 39 36 63 30 66 35 64 38 33 65 35 62 66 35 33 36 30 62 64 61 34 65 38 37
↪  35 64 62 63 64 38 32 66 62 63 65 62 37 32 64 35 37 33 32 62 64 39 37 66 63 31 62 39 33 36
↪  63 33 38 61 65 62 39 64 33 61 61 37 65 31 30 35 30 36 61 33 66 30 61 64 31 30 30 34 66 33
↪  34 62 33 34 35 62 64 65 33 35 32 30 33 66 65 31 36 35 65 66 34 66 66 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 31 7d 2e 06 7a 1c
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 30 2c 20 22 64 61 74 61 22 3a 20 22 39 38 39 32 62 32 35 31 35 31 33 38 39 32
↪  34 32 35 38 62 62 38 39 36 63 30 66 35 64 38 33 65 35 62 66 35 33 36 30 62 64 61 34 65 38
↪  37 35 64 62 63 64 38 32 66 62 63 65 62 37 32 64 35 37 33 32 62 64 39 37 66 63 31 62 39 33
↪  36 63 33 38 61 65 62 39 64 33 61 61 37 65 31 30 35 30 36 61 33 66 30 61 64 31 30 30 34 66
↪  33 34 62 33 34 35 62 64 65 33 35 32 30 33 66 65 31 36 35 65 66 34 66 66 22 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 31 7d 2e 06 7a 1c
SP server: RX <- service: authentification request, data_id: key, status: okay, data:
↪  "u'9892b2515138924258bb896c0f5d83e5bf5360bda4e875dbcd82fbceb72d5732bd97fc1b936c38aeb9d3aa
↪  7e10506a3f0ad1004f34b345bde35203fe165ef4ff'"
SP server: RX <- Executing callback __authentificate_check_key__ to process received data
SP server: TX <- service: authentification response, data_id: key, status: okay, data: "True"
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 74 72 75 65 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 31 7d 11 d3 26 78
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 74 72 75 65 2c 20 22 64 61 74 61 5f 69 64 22 3a
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
Receive data (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
SP server: RX <- service: 17, data_id: 34, status: okay, data: "u'msg1_data_to_be_transfered'"
```

```
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Received message on server side is correct (Content {u'status': 0, u'service_id': 17, u'data': u'msg1_data_to_be_transfered', u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {u'status': 0, u'service_id': 17, u'data':
↪  u'msg1_data_to_be_transfered', u'data_id': 34} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on server side): result = {'status': 0, 'service_id': 17,
↪  'data': 'msg1_data_to_be_transfered', 'data_id': 34} (<class
↪  'socket_protocol.data_storage'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
```

```
Receive data (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
```

```
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "u'msg2_data_to_be_transfered'"
```

```
SP client: RX <- Message has a peculiar status: status: service or data unknown
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Server send Method): result = True (<type 'bool'>)
```

| | |
|---|---|
| **Success** | Received message on client side is correct (Content {u'status': 4, u'service_id': 17, u'data': u'msg2_data_to_be_transfered', u'data_id': 35} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Received message on client side): {u'status': 4, u'service_id': 17, u'data':
↪  u'msg2_data_to_be_transfered', u'data_id': 35} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on client side): result = {'status': 4, 'service_id': 17,
↪  'data': 'msg2_data_to_be_transfered', 'data_id': 35} (<class
↪  'socket_protocol.data_storage'>)
```

### A.1.10 A whitelist for communication (rx and tx) shall be available to enable communication for unauthorised counterparts

**Description**

It shall be possible to add a specific message, identified by Service-ID and Data-ID, to a whitelist. All messages added to that whitelist shall be transmitted and received, if no authentification was successfull performed.

**Reason for the implementation**

Give the user the possibility to define messages which will not be protected behind the authentification mechanism.

**Fitcriterion**

Transmition and Reception will be enabled, after the message has been added to the whitelist.

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX -> Authentification is required. Message service: 17, data_id: 34, status:
↪  okay, data: 'msg1_data_to_be_transfered' will be ignored.
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content False and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): False (<type 'bool'>)
Expectation (Returnvalue of Client send Method): result = False (<type 'bool'>)
```

**Success** Received message on server side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on server side): None (<type 'NoneType'>)
Expectation (Received message on server side): result = None (<type 'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX -> Authentification is required. Message service: 17, data_id: 35, status:
↪  service or data unknown, data: 'msg2_data_to_be_transfered' will be ignored.
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content False and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): False (<type 'bool'>)
Expectation (Returnvalue of Server send Method): result = False (<type 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on client side): None (<type 'NoneType'>)
Expectation (Received message on client side): result = None (<type 'NoneType'>)
```

**Info** Added msg1 to client whitelist (sid=17, did=34)

```
SP client: Adding Message (service: 17, data_id: 34) to the authentification whitelist
```

**Info** Transfering a message client → server

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
Send data: (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
Receive data (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
SP server: RX <- Authentification is required. Message will be ignored.
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Received message on server side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on server side): None (<type 'NoneType'>)
Expectation (Received message on server side): result = None (<type 'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX -> Authentification is required. Message service: 17, data_id: 35, status:
↪  service or data unknown, data: 'msg2_data_to_be_transfered' will be ignored.
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content False and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): False (<type 'bool'>)
Expectation (Returnvalue of Server send Method): result = False (<type 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on client side): None (<type 'NoneType'>)
Expectation (Received message on client side): result = None (<type 'NoneType'>)
```

**Info** Added msg1 to server whitelist (sid=17, did=34)

```
SP server: Adding Message (service: 17, data_id: 34) to the authentification whitelist
```

**Info** Transfering a message client → server

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
Send data: (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
Receive data (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
SP server: RX <- service: 17, data_id: 34, status: okay, data: "u'msg1_data_to_be_transfered'"
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Received message on server side is correct (Content {u'status': 0, u'service_id': 17, u'data': u'msg1_data_to_be_transfered', u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {u'status': 0, u'service_id': 17, u'data':
↪  u'msg1_data_to_be_transfered', u'data_id': 34} (<class 'socket_protocol.data_storage'>)
Expectation (Received message on server side): result = {'status': 0, 'service_id': 17,
↪  'data': 'msg1_data_to_be_transfered', 'data_id': 34} (<class
↪  'socket_protocol.data_storage'>)
```

**Info** Transfering a message server → client

```
SP server: TX -> Authentification is required. Message service: 17, data_id: 35, status:
↪  service or data unknown, data: 'msg2_data_to_be_transfered' will be ignored.
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content False and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): False (<type 'bool'>)
Expectation (Returnvalue of Server send Method): result = False (<type 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on client side): None (<type 'NoneType'>)
Expectation (Received message on client side): result = None (<type 'NoneType'>)
```

**Info** Added msg2 to client and server whitelist (sid=17, did=35)

```
SP client: Adding Message (service: 17, data_id: 35) to the authentification whitelist
SP server: Adding Message (service: 17, data_id: 35) to the authentification whitelist
```

**Info** Transfering a message client → server

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
Send data: (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
Receive data (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
SP server: RX <- service: 17, data_id: 34, status: okay, data: "u'msg1_data_to_be_transfered'"
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Received message on server side is correct (Content {u'status': 0, u'service_id': 17, u'data': u'msg1_data_to_be_transfered', u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {u'status': 0, u'service_id': 17, u'data':
↪  u'msg1_data_to_be_transfered', u'data_id': 34} (<class 'socket_protocol.data_storage'>)
Expectation (Received message on server side): result = {'status': 0, 'service_id': 17,
↪  'data': 'msg1_data_to_be_transfered', 'data_id': 34} (<class
↪  'socket_protocol.data_storage'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
Send data: (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
Receive data (88): 7b 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 73 65 72 76 69 63 65 5f 69 64
↪  22 3a 20 31 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 35 7d 20
↪  18 19 e8
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "u'msg2_data_to_be_transfered'"
SP client: RX <- Message has a peculiar status: status: service or data unknown
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): True (<type 'bool'>)
Expectation (Returnvalue of Server send Method): result = True (<type 'bool'>)
```

**Success** Received message on client side is correct (Content {u'status': 4, u'service_id': 17, u'data': u'msg2_data_to_be_transfered', u'data_id': 35} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on client side): {u'status': 4, u'service_id': 17, u'data':
↪  u'msg2_data_to_be_transfered', u'data_id': 35} (<class 'socket_protocol.data_storage'>)
Expectation (Received message on client side): result = {'status': 4, 'service_id': 17,
↪  'data': 'msg2_data_to_be_transfered', 'data_id': 35} (<class
↪  'socket_protocol.data_storage'>)
```

#### A.1.11 Define a channel name for the server and client after connection is established

**Description**

After the connection is established, the client will initiate the channel name exchange. The channel name defined on the client side will be dominant.

**Reason for the implementation**

Structured logging by creating logger childs for each channel.

**Fitcriterion**

Perform a channel name exchange with no channel name definition, differing channel name definition and identical channel name definition. In all cases, the channel name of the client will be used. Perform two channel name exchanges with only one channel name definition. This definition will be used.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 28 3b d3 54
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 38 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 28 3b d3 54
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 14 5b 30 5c
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 39 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 14 5b 30 5c
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Channel name of server is correct (Content None and Type is <type 'NoneType'>).

```
Result (Channel name of server): None (<type 'NoneType'>)
Expectation (Channel name of server): result = None (<type 'NoneType'>)
```

**Success** Channel name of client is correct (Content None and Type is <type 'NoneType'>).

```
Result (Channel name of client): None (<type 'NoneType'>)
Expectation (Channel name of client): result = None (<type 'NoneType'>)
```

**Info** Setting different Channel names for client and Server

**Info** Server and Client connect callback triggered

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, data: "'client'"
Send data: (66): 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 38 2c 20 22 64 61 74 61 22 3a 20 22 63 6c 69 65 6e 74 22 2c 20 22 64 61 74 61 5f
↪  69 64 22 3a 20 30 7d 93 56 e3 b4
Receive data (66): 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 38 2c 20 22 64 61 74 61 22 3a 20 22 63 6c 69 65 6e 74 22 2c 20 22 64 61 74 61 5f
↪  69 64 22 3a 20 30 7d 93 56 e3 b4
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "u'client'"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: overwriting user defined channel name from 'server' to u'client'
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 14 5b 30 5c
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 39 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 14 5b 30 5c
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

| | |
|---|---|
| **Success** | Channel name of server is correct (Content 'client' and Type is <type 'str'>). |

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

| | |
|---|---|
| **Success** | Channel name of client is correct (Content 'client' and Type is <type 'str'>). |

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

| | |
|---|---|
| **Info** | Setting identical Channel names for client and server |

| | |
|---|---|
| **Info** | Server and Client connect callback triggered |

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, data:
↪  "'unittest'"
Send data: (68): 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 38 2c 20 22 64 61 74 61 22 3a 20 22 75 6e 69 74 74 65 73 74 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 30 7d b0 bd 92 06
Receive data (68): 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 38 2c 20 22 64 61 74 61 22 3a 20 22 75 6e 69 74 74 65 73 74 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 30 7d b0 bd 92 06
SP server: RX <- service: channel name request, data_id: name, status: okay, data:
↪  "u'unittest'"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 14 5b 30 5c
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 39 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 14 5b 30 5c
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Channel name of server is correct (Content 'unittest' and Type is <type 'str'>).

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

**Success** Channel name of client is correct (Content 'unittest' and Type is <type 'str'>).

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

**Info** Setting Channel name for client only

**Info** Server and Client connect callback triggered

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, data: "'client'"
Send data: (66): 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 38 2c 20 22 64 61 74 61 22 3a 20 22 63 6c 69 65 6e 74 22 2c 20 22 64 61 74 61 5f
↪  69 64 22 3a 20 30 7d 93 56 e3 b4
Receive data (66): 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 38 2c 20 22 64 61 74 61 22 3a 20 22 63 6c 69 65 6e 74 22 2c 20 22 64 61 74 61 5f
↪  69 64 22 3a 20 30 7d 93 56 e3 b4
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "u'client'"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: channel name is now 'client'
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 14 5b 30 5c
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 39 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 14 5b 30 5c
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

| | |
|---|---|
| **Success** | Channel name of server is correct (Content 'client' and Type is <type 'str'>). |

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

| | |
|---|---|
| **Success** | Channel name of client is correct (Content 'client' and Type is <type 'str'>). |

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

| | |
|---|---|
| **Info** | Setting Channel name for server only |

| | |
|---|---|
| **Info** | Server and Client connect callback triggered |

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 28 3b d3 54
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 38 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 28 3b d3 54
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, data: "'server'"
Send data: (66): 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 39 2c 20 22 64 61 74 61 22 3a 20 22 73 65 72 76 65 72 22 2c 20 22 64 61 74 61 5f
↪  69 64 22 3a 20 30 7d 9c 48 3b b3
Receive data (66): 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 39 2c 20 22 64 61 74 61 22 3a 20 22 73 65 72 76 65 72 22 2c 20 22 64 61 74 61 5f
↪  69 64 22 3a 20 30 7d 9c 48 3b b3
SP client: RX <- service: channel name response, data_id: name, status: okay, data:
↪  "u'server'"
SP client: Executing callback __channel_name_response__ to process received data
SP client: channel name is now 'server'
```

**Success** Channel name of server is correct (Content 'server' and Type is <type 'str'>).

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

**Success** Channel name of client is correct (Content 'server' and Type is <type 'str'>).

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

### A.1.12 The User shall be able to define a new service

**Description**

The service is defined by a Request Service-ID and a Response Service-ID.

**Reason for the implementation**

Definition of Request and Response SIDs.

**Fitcriterion**

Define a service and check, that the server will respond on the new Service-ID. The Status shall be "Request has no callback. Data buffered.", because no callback is registered for that request.

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
Receive data (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
SP server: RX <- service: 17, data_id: 34, status: okay, data: "u'msg1_data_to_be_transfered'"
```

```
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

```
SP client: TIMEOUT (0.5s): Requested data (service_id: 18; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Received message on server side is correct (Content None and Type is <type 'NoneType'>).

```
Result (Received message on server side): None (<type 'NoneType'>)
```

```
Expectation (Received message on server side): result = None (<type 'NoneType'>)
```

**Info** Adding service to server instance for the transmit message

```
SP server: Adding Service with Request=17 and Response=18
```

**Info** Transfering a message client → server → client

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
Receive data (88): 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 37 2c 20 22 64 61 74 61 22 3a 20 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62
↪  65 5f 74 72 61 6e 73 66 65 72 65 64 22 2c 20 22 64 61 74 61 5f 69 64 22 3a 20 33 34 7d 7a
↪  6c e4 9b
```

```
SP server: RX <- service: 17, data_id: 34, status: okay, data: "u'msg1_data_to_be_transfered'"
```

```
SP server: RX <- Message with no registered callback. Sending negative response.
```

```
SP server: TX <- service: 18, data_id: 34, status: no callback for service, data buffered.,
↪  data: "None"
```

```
Send data: (64): 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 38 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 33 34 7d e8 ee d8 5c
```

```
Receive data (64): 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 38 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 33 34 7d e8 ee d8 5c
```

```
SP client: RX <- service: 18, data_id: 34, status: no callback for service, data buffered.,
↪  data: "None"
```

```
SP client: RX <- Message has a peculiar status: status: no callback for service, data
↪  buffered.
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>). |

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

| | |
|---|---|
| **Success** | Received message on server side is correct (Content {u'status': 1, u'service_id': 18, u'data': None, u'data_id': 34} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Received message on server side): {u'status': 1, u'service_id': 18, u'data': None,
↪  u'data_id': 34} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on server side): result = {'status': 1, 'service_id': 18,
↪  'data': None, 'data_id': 34} (<class 'socket_protocol.data_storage'>)
```

### A.1.13 Registration of already registered request Service-ID or response Service-ID shall not be possible

**Description**

An exception shall be raised, if a service registration with an existing request SID or response SID is performed.

**Reason for the implementation**

Changing existing services will create strange situations with already registered callbacks.

**Fitcriterion**

Catch exception for registration of existing request and response SID.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Service with Request-SID=10 and Response-SID=18 not added, because request SID is
↪  already registered
```

| | |
|---|---|
| **Success** | Expected Exception RequestSidExistsError was triggered |

| | |
|---|---|
| **Info** | Adding a service with an already registered response SID |

```
SP server: Service with Request-SID=17 and Response-SID=11 not added, because response SID is
↪  already registered
```

| | |
|---|---|
| **Success** | Expected Exception ResponseSidExistsError was triggered |

### A.1.14 It shall be possible to register a callback for a specific Service- and Data-ID

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback__' for SID=10 and DID=0
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
Send data: (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
```

```
Receive data (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {u'status': 0, u'service_id': 11, u'data': 33,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'status': 0, 'service_id': 11, 'data':
↪  33, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

**Info** Overwriting existing Callback using one with faulty return values

```
SP server: Overwriting existing callback '__callback__' for service_id (10) and data_id (0)
↪  to '__callback_error__'!
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
SP server: RX <- Executing callback __callback_error__ to process received data
SP server: RX <- Exception raised. Check callback __callback_error__ and it's return values
↪  for service_id  10 and data_id 0
SP server: TX <- service: read data response, data_id: 0, status: callback error., data:
↪  "None"
Send data: (63): 7b 22 73 74 61 74 75 73 22 3a 20 32 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 30 7d 3f 8f 7d 86
Receive data (63): 7b 22 73 74 61 74 75 73 22 3a 20 32 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 30 7d 3f 8f 7d 86
SP client: RX <- service: read data response, data_id: 0, status: callback error., data:
↪  "None"
SP client: RX <- Message has a peculiar status: status: callback error.
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {u'status': 2, u'service_id': 11, u'data': None, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {u'status': 2, u'service_id': 11, u'data': None,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
Expectation (Message received by client): result = {'status': 2, 'service_id': 11, 'data':
↪  None, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

**Info** Removing the registered Callback

```
SP server: Deleting existing callback '__callback_error__' for service_id (10) and data_id
↪  (0)!
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
SP server: RX <- Message with no registered callback. Sending negative response.
SP server: TX <- service: read data response, data_id: 0, status: no callback for service,
↪  data buffered., data: "None"
Send data: (63): 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 30 7d 79 5d 48 e2
Receive data (63): 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 30 7d 79 5d 48 e2
SP client: RX <- service: read data response, data_id: 0, status: no callback for service,
↪  data buffered., data: "None"
SP client: RX <- Message has a peculiar status: status: no callback for service, data
↪  buffered.
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Message stored inside callback is correct (Content None and Type is <type 'NoneType'>). |

```
Result (Message stored inside callback): None (<type 'NoneType'>)
Expectation (Message stored inside callback): result = None (<type 'NoneType'>)
```

| | |
|---|---|
| **Success** | Message received by client is correct (Content {u'status': 1, u'service_id': 11, u'data': None, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message received by client): {u'status': 1, u'service_id': 11, u'data': None,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
Expectation (Message received by client): result = {'status': 1, 'service_id': 11, 'data':
↪  None, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

### A.1.15 It shall be possible to register a callback for a specific Service-ID and all Data-IDs

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback__' for SID=10 and DID=None
```

| **Info** | Transfering data |
|---|---|

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
SP server: RX <- Executing callback __callback__ to process received data
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
Send data: (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
```

```
Receive data (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
|---|---|

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

| **Success** | Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |
|---|---|

```
Result (Message received by client): {u'status': 0, u'service_id': 11, u'data': 33,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'status': 0, 'service_id': 11, 'data':
↪  33, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

### A.1.16 It shall be possible to register a callback for a specific Data-IDs and all Service-IDs

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback__' for SID=None and DID=0
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
Send data: (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
```

```
Receive data (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {u'status': 0, u'service_id': 11, u'data': 33,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'status': 0, 'service_id': 11, 'data':
↪  33, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

### A.1.17 It shall be possible to register a callback for all incomming messages

**Testresult**

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

**Info** Setting up communication

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback__' for SID=None and DID=None
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
Send data: (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
```

```
Receive data (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {u'status': 0, u'service_id': 11, u'data': 33,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'status': 0, 'service_id': 11, 'data':
↪  33, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

#### A.1.18 Callback choice, if several callbacks are available (caused by wildcard callbacks)

**Testresult**

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

**Info** Setting up communication

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback3__' for SID=None and DID=None
SP server: Adding callback '__callback2__' for SID=None and DID=0
SP server: Adding callback '__callback1__' for SID=10 and DID=None
SP server: Adding callback '__callback__' for SID=10 and DID=0
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
SP server: RX <- Executing callback __callback__ to process received data
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
Send data: (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
Receive data (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 33 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 60 02 24 68
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 33, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {u'status': 0, u'service_id': 11, u'data': 33,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
Expectation (Message received by client): result = {'status': 0, 'service_id': 11, 'data':
↪  33, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

**Info** Removing Callback for a specific Data- and Service-ID

```
SP server: Deleting existing callback '__callback__' for service_id (10) and data_id (0)!
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback1__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: operation not permitted,
↪  data: "34"
```

```
Send data: (61): 7b 22 73 74 61 74 75 73 22 3a 20 36 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 33 34 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 46 3f 83 36
```

```
Receive data (61): 7b 22 73 74 61 74 75 73 22 3a 20 36 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 33 34 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 46 3f 83 36
```

```
SP client: RX <- service: read data response, data_id: 0, status: operation not permitted,
↪  data: "34"
```

```
SP client: RX <- Message has a peculiar status: status: operation not permitted
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

| | |
|---|---|
| **Success** | Message received by client is correct (Content {u'status': 6, u'service_id': 11, u'data': 34, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message received by client): {u'status': 6, u'service_id': 11, u'data': 34,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'status': 6, 'service_id': 11, 'data':
↪  34, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

| | |
|---|---|
| **Info** | Removing Callback for a specific Service-ID and all Data-IDs |

```
SP server: Deleting existing callback '__callback1__' for service_id (10) and data_id (None)!
```

| | |
|---|---|
| **Info** | Transfering data |

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback2__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: operation not permitted,
↪  data: "35"
```

```
Send data: (61): 7b 22 73 74 61 74 75 73 22 3a 20 36 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 33 35 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e8 57 12 a7
```

```
Receive data (61): 7b 22 73 74 61 74 75 73 22 3a 20 36 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 33 35 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e8 57 12 a7
```

```
SP client: RX <- service: read data response, data_id: 0, status: operation not permitted,
↪  data: "35"
```

```
SP client: RX <- Message has a peculiar status: status: operation not permitted
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

| | |
|---|---|
| **Success** | Message received by client is correct (Content {u'status': 6, u'service_id': 11, u'data': 35, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message received by client): {u'status': 6, u'service_id': 11, u'data': 35,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'status': 6, 'service_id': 11, 'data':
↪  35, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

| | |
|---|---|
| **Info** | Removing Callback for a specific Data-ID and all Serice-IDs |

```
SP server: Deleting existing callback '__callback2__' for service_id (None) and data_id (0)!
```

| | |
|---|---|
| **Info** | Transfering data |

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
Receive data (61): 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 33 31 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d e6 17 fc 16
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback3__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "36"
```

```
Send data: (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 36 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 1a 5b f9 7e
```

```
Receive data (61): 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 31 2c 20 22 64 61 74 61 22 3a 20 33 36 2c 20 22 64 61 74 61 5f 69 64 22 3a 20
↪  30 7d 1a 5b f9 7e
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "36"
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Message stored inside callback is correct (Content {u'status': 0, u'service_id': 10, u'data': 31, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message stored inside callback): {u'status': 0, u'service_id': 10, u'data': 31,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'status': 0, 'service_id': 10,
↪  'data': 31, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

| | |
|---|---|
| **Success** | Message received by client is correct (Content {u'status': 0, u'service_id': 11, u'data': 36, u'data_id': 0} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message received by client): {u'status': 0, u'service_id': 11, u'data': 36,
↪  u'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'status': 0, 'service_id': 11, 'data':
↪  36, 'data_id': 0} (<class 'socket_protocol.data_storage'>)
```

### A.1.19 Connection established information

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
Result (Client connection status): False (<type 'bool'>)
Expectation (Client connection status): result = False (<type 'bool'>)
```

**Success** Server connection status is correct (Content False and Type is <type 'bool'>).

```
Result (Server connection status): False (<type 'bool'>)
Expectation (Server connection status): result = False (<type 'bool'>)
```

**Info** Connecting Client

```
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 28 3b d3 54
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 38 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 28 3b d3 54
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 14 5b 30 5c
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 39 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 14 5b 30 5c
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Client connection status is correct (Content True and Type is <type 'bool'>).

```
Result (Client connection status): True (<type 'bool'>)
Expectation (Client connection status): result = True (<type 'bool'>)
```

**Success** Server connection status is correct (Content False and Type is <type 'bool'>).

```
Result (Server connection status): False (<type 'bool'>)
Expectation (Server connection status): result = False (<type 'bool'>)
```

**Info** Connecting Server

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

**Success** Client connection status is correct (Content True and Type is <type 'bool'>).

```
Result (Client connection status): True (<type 'bool'>)
Expectation (Client connection status): result = True (<type 'bool'>)
```

**Success** Server connection status is correct (Content True and Type is <type 'bool'>).

```
Result (Server connection status): True (<type 'bool'>)
Expectation (Server connection status): result = True (<type 'bool'>)
```

**Info** Adding secrets to socket_protocol

**Success** Client connection status is correct (Content False and Type is <type 'bool'>).

```
Result (Client connection status): False (<type 'bool'>)
Expectation (Client connection status): result = False (<type 'bool'>)
```

**Success** Server connection status is correct (Content False and Type is <type 'bool'>).

```
Result (Server connection status): False (<type 'bool'>)
Expectation (Server connection status): result = False (<type 'bool'>)
```

**Info** Doing authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 30 7d 10 4d cd 55
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 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 30 7d 10 4d cd 55
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'1028eb0571fc1cf0706cfb1a55feb927b9ace9be360493f7fde36f10254157b2'"
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 31 2c 20 22 64 61 74 61 22 3a 20 22 31 30 32 38 65 62 30 35 37 31 66 63 31 63 66
↪  30 37 30 36 63 66 62 31 61 35 35 66 65 62 39 32 37 62 39 61 63 65 39 62 65 33 36 30 34 39
↪  33 66 37 66 64 65 33 36 66 31 30 32 35 34 31 35 37 62 32 22 2c 20 22 64 61 74 61 5f 69 64
↪  22 3a 20 30 7d 30 3b c6 e3
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 31 2c 20 22 64 61 74 61 22 3a 20 22 31 30 32 38 65 62 30 35 37 31 66 63 31 63
↪  66 30 37 30 36 63 66 62 31 61 35 35 66 65 62 39 32 37 62 39 61 63 65 39 62 65 33 36 30 34
↪  39 33 66 37 66 64 65 33 36 66 31 30 32 35 34 31 35 37 62 32 22 2c 20 22 64 61 74 61 5f 69
↪  64 22 3a 20 30 7d 30 3b c6 e3
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "u'1028eb0571fc1cf0706cfb1a55feb927b9ace9be360493f7fde36f10254157b2'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'4237c44df5fc44768bb88a234bfbae24ac1b2e8f32f41c6bcf246bf462d68b2e649e419813982afd1942ab4
↪  a84b0394ae82873e1f0cb6b1ea9dc45a36c666c5c'"
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 30 2c 20 22 64 61 74 61 22 3a 20 22 34 32 33 37 63 34 34 64 66 35 66 63 34 34 37
↪  36 38 62 62 38 38 61 32 33 34 62 66 62 61 65 32 34 61 63 31 62 32 65 38 66 33 32 66 34 31
↪  63 36 62 63 66 32 34 36 62 66 34 36 32 64 36 38 62 32 65 36 34 39 65 34 31 39 38 31 33 39
↪  38 32 61 66 64 31 39 34 32 61 62 34 61 38 34 62 30 33 39 34 61 65 38 32 38 37 33 65 31 66
↪  30 63 62 36 62 31 65 61 39 64 63 34 35 61 33 36 63 36 36 36 63 35 63 22 2c 20 22 64 61 74
↪  61 5f 69 64 22 3a 20 31 7d 54 97 32 2d
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 30 2c 20 22 64 61 74 61 22 3a 20 22 34 32 33 37 63 34 34 64 66 35 66 63 34 34
↪  37 36 38 62 62 38 38 61 32 33 34 62 66 62 61 65 32 34 61 63 31 62 32 65 38 66 33 32 66 34
↪  31 63 36 62 63 66 32 34 36 62 66 34 36 32 64 36 38 62 32 65 36 34 39 65 34 31 39 38 31 33
↪  39 38 32 61 66 64 31 39 34 32 61 62 34 61 38 34 62 30 33 39 34 61 65 38 32 38 37 33 65 31
↪  66 30 63 62 36 62 31 65 61 39 64 63 34 35 61 33 36 63 36 36 36 63 35 63 22 2c 20 22 64 61
↪  74 61 5f 69 64 22 3a 20 31 7d 54 97 32 2d
SP server: RX <- service: authentification request, data_id: key, status: okay, data:
↪  "u'4237c44df5fc44768bb88a234bfbae24ac1b2e8f32f41c6bcf246bf462d68b2e649e419813982afd1942ab
↪  4a84b0394ae82873e1f0cb6b1ea9dc45a36c666c5c'"
SP server: RX <- Executing callback __authentificate_check_key__ to process received data
SP server: TX <- service: authentification response, data_id: key, status: okay, data: "True"
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 74 72 75 65 2c 20 22 64 61 74 61 5f 69 64 22 3a
↪  20 31 7d 11 d3 26 78
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 74 72 75 65 2c 20 22 64 61 74 61 5f 69 64 22 3a
```

```
Result (Client connection status): True (<type 'bool'>)
Expectation (Client connection status): result = True (<type 'bool'>)
```

| | |
|---|---|
| **Success** | Server connection status is correct (Content True and Type is <type 'bool'>). |

```
Result (Server connection status): True (<type 'bool'>)
Expectation (Server connection status): result = True (<type 'bool'>)
```

### A.1.20 Is connected information

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
Result (Client Communication instance connection status): False (<type 'bool'>)
Expectation (Client Communication instance connection status): result = False (<type 'bool'>)
```

**Success** Server Communication instance connection status is correct (Content False and Type is <type 'bool'>).

```
Result (Server Communication instance connection status): False (<type 'bool'>)
Expectation (Server Communication instance connection status): result = False (<type 'bool'>)
```

**Info** Connecting Client

```
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 28 3b d3 54
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 38 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 28 3b d3 54
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 14 5b 30 5c
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 39 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 14 5b 30 5c
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Client Communication instance connection status is correct (Content True and Type is <type 'bool'>).

```
Result (Client Communication instance connection status): True (<type 'bool'>)
Expectation (Client Communication instance connection status): result = True (<type 'bool'>)
```

**Success** Server Communication instance connection status is correct (Content False and Type is <type 'bool'>).

```
Result (Server Communication instance connection status): False (<type 'bool'>)
Expectation (Server Communication instance connection status): result = False (<type 'bool'>)
```

**Info** Connecting Server

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

**Success** Client Communication instance connection status is correct (Content True and Type is <type 'bool'>).

```
Result (Client Communication instance connection status): True (<type 'bool'>)
Expectation (Client Communication instance connection status): result = True (<type 'bool'>)
```

| **Success** | Server Communication instance connection status is correct (Content True and Type is <type 'bool'>). |
|---|---|

```
Result (Server Communication instance connection status): True (<type 'bool'>)
Expectation (Server Communication instance connection status): result = True (<type 'bool'>)
```

### A.1.21 Reconnect Method

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
Result (Reconnect executed marker): False (<type 'bool'>)
Expectation (Reconnect executed marker): result = False (<type 'bool'>)
```

**Success** Reconnect executed marker is correct (Content False and Type is <type 'bool'>).

```
Result (Reconnect executed marker): False (<type 'bool'>)
Expectation (Reconnect executed marker): result = False (<type 'bool'>)
```

**Info** Executing reconnect for Server

**Success** Reconnect executed marker is correct (Content True and Type is <type 'bool'>).

```
Result (Reconnect executed marker): True (<type 'bool'>)
Expectation (Reconnect executed marker): result = True (<type 'bool'>)
```

**Success** Reconnect executed marker is correct (Content False and Type is <type 'bool'>).

```
Result (Reconnect executed marker): False (<type 'bool'>)
Expectation (Reconnect executed marker): result = False (<type 'bool'>)
```

**Info** Executing reconnect for Client

**Success** Reconnect executed marker is correct (Content True and Type is <type 'bool'>).

```
Result (Reconnect executed marker): True (<type 'bool'>)
Expectation (Reconnect executed marker): result = True (<type 'bool'>)
```

**Success** Reconnect executed marker is correct (Content True and Type is <type 'bool'>).

```
Result (Reconnect executed marker): True (<type 'bool'>)
Expectation (Reconnect executed marker): result = True (<type 'bool'>)
```

### A.1.22 A full Message Object including the defined properties and data shall be transfered.

**Description**

Every Communication shall transfer a complete message with its content.

**Reason for the implementation**

See Reasons for every single information of the Message Object.

**Fitcriterion**

Send two different messages and compare the received message with each sent message.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
Send data: (41): 00 00 00 00 00 00 00 11 00 00 00 22 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f
↪  5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d
Receive data (41): 00 00 00 00 00 00 00 11 00 00 00 22 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f
↪  5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7e
SP server: RX <- Received message has a wrong checksum. Message will be ignored.
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Client send Method): True (<type 'bool'>)
Expectation (Returnvalue of Client send Method): result = True (<type 'bool'>)
```

**Success** Checksum Error → No message received by server is correct (Content None and Type is <type 'NoneType'>).

```
Result (Checksum Error -> No message received by server): None (<type 'NoneType'>)
Expectation (Checksum Error -> No message received by server): result = None (<type
↪  'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
Send data: (41): 00 00 00 04 00 00 00 11 00 00 00 23 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f
↪  5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7b
Receive data (41): 00 00 00 04 00 00 00 11 00 00 00 23 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f
↪  5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7b
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "u'msg2_data_to_be_transfered'"
SP client: RX <- Message has a peculiar status: status: service or data unknown
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <type 'bool'>).

```
Result (Returnvalue of Server send Method): True (<type 'bool'>)
Expectation (Returnvalue of Server send Method): result = True (<type 'bool'>)
```

**Success** Checksum Error → No message received by client is correct (Content None and Type is <type 'NoneType'>).

```
Result (Checksum Error -> No message received by client): None (<type 'NoneType'>)
Expectation (Checksum Error -> No message received by client): result = None (<type
↪  'NoneType'>)
```

# B Trace for testrun with python 3.8.5 (final)

## B.1 Tests with status Info (22)

### B.1.1 Status

**Description**

The Status shall hold some general information (in most cases it is used by the responder). Examples: Okay, Service or Data unknown, Operation not permitted, Authentification required, ...

**Reason for the implementation**

Give the possibility to transfer additional status information (e.g. to explain negative responses).

**Fitcriterion**

A Status is part of the Message Object and it is holding the Status information.

**Testresult**

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

| | |
|---|---|
| **Info** | Creating empty message object: {'data': None, 'data_id': None, 'service_id': None, 'status': None} |

| | |
|---|---|
| **Success** | status is part of the message object is correct ('status' is in the list or dict). |

```
Result (status is part of the message object): {'data': None, 'data_id': None, 'service_id':
↪  None, 'status': None} (<class 'socket_protocol.data_storage'>)
Expectation (status is part of the message object): 'status' in result
```

| | |
|---|---|
| **Info** | Creating a maximum message object: {'data': 'D', 'data_id': 'DID', 'service_id': 'SID', 'status': 'S'} |

| | |
|---|---|
| **Success** | status is part of the message object is correct ('status' is in the list or dict). |

```
Result (status is part of the message object): {'data': 'D', 'data_id': 'DID', 'service_id':
↪  'SID', 'status': 'S'} (<class 'socket_protocol.data_storage'>)
Expectation (status is part of the message object): 'status' in result
```

| | |
|---|---|
| **Success** | Content in message object for status is correct (Content 'S' and Type is <class 'str'>). |

```
Result (Content in message object for status): 'S' (<class 'str'>)
Expectation (Content in message object for status): result = 'S' (<class 'str'>)
```

### B.1.2 Service-ID

**Description**

The Service-ID shall hold information about the type of the request / corresponding response. Examples: read request, write request, read response, write response, ...

**Reason for the implementation**

Give the requestor the possibility to use different types (Services) for a transfer.

**Fitcriterion**

A Service-ID is part of the Message Object and it is holding the Service-ID information.

**Testresult**

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

| **Info** | Creating empty message object: {'data': None, 'data_id': None, 'service_id': None, 'status': None} |
|---|---|

| **Success** | service_id is part of the message object is correct ('service_id' is in the list or dict). |
|---|---|

```
Result (service_id is part of the message object): {'data': None, 'data_id': None,
↪  'service_id': None, 'status': None} (<class 'socket_protocol.data_storage'>)
Expectation (service_id is part of the message object): 'service_id' in result
```

| **Info** | Creating a maximum message object: {'data': 'D', 'data_id': 'DID', 'service_id': 'SID', 'status': 'S'} |
|---|---|

| **Success** | service_id is part of the message object is correct ('service_id' is in the list or dict). |
|---|---|

```
Result (service_id is part of the message object): {'data': 'D', 'data_id': 'DID',
↪  'service_id': 'SID', 'status': 'S'} (<class 'socket_protocol.data_storage'>)
Expectation (service_id is part of the message object): 'service_id' in result
```

| **Success** | Content in message object for service_id is correct (Content 'SID' and Type is <class 'str'>). |
|---|---|

```
Result (Content in message object for service_id): 'SID' (<class 'str'>)
Expectation (Content in message object for service_id): result = 'SID' (<class 'str'>)
```

### B.1.3 Data-ID

**Description**

The Data-ID shall hold information to differtiate the data for a specific Service.

**Reason for the implementation**

Give the possibility to transfer different information for each Service.

**Fitcriterion**

A Data-ID is part of the Message Object and it is holding the Data-ID information.

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

| | |
|---|---|
| **Info** | Creating empty message object: {'data': None, 'data_id': None, 'service_id': None, 'status': None} |

| | |
|---|---|
| **Success** | data_id is part of the message object is correct ('data_id' is in the list or dict). |

```
Result (data_id is part of the message object): {'data': None, 'data_id': None, 'service_id':
↪  None, 'status': None} (<class 'socket_protocol.data_storage'>)
Expectation (data_id is part of the message object): 'data_id' in result
```

| | |
|---|---|
| **Info** | Creating a maximum message object: {'data': 'D', 'data_id': 'DID', 'service_id': 'SID', 'status': 'S'} |

| | |
|---|---|
| **Success** | data_id is part of the message object is correct ('data_id' is in the list or dict). |

```
Result (data_id is part of the message object): {'data': 'D', 'data_id': 'DID', 'service_id':
↪  'SID', 'status': 'S'} (<class 'socket_protocol.data_storage'>)
Expectation (data_id is part of the message object): 'data_id' in result
```

| | |
|---|---|
| **Success** | Content in message object for data_id is correct (Content 'DID' and Type is <class 'str'>). |

```
Result (Content in message object for data_id): 'DID' (<class 'str'>)
Expectation (Content in message object for data_id): result = 'DID' (<class 'str'>)
```

### B.1.4 Data

**Description**
The Data shall hold the data to be transfered. For the most requests not data is transmitted.

**Reason for the implementation**
Give the possibility to transfer Data.

**Fitcriterion**
Data is part of the Message Object and it is holding the Data information.

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

| | |
|---|---|
| **Info** | Creating empty message object: {'data': None, 'data_id': None, 'service_id': None, 'status': None} |

| | |
|---|---|
| **Success** | data is part of the message object is correct ('data' is in the list or dict). |

```
Result (data is part of the message object): {'data': None, 'data_id': None, 'service_id':
↪  None, 'status': None} (<class 'socket_protocol.data_storage'>)
Expectation (data is part of the message object): 'data' in result
```

| **Info** | Creating a maximum message object: {'data': 'D', 'data_id': 'DID', 'service_id': 'SID', 'status': 'S'} |
|---|---|

| **Success** | data is part of the message object is correct ('data' is in the list or dict). |
|---|---|

```
Result (data is part of the message object): {'data': 'D', 'data_id': 'DID', 'service_id':
↪  'SID', 'status': 'S'} (<class 'socket_protocol.data_storage'>)
Expectation (data is part of the message object): 'data' in result
```

| **Success** | Content in message object for data is correct (Content 'D' and Type is <class 'str'>). |
|---|---|

```
Result (Content in message object for data): 'D' (<class 'str'>)
Expectation (Content in message object for data): result = 'D' (<class 'str'>)
```

#### B.1.5 A full Message Object including the defined properties and data shall be transfered.

**Description**

Every Communication shall transfer a complete message with its content.

**Reason for the implementation**

See Reasons for every single information of the Message Object.

**Fitcriterion**

Send two different messages and compare the received message with each sent message.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
SP server: RX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Received message on server side is correct (Content {'data_id': 34, 'service_id': 17, 'status': 0, 'data': 'msg1_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {'data_id': 34, 'service_id': 17, 'status': 0,
↪  'data': 'msg1_data_to_be_transfered'} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on server side): result = {'service_id': 17, 'data_id': 34,
↪  'status': 0, 'data': 'msg1_data_to_be_transfered'} (<class
↪  'socket_protocol.data_storage'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
```

```
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
SP client: RX <- Message has a peculiar status: status: service or data unknown
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Server send Method): result = True (<class 'bool'>)
```

| | |
|---|---|
| **Success** | Received message on client side is correct (Content {'data_id': 35, 'service_id': 17, 'status': 4, 'data': 'msg2_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Received message on client side): {'data_id': 35, 'service_id': 17, 'status': 4,
↪  'data': 'msg2_data_to_be_transfered'} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on client side): result = {'service_id': 17, 'data_id': 35,
↪  'status': 4, 'data': 'msg2_data_to_be_transfered'} (<class
↪  'socket_protocol.data_storage'>)
```

### B.1.6 A checksumm shall ensure the correct transmition

**Description**

If the checksum does not fit to the checksum of the transferred data, the message will be ignored, because the complete content including the Service- and Data-ID is possibly corrupted.

**Reason for the implementation**

Ensure correct data transfer.

**Fitcriterion**

Corrupted message is not in the receive buffer after transmission.

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1c
```

```
SP server: RX <- Received message has a wrong checksum. Message will be ignored.
```

```
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Checksum Error → No message received by server is correct (Content None and Type is <class 'NoneType'>).

```
Result (Checksum Error -> No message received by server): None (<class 'NoneType'>)
```

```
Expectation (Checksum Error -> No message received by server): result = None (<class
↪  'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
```

```
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
SP client: RX <- Message has a peculiar status: status: service or data unknown
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

```
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): True (<class 'bool'>)
Expectation (Returnvalue of Server send Method): result = True (<class 'bool'>)
```

| | |
|---|---|
| **Success** | Checksum Error → No message received by client is correct (Content None and Type is <class 'NoneType'>). |

```
Result (Checksum Error -> No message received by client): None (<class 'NoneType'>)
Expectation (Checksum Error -> No message received by client): result = None (<class
↪  'NoneType'>)
```

### B.1.7 An authentification between server and client shall be possible including status feedback methods

**Description**

The Client shall have a method to initiate the authentification. In case that the server and the client do have identical secrets, the authentification shall be successfull.

**Reason for the implementation**

Message protection (e.g. for secure functions or data)

**Fitcriterion**

Check authentification method feedback (client) and authentification feedback (client and server), in case of differing and identical secrets.

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

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

**Success** Authentification state of server is correct (Content True and Type is <class 'bool'>).

```
Result (Authentification state of server): True (<class 'bool'>)
Expectation (Authentification state of server): result = True (<class 'bool'>)
```

**Success** Authentification state of client is correct (Content True and Type is <class 'bool'>).

```
Result (Authentification state of client): True (<class 'bool'>)
Expectation (Authentification state of client): result = True (<class 'bool'>)
```

**Info** Different secrets set

**Success** Authentification state of server is correct (Content False and Type is <class 'bool'>).

```
Result (Authentification state of server): False (<class 'bool'>)
Expectation (Authentification state of server): result = False (<class 'bool'>)
```

**Success** Authentification state of client is correct (Content False and Type is <class 'bool'>).

```
Result (Authentification state of client): False (<class 'bool'>)
Expectation (Authentification state of client): result = False (<class 'bool'>)
```

**Info** Performing Authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 fd 82 a2 a9
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 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 fd 82 a2 a9
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'84abf044252020404e0def53e859f099dff329af1ef95e7557b8a2c514d6b776'"
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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 38
↪  34 61 62 66 30 34 34 32 35 32 30 32 30 34 30 34 65 30 64 65 66 35 33 65 38 35 39 66 30 39
↪  39 64 66 66 33 32 39 61 66 31 65 66 39 35 65 37 35 35 37 62 38 61 32 63 35 31 34 64 36 62
↪  37 37 36 22 7d de f9 e9 e1
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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  38 34 61 62 66 30 34 34 32 35 32 30 32 30 34 30 34 65 30 64 65 66 35 33 65 38 35 39 66 30
↪  39 39 64 66 66 33 32 39 61 66 31 65 66 39 35 65 37 35 35 37 62 38 61 32 63 35 31 34 64 36
↪  62 37 37 36 22 7d de f9 e9 e1
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'84abf044252020404e0def53e859f099dff329af1ef95e7557b8a2c514d6b776'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'233acc33232b34a93d7e85c2dcd3846997d4ad446326385a2949a5ee46df3155c296e0a424895a13c383760
↪  10e46bee1f183b94245344c740b9f3770ef9d3edb'"
Send data: (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 30 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
↪  33 33 61 63 63 33 33 32 33 32 62 33 34 61 39 33 64 37 65 38 35 63 32 64 63 64 33 38 34 36
↪  39 39 37 64 34 61 64 34 34 36 33 32 36 33 38 35 61 32 39 34 39 61 35 65 65 34 36 64 66 33
↪  31 35 35 63 32 39 36 65 30 61 34 32 34 38 39 35 61 31 33 63 33 38 33 37 36 30 31 30 65 34
↪  36 62 65 65 31 66 31 38 33 62 39 34 32 34 35 33 34 34 63 37 34 30 62 39 66 33 37 37 30 65
↪  66 39 64 33 65 64 62 22 7d f3 0a 81 cb
Receive data (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 30 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 33 33 61 63 63 33 33 32 33 32 62 33 34 61 39 33 64 37 65 38 35 63 32 64 63 64 33 38 34
↪  36 39 39 37 64 34 61 64 34 34 36 33 32 36 33 38 35 61 32 39 34 39 61 35 65 65 34 36 64 66
↪  33 31 35 35 63 32 39 36 65 30 61 34 32 34 38 39 35 61 31 33 63 33 38 33 37 36 30 31 30 65
↪  34 36 62 65 65 31 66 31 38 33 62 39 34 32 34 35 33 34 34 63 37 34 30 62 39 66 33 37 37 30
↪  65 66 39 64 33 65 64 62 22 7d f3 0a 81 cb
SP server: RX <- service: authentification request, data_id: key, status: okay, data:
↪  "'233acc33232b34a93d7e85c2dcd3846997d4ad446326385a2949a5ee46df3155c296e0a424895a13c383760
↪  10e46bee1f183b94245344c740b9f3770ef9d3edb'"
SP server: RX <- Executing callback __authentificate_check_key__ to process received data
SP server: TX <- service: authentification response, data_id: key, status: okay, data: "False"
Send data: (63): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 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 66 61
↪  6c 73 65 7d ea 0a 5c b4
Receive data (63): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 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 66 61
```

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

**Success** Authentification state of server is correct (Content False and Type is <class 'bool'>).

```
Result (Authentification state of server): False (<class 'bool'>)
Expectation (Authentification state of server): result = False (<class 'bool'>)
```

**Success** Authentification state of client is correct (Content False and Type is <class 'bool'>).

```
Result (Authentification state of client): False (<class 'bool'>)
Expectation (Authentification state of client): result = False (<class 'bool'>)
```

**Info** Identical secrets set

**Info** Performing Authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 fd 82 a2 a9
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 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 fd 82 a2 a9
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'b57aec62234fe96f003daf58d45caa885979c4fb9c328a975bc068efbc455ce2'"
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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 62
↪  35 37 61 65 63 36 32 32 33 34 66 65 39 36 66 30 30 33 64 61 66 35 38 64 34 35 63 61 61 38
↪  38 35 39 37 39 63 34 66 62 39 63 33 32 38 61 39 37 35 62 63 30 36 38 65 66 62 63 34 35 35
↪  63 65 32 22 7d 37 a9 b8 63
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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  62 35 37 61 65 63 36 32 32 33 34 66 65 39 36 66 30 30 33 64 61 66 35 38 64 34 35 63 61 61
↪  38 38 35 39 37 39 63 34 66 62 39 63 33 32 38 61 39 37 35 62 63 30 36 38 65 66 62 63 34 35
↪  35 63 65 32 22 7d 37 a9 b8 63
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'b57aec62234fe96f003daf58d45caa885979c4fb9c328a975bc068efbc455ce2'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'89cccf6a7300e968bc7eb90ffe7cb05897a65700e13bf8594657757783328a5f1efd7be32c8a0b5b461b6c8
↪  fe6af704fb3fa5faeedf3ea9473b3aa6c1007e648'"
Send data: (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 38
↪  39 63 63 63 66 36 61 37 33 30 30 65 39 36 38 62 63 37 65 62 39 30 66 66 65 37 63 62 30 35
↪  38 39 37 61 36 35 37 30 30 65 31 33 62 66 38 35 39 34 36 35 37 37 35 37 37 38 33 33 32 38
↪  61 35 66 31 65 66 64 37 62 65 33 32 63 38 61 30 62 35 62 34 36 31 62 36 63 38 66 65 36 61
↪  66 37 30 34 66 62 33 66 61 35 66 61 65 65 64 66 33 65 61 39 34 37 33 62 33 61 61 36 63 31
↪  30 30 37 65 36 34 38 22 7d 13 b4 0f 76
Receive data (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  38 39 63 63 63 66 36 61 37 33 30 30 65 39 36 38 62 63 37 65 62 39 30 66 66 65 37 63 62 30
↪  35 38 39 37 61 36 35 37 30 30 65 31 33 62 66 38 35 39 34 36 35 37 37 35 37 37 38 33 33 32
↪  38 61 35 66 31 65 66 64 37 62 65 33 32 63 38 61 30 62 35 62 34 36 31 62 36 63 38 66 65 36
↪  61 66 37 30 34 66 62 33 66 61 35 66 61 65 65 64 66 33 65 61 39 34 37 33 62 33 61 61 36 63
↪  31 30 30 37 65 36 34 38 22 7d 13 b4 0f 76
SP server: RX <- service: authentification request, data_id: key, status: okay, data:
↪  "'89cccf6a7300e968bc7eb90ffe7cb05897a65700e13bf8594657757783328a5f1efd7be32c8a0b5b461b6c8
↪  fe6af704fb3fa5faeedf3ea9473b3aa6c1007e648'"
SP server: RX <- Executing callback __authentificate_check_key__ to process received data
SP server: TX <- service: authentification response, data_id: key, status: okay, data: "True"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 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 74 72
↪  75 65 7d 94 fe 74 32
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 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 74 72
```

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

**Success** Authentification state of server is correct (Content True and Type is <class 'bool'>).

```
Result (Authentification state of server): True (<class 'bool'>)
Expectation (Authentification state of server): result = True (<class 'bool'>)
```

**Success** Authentification state of client is correct (Content True and Type is <class 'bool'>).

```
Result (Authentification state of client): True (<class 'bool'>)
Expectation (Authentification state of client): result = True (<class 'bool'>)
```

**Info** Corrupting the authentification mechanism

```
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
```

**Info** Performing Authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 fd 82 a2 a9
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 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 fd 82 a2 a9
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: Executing callback __authentificate_create_seed__ to process received data
```

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

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

**Success** Authentification state of server is correct (Content False and Type is <class 'bool'>).

```
Result (Authentification state of server): False (<class 'bool'>)
Expectation (Authentification state of server): result = False (<class 'bool'>)
```

**Success** Authentification state of client is correct (Content False and Type is <class 'bool'>).

```
Result (Authentification state of client): False (<class 'bool'>)
Expectation (Authentification state of client): result = False (<class 'bool'>)
```

### B.1.8 An automatic authentification shall available

**Description**

An authentification is executed by the client on every connect.

**Reason for the implementation**

Simplify handling for authentification.

**Fitcriterion**

Check authentification feedback (client and server) after connect has been triggered.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
Result (Authentification state of server): False (<class 'bool'>)
Expectation (Authentification state of server): result = False (<class 'bool'>)
```

**Success** Authentification state of client is correct (Content False and Type is <class 'bool'>).

```
Result (Authentification state of client): False (<class 'bool'>)
Expectation (Authentification state of client): result = False (<class 'bool'>)
```

**Info** Server and Client connect callback triggered

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 53 5e 67 0b
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 38 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 53 5e 67 0b
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 30 59 be 2f
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 39 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 30 59 be 2f
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 fd 82 a2 a9
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 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 fd 82 a2 a9
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'4b113d63f8abce877c47ce12b08f6853f69d661539f981177eeedbd742eebad6'"
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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 34
↪  62 31 31 33 64 36 33 66 38 61 62 63 65 38 37 37 63 34 37 63 65 31 32 62 30 38 66 36 38 35
↪  33 66 36 39 64 36 36 31 35 33 39 66 39 38 31 31 37 37 65 65 65 64 62 64 37 34 32 65 65 62
↪  61 64 36 22 7d df f9 3b be
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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  34 62 31 31 33 64 36 33 66 38 61 62 63 65 38 37 37 63 34 37 63 65 31 32 62 30 38 66 36 38
↪  35 33 66 36 39 64 36 36 31 35 33 39 66 39 38 31 31 37 37 65 65 65 64 62 64 37 34 32 65 65
↪  62 61 64 36 22 7d df f9 3b be
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'4b113d63f8abce877c47ce12b08f6853f69d661539f981177eeedbd742eebad6'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'42fe894ec6b6928e3998e3019538576f3abea09f26af28523ef41bd8dff89522e05cc05cd777a69a0b74f67
↪  d3ed4fb5e6af25bb054e2744afc20f7451b1d3886'"
Send data: (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 34
↪  32 66 65 38 39 34 65 63 36 62 36 39 32 38 65 33 39 39 38 65 33 30 31 39 35 33 38 35 37 36
↪  66 33 61 62 65 61 30 39 66 32 36 61 66 32 38 35 32 33 65 66 34 31 62 64 38 64 66 66 38 39
↪  35 32 32 65 30 35 63 63 30 35 63 64 37 37 37 61 36 39 61 30 62 37 34 66 36 37 64 33 65 64
```

```
Result (Authentification state of server): True (<class 'bool'>)
Expectation (Authentification state of server): result = True (<class 'bool'>)
```

| **Success** | Authentification state of client is correct (Content True and Type is <class 'bool'>). |
|---|---|

```
Result (Authentification state of client): True (<class 'bool'>)
Expectation (Authentification state of client): result = True (<class 'bool'>)
```

### B.1.9 Communication (rx and tx) shall be disabled, if a secret is given but no authentification had been successfully performed.

**Description**

Communication (rx and tx) shall be disabled, if a secret is given. Except of a response for registered services, saying that a Authentification is required.

**Reason for the implementation**

Message protection (e.g. for secure functions or data)

**Fitcriterion**

RX and TX is not possible, till a successfull authentification has been performed.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: execute request, data_id: 36, status: okay, data:
↪  "'msg3_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 36 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 33 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 33 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 13
↪  e9 64 3d
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 36 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 33 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 33 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 13
↪  e9 64 3d
```

```
SP server: RX <- Authentification is required. Just sending negative response.
```

```
SP server: TX <- service: execute response, data_id: 36, status: authentification required,
↪  data: "None"
```

```
Send data: (64): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 36 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 33 31 2c 20 22 73 74 61 74 75 73 22 3a 20 33 2c 20 22 64 61 74 61 22 3a 20
↪  6e 75 6c 6c 7d 5d 78 af a4
```

```
Receive data (64): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 36 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 33 31 2c 20 22 73 74 61 74 75 73 22 3a 20 33 2c 20 22 64 61 74 61 22 3a 20
↪  6e 75 6c 6c 7d 5d 78 af a4
```

```
SP client: RX <- service: execute response, data_id: 36, status: authentification required,
↪  data: "None"
```

```
SP client: RX <- Message has a peculiar status: status: authentification required
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Received message on server side is correct (Content {'data_id': 36, 'service_id': 31, 'status': 3, 'data': None} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {'data_id': 36, 'service_id': 31, 'status': 3,
↪  'data': None} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on server side): result = {'service_id': 31, 'data_id': 36,
↪  'status': 3, 'data': None} (<class 'socket_protocol.data_storage'>)
```

**Info** Setting no Server secret but a Client secret

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
```

```
SP client: RX <- Authentification is required. Message will be ignored.
```

```
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Server send Method): result = True (<class 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on client side): None (<class 'NoneType'>)
```

```
Expectation (Received message on client side): result = None (<class 'NoneType'>)
```

**Info** Identical secrets set

**Info** Transfering a message client → server

```
SP client: TX -> Authentification is required. Message service: 17, data_id: 34, status:
↪  okay, data: 'msg1_data_to_be_transfered' will be ignored.
```

```
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content False and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): False (<class 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = False (<class 'bool'>)
```

**Success** Received message on server side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on server side): None (<class 'NoneType'>)
```

```
Expectation (Received message on server side): result = None (<class 'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX -> Authentification is required. Message service: 17, data_id: 35, status:
↪  service or data unknown, data: 'msg2_data_to_be_transfered' will be ignored.
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content False and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): False (<class 'bool'>)
Expectation (Returnvalue of Server send Method): result = False (<class 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on client side): None (<class 'NoneType'>)
Expectation (Received message on client side): result = None (<class 'NoneType'>)
```

**Info** Performing Authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 fd 82 a2 a9
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 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 fd 82 a2 a9
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'41f7a354bad5b87f222af96df7699c2ca907ea76a307ca31303b1ae2798f21b4'"
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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 34
↪  31 66 37 61 33 35 34 62 61 64 35 62 38 37 66 32 32 32 61 66 39 36 64 66 37 36 39 39 63 32
↪  63 61 39 30 37 65 61 37 36 61 33 30 37 63 61 33 31 33 30 33 62 31 61 65 32 37 39 38 66 32
↪  31 62 34 22 7d 6d 22 3f b7
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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  34 31 66 37 61 33 35 34 62 61 64 35 62 38 37 66 32 32 32 61 66 39 36 64 66 37 36 39 39 63
↪  32 63 61 39 30 37 65 61 37 36 61 33 30 37 63 61 33 31 33 30 33 62 31 61 65 32 37 39 38 66
↪  32 31 62 34 22 7d 6d 22 3f b7
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'41f7a354bad5b87f222af96df7699c2ca907ea76a307ca31303b1ae2798f21b4'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'6485a87794493911204e7f17d57cdcc3c043590ff6793a2f8b97c47c5c5da24f7cb1aa1b1fc4996a6224091
↪  528c4577fb9531128dfb829d214351bcc7eb46275'"
Send data: (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 36
↪  34 38 35 61 38 37 37 39 34 34 39 33 39 31 31 32 30 34 65 37 66 31 37 64 35 37 63 64 63 63
↪  33 63 30 34 33 35 39 30 66 66 36 37 39 33 61 32 66 38 62 39 37 63 34 37 63 35 63 35 64 61
↪  32 34 66 37 63 62 31 61 61 31 62 31 66 63 34 39 39 36 61 36 32 32 34 30 39 31 35 32 38 63
↪  34 35 37 37 66 62 39 35 33 31 31 32 38 64 66 62 38 32 39 64 32 31 34 33 35 31 62 63 63 37
↪  65 62 34 36 32 37 35 22 7d 83 7d e8 6d
Receive data (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  36 34 38 35 61 38 37 37 39 34 34 39 33 39 31 31 32 30 34 65 37 66 31 37 64 35 37 63 64 63
↪  63 33 63 30 34 33 35 39 30 66 66 36 37 39 33 61 32 66 38 62 39 37 63 34 37 63 35 63 35 64
↪  61 32 34 66 37 63 62 31 61 61 31 62 31 66 63 34 39 39 36 61 36 32 32 34 30 39 31 35 32 38
↪  63 34 35 37 37 66 62 39 35 33 31 31 32 38 64 66 62 38 32 39 64 32 31 34 33 35 31 62 63 63
↪  37 65 62 34 36 32 37 35 22 7d 83 7d e8 6d
SP server: RX <- service: authentification request, data_id: key, status: okay, data:
↪  "'6485a87794493911204e7f17d57cdcc3c043590ff6793a2f8b97c47c5c5da24f7cb1aa1b1fc4996a6224091
↪  528c4577fb9531128dfb829d214351bcc7eb46275'"
SP server: RX <- Executing callback __authentificate_check_key__ to process received data
SP server: TX <- service: authentification response, data_id: key, status: okay, data: "True"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 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 74 72
↪  75 65 7d 94 fe 74 32
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 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 74 72
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
SP server: RX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Received message on server side is correct (Content {'data_id': 34, 'service_id': 17, 'status': 0, 'data': 'msg1_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {'data_id': 34, 'service_id': 17, 'status': 0,
↪  'data': 'msg1_data_to_be_transfered'} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on server side): result = {'service_id': 17, 'data_id': 34,
↪  'status': 0, 'data': 'msg1_data_to_be_transfered'} (<class
↪  'socket_protocol.data_storage'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
```

```
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
```

```
SP client: RX <- Message has a peculiar status: status: service or data unknown
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Server send Method): result = True (<class 'bool'>)
```

| | |
|---|---|
| **Success** | Received message on client side is correct (Content {'data_id': 35, 'service_id': 17, 'status': 4, 'data': 'msg2_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Received message on client side): {'data_id': 35, 'service_id': 17, 'status': 4,
↪  'data': 'msg2_data_to_be_transfered'} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on client side): result = {'service_id': 17, 'data_id': 35,
↪  'status': 4, 'data': 'msg2_data_to_be_transfered'} (<class
↪  'socket_protocol.data_storage'>)
```

### B.1.10 A whitelist for communication (rx and tx) shall be available to enable communication for unauthorised counterparts

**Description**

It shall be possible to add a specific message, identified by Service-ID and Data-ID, to a whitelist. All messages added to that whitelist shall be transmitted and received, if no authentification was successfull performed.

**Reason for the implementation**

Give the user the possibility to define messages which will not be protected behind the authentification mechanism.

**Fitcriterion**

Transmition and Reception will be enabled, after the message has been added to the whitelist.

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX -> Authentification is required. Message service: 17, data_id: 34, status:
↪  okay, data: 'msg1_data_to_be_transfered' will be ignored.
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content False and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): False (<class 'bool'>)
Expectation (Returnvalue of Client send Method): result = False (<class 'bool'>)
```

**Success** Received message on server side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on server side): None (<class 'NoneType'>)
Expectation (Received message on server side): result = None (<class 'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX -> Authentification is required. Message service: 17, data_id: 35, status:
↪  service or data unknown, data: 'msg2_data_to_be_transfered' will be ignored.
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content False and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): False (<class 'bool'>)
Expectation (Returnvalue of Server send Method): result = False (<class 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on client side): None (<class 'NoneType'>)
Expectation (Received message on client side): result = None (<class 'NoneType'>)
```

**Info** Added msg1 to client whitelist (sid=17, did=34)

```
SP client: Adding Message (service: 17, data_id: 34) to the authentification whitelist
```

**Info** Transfering a message client → server

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
SP server: RX <- Authentification is required. Message will be ignored.
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Received message on server side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on server side): None (<class 'NoneType'>)
Expectation (Received message on server side): result = None (<class 'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX -> Authentification is required. Message service: 17, data_id: 35, status:
↪  service or data unknown, data: 'msg2_data_to_be_transfered' will be ignored.
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content False and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): False (<class 'bool'>)
Expectation (Returnvalue of Server send Method): result = False (<class 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on client side): None (<class 'NoneType'>)
Expectation (Received message on client side): result = None (<class 'NoneType'>)
```

**Info** Added msg1 to server whitelist (sid=17, did=34)

```
SP server: Adding Message (service: 17, data_id: 34) to the authentification whitelist
```

**Info** Transfering a message client → server

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
SP server: RX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Received message on server side is correct (Content {'data_id': 34, 'service_id': 17, 'status': 0, 'data': 'msg1_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {'data_id': 34, 'service_id': 17, 'status': 0,
↪  'data': 'msg1_data_to_be_transfered'} (<class 'socket_protocol.data_storage'>)
Expectation (Received message on server side): result = {'service_id': 17, 'data_id': 34,
↪  'status': 0, 'data': 'msg1_data_to_be_transfered'} (<class
↪  'socket_protocol.data_storage'>)
```

**Info** Transfering a message server → client

```
SP server: TX -> Authentification is required. Message service: 17, data_id: 35, status:
↪  service or data unknown, data: 'msg2_data_to_be_transfered' will be ignored.
SP client: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content False and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): False (<class 'bool'>)
Expectation (Returnvalue of Server send Method): result = False (<class 'bool'>)
```

**Success** Received message on client side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on client side): None (<class 'NoneType'>)
Expectation (Received message on client side): result = None (<class 'NoneType'>)
```

**Info** Added msg2 to client and server whitelist (sid=17, did=35)

```
SP client: Adding Message (service: 17, data_id: 35) to the authentification whitelist
SP server: Adding Message (service: 17, data_id: 35) to the authentification whitelist
```

**Info** Transfering a message client → server

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
SP server: RX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Received message on server side is correct (Content {'data_id': 34, 'service_id': 17, 'status': 0, 'data': 'msg1_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on server side): {'data_id': 34, 'service_id': 17, 'status': 0,
↪  'data': 'msg1_data_to_be_transfered'} (<class 'socket_protocol.data_storage'>)
Expectation (Received message on server side): result = {'service_id': 17, 'data_id': 34,
↪  'status': 0, 'data': 'msg1_data_to_be_transfered'} (<class
↪  'socket_protocol.data_storage'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 35 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 34 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 32 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 73
↪  e9 96 7f
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
SP client: RX <- Message has a peculiar status: status: service or data unknown
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): True (<class 'bool'>)
Expectation (Returnvalue of Server send Method): result = True (<class 'bool'>)
```

**Success** Received message on client side is correct (Content {'data_id': 35, 'service_id': 17, 'status': 4, 'data': 'msg2_data_to_be_transfered'} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Received message on client side): {'data_id': 35, 'service_id': 17, 'status': 4,
↪  'data': 'msg2_data_to_be_transfered'} (<class 'socket_protocol.data_storage'>)
Expectation (Received message on client side): result = {'service_id': 17, 'data_id': 35,
↪  'status': 4, 'data': 'msg2_data_to_be_transfered'} (<class
↪  'socket_protocol.data_storage'>)
```

#### B.1.11 Define a channel name for the server and client after connection is established

**Description**

After the connection is established, the client will initiate the channel name exchange. The channel name defined on the client side will be dominant.

**Reason for the implementation**

Structured logging by creating logger childs for each channel.

**Fitcriterion**

Perform a channel name exchange with no channel name definition, differing channel name definition and identical channel name definition. In all cases, the channel name of the client will be used. Perform two channel name exchanges with only one channel name definition. This definition will be used.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 53 5e 67 0b
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 38 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 53 5e 67 0b
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 30 59 be 2f
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 39 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 30 59 be 2f
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Channel name of server is correct (Content None and Type is <class 'NoneType'>).

```
Result (Channel name of server): None (<class 'NoneType'>)
Expectation (Channel name of server): result = None (<class 'NoneType'>)
```

**Success** Channel name of client is correct (Content None and Type is <class 'NoneType'>).

```
Result (Channel name of client): None (<class 'NoneType'>)
Expectation (Channel name of client): result = None (<class 'NoneType'>)
```

**Info** Setting different Channel names for client and Server

**Info** Server and Client connect callback triggered

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, data: "'client'"
Send data: (66): 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 38 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 63
↪  6c 69 65 6e 74 22 7d ee af 7b 7e
Receive data (66): 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 38 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 63
↪  6c 69 65 6e 74 22 7d ee af 7b 7e
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "'client'"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: overwriting user defined channel name from 'server' to 'client'
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 30 59 be 2f
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 39 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 30 59 be 2f
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Channel name of server is correct (Content 'client' and Type is <class 'str'>).

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

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

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

**Info** Setting identical Channel names for client and server

**Info** Server and Client connect callback triggered

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, data:
↪  "'unittest'"
Send data: (68): 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 38 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
↪  6e 69 74 74 65 73 74 22 7d f8 f6 c9 e9
Receive data (68): 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 38 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
↪  6e 69 74 74 65 73 74 22 7d f8 f6 c9 e9
SP server: RX <- service: channel name request, data_id: name, status: okay, data:
↪  "'unittest'"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 30 59 be 2f
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 39 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 30 59 be 2f
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Channel name of server is correct (Content 'unittest' and Type is <class 'str'>).

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

**Success** Channel name of client is correct (Content 'unittest' and Type is <class 'str'>).

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

**Info** Setting Channel name for client only

**Info** Server and Client connect callback triggered

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, data: "'client'"
Send data: (66): 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 38 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 63
↪  6c 69 65 6e 74 22 7d ee af 7b 7e
Receive data (66): 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 38 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 63
↪  6c 69 65 6e 74 22 7d ee af 7b 7e
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "'client'"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: channel name is now 'client'
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 30 59 be 2f
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 39 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 30 59 be 2f
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Channel name of server is correct (Content 'client' and Type is <class 'str'>).

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

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

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

**Info** Setting Channel name for server only

**Info** Server and Client connect callback triggered

```
SP server: Cleaning up receive-buffer
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 53 5e 67 0b
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 38 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 53 5e 67 0b
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, data: "'server'"
Send data: (66): 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 39 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 73
↪  65 72 76 65 72 22 7d ac a3 7b cc
Receive data (66): 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 39 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 73
↪  65 72 76 65 72 22 7d ac a3 7b cc
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "'server'"
SP client: Executing callback __channel_name_response__ to process received data
SP client: channel name is now 'server'
```

| | |
|---|---|
| **Success** | Channel name of server is correct (Content 'server' and Type is <class 'str'>). |

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

| | |
|---|---|
| **Success** | Channel name of client is correct (Content 'server' and Type is <class 'str'>). |

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

#### B.1.12 The User shall be able to define a new service

**Description**

The service is defined by a Request Service-ID and a Response Service-ID.

**Reason for the implementation**

Definition of Request and Response SIDs.

**Fitcriterion**

Define a service and check, that the server will respond on the new Service-ID. The Status shall be "Request has no callback. Data buffered.", because no callback is registered for that request.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
SP server: RX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
SP server: Message data is stored in buffer and is now ready to be retrieved by receive method
```

```
SP client: TIMEOUT (0.5s): Requested data (service_id: 18; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Received message on server side is correct (Content None and Type is <class 'NoneType'>).

```
Result (Received message on server side): None (<class 'NoneType'>)
```

```
Expectation (Received message on server side): result = None (<class 'NoneType'>)
```

**Info** Adding service to server instance for the transmit message

```
SP server: Adding Service with Request=17 and Response=18
```

**Info** Transfering a message client → server → client

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
Send data: (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
Receive data (88): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 37 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20
↪  22 6d 73 67 31 5f 64 61 74 61 5f 74 6f 5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d 4c
↪  bc bd 1b
```

```
SP server: RX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
```

```
SP server: RX <- Message with no registered callback. Sending negative response.
```

```
SP server: TX <- service: 18, data_id: 34, status: no callback for service, data buffered.,
↪  data: "None"
```

```
Send data: (64): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 38 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 bd 30 46 9b
```

```
Receive data (64): 7b 22 64 61 74 61 5f 69 64 22 3a 20 33 34 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 31 38 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 bd 30 46 9b
```

```
SP client: RX <- service: 18, data_id: 34, status: no callback for service, data buffered.,
↪  data: "None"
```

```
SP client: RX <- Message has a peculiar status: status: no callback for service, data
↪  buffered.
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>). |

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
```

```
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

| | |
|---|---|
| **Success** | Received message on server side is correct (Content {'data_id': 34, 'service_id': 18, 'status': 1, 'data': None} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Received message on server side): {'data_id': 34, 'service_id': 18, 'status': 1,
↪  'data': None} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Received message on server side): result = {'service_id': 18, 'data_id': 34,
↪  'status': 1, 'data': None} (<class 'socket_protocol.data_storage'>)
```

### B.1.13 Registration of already registered request Service-ID or response Service-ID shall not be possible

**Description**

An exception shall be raised, if a service registration with an existing request SID or response SID is performed.

**Reason for the implementation**

Changing existing services will create strange situations with already registered callbacks.

**Fitcriterion**

Catch exception for registration of existing request and response SID.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Service with Request-SID=10 and Response-SID=18 not added, because request SID is
↪  already registered
```

| | |
|---|---|
| **Success** | Expected Exception RequestSidExistsError was triggered |

| | |
|---|---|
| **Info** | Adding a service with an already registered response SID |

```
SP server: Service with Request-SID=17 and Response-SID=11 not added, because response SID is
↪  already registered
```

| | |
|---|---|
| **Success** | Expected Exception ResponseSidExistsError was triggered |

### B.1.14 It shall be possible to register a callback for a specific Service- and Data-ID

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback__' for SID=10 and DID=0
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
Send data: (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
```

```
Receive data (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 0, 'data':
↪  33} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'data': 33, 'data_id': 0, 'service_id':
↪  11, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

**Info** Overwriting existing Callback using one with faulty return values

```
SP server: Overwriting existing callback '__callback__' for service_id (10) and data_id (0)
↪  to '__callback_error__'!
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
SP server: RX <- Executing callback __callback_error__ to process received data
SP server: RX <- Exception raised. Check callback __callback_error__ and it's return values
↪  for service_id  10 and data_id 0
SP server: TX <- service: read data response, data_id: 0, status: callback error., data:
↪  "None"
Send data: (63): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 32 2c 20 22 64 61 74 61 22 3a 20 6e
↪  75 6c 6c 7d a1 a2 87 f3
Receive data (63): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 32 2c 20 22 64 61 74 61 22 3a 20 6e
↪  75 6c 6c 7d a1 a2 87 f3
SP client: RX <- service: read data response, data_id: 0, status: callback error., data:
↪  "None"
SP client: RX <- Message has a peculiar status: status: callback error.
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

| | |
|---|---|
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 2, 'data': None} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 2, 'data':
↪  None} (<class 'socket_protocol.data_storage'>)
Expectation (Message received by client): result = {'data': None, 'data_id': 0, 'service_id':
↪  11, 'status': 2} (<class 'socket_protocol.data_storage'>)
```

| | |
|---|---|
| **Info** | Removing the registered Callback |

```
SP server: Deleting existing callback '__callback_error__' for service_id (10) and data_id
↪  (0)!
```

| | |
|---|---|
| **Info** | Transfering data |

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Message with no registered callback. Sending negative response.
```

```
SP server: TX <- service: read data response, data_id: 0, status: no callback for service,
↪  data buffered., data: "None"
```

```
Send data: (63): 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 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 88 6a 33 01
```

```
Receive data (63): 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 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 88 6a 33 01
```

```
SP client: RX <- service: read data response, data_id: 0, status: no callback for service,
↪  data buffered., data: "None"
```

```
SP client: RX <- Message has a peculiar status: status: no callback for service, data
↪  buffered.
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Message stored inside callback is correct (Content None and Type is <class 'NoneType'>). |

```
Result (Message stored inside callback): None (<class 'NoneType'>)
```

```
Expectation (Message stored inside callback): result = None (<class 'NoneType'>)
```

| | |
|---|---|
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 1, 'data': None} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 1, 'data':
↪  None} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'data': None, 'data_id': 0, 'service_id':
↪  11, 'status': 1} (<class 'socket_protocol.data_storage'>)
```

### B.1.15 It shall be possible to register a callback for a specific Service-ID and all Data-IDs

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback__' for SID=10 and DID=None
```

| **Info** | Transfering data |
|---|---|

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
Send data: (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
```

```
Receive data (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
|---|---|

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>). |
|---|---|

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 0, 'data':
↪  33} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'data': 33, 'data_id': 0, 'service_id':
↪  11, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

#### B.1.16 It shall be possible to register a callback for a specific Data-IDs and all Service-IDs

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback__' for SID=None and DID=0
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
Send data: (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
```

```
Receive data (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 0, 'data':
↪  33} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'data': 33, 'data_id': 0, 'service_id':
↪  11, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

### B.1.17 It shall be possible to register a callback for all incomming messages

**Testresult**

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

**Info** Setting up communication

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback__' for SID=None and DID=None
```

| **Info** | Transfering data |
|---|---|

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
SP server: RX <- Executing callback __callback__ to process received data
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
```

```
Send data: (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
```

```
Receive data (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |
|---|---|

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>). |
|---|---|

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 0, 'data':
↪  33} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'data': 33, 'data_id': 0, 'service_id':
↪  11, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

#### B.1.18 Callback choice, if several callbacks are available (caused by wildcard callbacks)

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP server: Adding callback '__callback3__' for SID=None and DID=None
SP server: Adding callback '__callback2__' for SID=None and DID=0
SP server: Adding callback '__callback1__' for SID=10 and DID=None
SP server: Adding callback '__callback__' for SID=10 and DID=0
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
SP server: RX <- Executing callback __callback__ to process received data
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "33"
Send data: (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
Receive data (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  33 7d e4 e1 8c bb
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "33"
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 33} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 0, 'data':
↪  33} (<class 'socket_protocol.data_storage'>)
Expectation (Message received by client): result = {'data': 33, 'data_id': 0, 'service_id':
↪  11, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

**Info** Removing Callback for a specific Data- and Service-ID

```
SP server: Deleting existing callback '__callback__' for service_id (10) and data_id (0)!
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback1__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: operation not permitted,
↪  data: "34"
```

```
Send data: (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 33
↪  34 7d 53 62 51 ca
```

```
Receive data (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 33
↪  34 7d 53 62 51 ca
```

```
SP client: RX <- service: read data response, data_id: 0, status: operation not permitted,
↪  data: "34"
```

```
SP client: RX <- Message has a peculiar status: status: operation not permitted
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 6, 'data': 34} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 6, 'data':
↪  34} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'data': 34, 'data_id': 0, 'service_id':
↪  11, 'status': 6} (<class 'socket_protocol.data_storage'>)
```

**Info** Removing Callback for a specific Service-ID and all Data-IDs

```
SP server: Deleting existing callback '__callback1__' for service_id (10) and data_id (None)!
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback2__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: operation not permitted,
↪  data: "35"
```

```
Send data: (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 33
↪  35 7d 4a 79 60 8b
```

```
Receive data (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 36 2c 20 22 64 61 74 61 22 3a 20 33
↪  35 7d 4a 79 60 8b
```

```
SP client: RX <- service: read data response, data_id: 0, status: operation not permitted,
↪  data: "35"
```

```
SP client: RX <- Message has a peculiar status: status: operation not permitted
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

**Success** Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

**Success** Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 6, 'data': 35} and Type is <class 'socket_protocol.data_storage'>).

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 6, 'data':
↪  35} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'data': 35, 'data_id': 0, 'service_id':
↪  11, 'status': 6} (<class 'socket_protocol.data_storage'>)
```

**Info** Removing Callback for a specific Data-ID and all Serice-IDs

```
SP server: Deleting existing callback '__callback2__' for service_id (None) and data_id (0)!
```

**Info** Transfering data

```
SP client: TX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
Send data: (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
Receive data (61): 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 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  31 7d b8 5b f5 78
```

```
SP server: RX <- service: read data request, data_id: 0, status: okay, data: "31"
```

```
SP server: RX <- Executing callback __callback3__ to process received data
```

```
SP server: TX <- service: read data response, data_id: 0, status: okay, data: "36"
```

```
Send data: (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  36 7d 99 96 78 fe
```

```
Receive data (61): 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 31 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 33
↪  36 7d 99 96 78 fe
```

```
SP client: RX <- service: read data response, data_id: 0, status: okay, data: "36"
```

```
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
```

| | |
|---|---|
| **Success** | Message stored inside callback is correct (Content {'data_id': 0, 'service_id': 10, 'status': 0, 'data': 31} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message stored inside callback): {'data_id': 0, 'service_id': 10, 'status': 0,
↪  'data': 31} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message stored inside callback): result = {'data': 31, 'data_id': 0,
↪  'service_id': 10, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

| | |
|---|---|
| **Success** | Message received by client is correct (Content {'data_id': 0, 'service_id': 11, 'status': 0, 'data': 36} and Type is <class 'socket_protocol.data_storage'>). |

```
Result (Message received by client): {'data_id': 0, 'service_id': 11, 'status': 0, 'data':
↪  36} (<class 'socket_protocol.data_storage'>)
```

```
Expectation (Message received by client): result = {'data': 36, 'data_id': 0, 'service_id':
↪  11, 'status': 0} (<class 'socket_protocol.data_storage'>)
```

### B.1.19 Connection established information

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
Result (Client connection status): False (<class 'bool'>)
Expectation (Client connection status): result = False (<class 'bool'>)
```

**Success** Server connection status is correct (Content False and Type is <class 'bool'>).

```
Result (Server connection status): False (<class 'bool'>)
Expectation (Server connection status): result = False (<class 'bool'>)
```

**Info** Connecting Client

```
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 53 5e 67 0b
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 38 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 53 5e 67 0b
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 30 59 be 2f
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 39 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 30 59 be 2f
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Client connection status is correct (Content True and Type is <class 'bool'>).

```
Result (Client connection status): True (<class 'bool'>)
Expectation (Client connection status): result = True (<class 'bool'>)
```

**Success** Server connection status is correct (Content False and Type is <class 'bool'>).

```
Result (Server connection status): False (<class 'bool'>)
Expectation (Server connection status): result = False (<class 'bool'>)
```

**Info** Connecting Server

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

**Success** Client connection status is correct (Content True and Type is <class 'bool'>).

```
Result (Client connection status): True (<class 'bool'>)
Expectation (Client connection status): result = True (<class 'bool'>)
```

**Success** Server connection status is correct (Content True and Type is <class 'bool'>).

```
Result (Server connection status): True (<class 'bool'>)
Expectation (Server connection status): result = True (<class 'bool'>)
```

**Info** Adding secrets to socket_protocol

**Success** Client connection status is correct (Content False and Type is <class 'bool'>).

```
Result (Client connection status): False (<class 'bool'>)
Expectation (Client connection status): result = False (<class 'bool'>)
```

**Success** Server connection status is correct (Content False and Type is <class 'bool'>).

```
Result (Server connection status): False (<class 'bool'>)
Expectation (Server connection status): result = False (<class 'bool'>)
```

**Info** Doing authentification

```
SP client: TX <- service: authentification request, data_id: seed, status: okay, 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 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 fd 82 a2 a9
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 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 fd 82 a2 a9
SP server: RX <- service: authentification request, data_id: seed, status: okay, data: "None"
SP server: RX <- Executing callback __authentificate_create_seed__ to process received data
SP server: TX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'23168fe5dbc8cf6ff8f24e7fb999a3fbe667da5a6a158eb3ada8c83f11d967ad'"
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 31 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
↪  33 31 36 38 66 65 35 64 62 63 38 63 66 36 66 66 38 66 32 34 65 37 66 62 39 39 39 61 33 66
↪  62 65 36 36 37 64 61 35 61 36 61 31 35 38 65 62 33 61 64 61 38 63 38 33 66 31 31 64 39 36
↪  37 61 64 22 7d af 35 34 74
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 31 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 33 31 36 38 66 65 35 64 62 63 38 63 66 36 66 66 38 66 32 34 65 37 66 62 39 39 39 61 33
↪  66 62 65 36 36 37 64 61 35 61 36 61 31 35 38 65 62 33 61 64 61 38 63 38 33 66 31 31 64 39
↪  36 37 61 64 22 7d af 35 34 74
SP client: RX <- service: authentification response, data_id: seed, status: okay, data:
↪  "'23168fe5dbc8cf6ff8f24e7fb999a3fbe667da5a6a158eb3ada8c83f11d967ad'"
SP client: Executing callback __authentificate_create_key__ to process received data
SP client: TX <- service: authentification request, data_id: key, status: okay, data:
↪  "'52cffa52636d4dc117438088e2df14d7d65557a263b0b636f84eec85d1ee2b0e23459582b543eca30bf63f9
↪  39d1fc10777e9ef2cf943be99d7a33f292e9bbf71'"
Send data: (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f 69
↪  64 22 3a 20 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22 35
↪  32 63 66 66 61 35 32 36 33 36 64 34 64 63 31 31 37 34 33 38 30 38 38 65 32 64 66 31 34 64
↪  37 64 36 35 35 35 37 61 32 36 33 62 30 62 36 33 36 66 38 34 65 65 63 38 35 64 31 65 65 32
↪  62 30 65 32 33 34 35 39 35 38 32 62 35 34 33 65 63 61 33 30 62 66 36 33 66 39 33 39 64 31
↪  66 63 31 30 37 37 37 65 39 65 66 32 63 66 39 34 33 62 65 39 39 64 37 61 33 33 66 32 39 32
↪  65 39 62 62 66 37 31 22 7d 93 4c bc 2d
Receive data (188): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 2c 20 22 73 65 72 76 69 63 65 5f
↪  69 64 22 3a 20 30 2c 20 22 73 74 61 74 75 73 22 3a 20 30 2c 20 22 64 61 74 61 22 3a 20 22
↪  35 32 63 66 66 61 35 32 36 33 36 64 34 64 63 31 31 37 34 33 38 30 38 38 65 32 64 66 31 34
↪  64 37 64 36 35 35 35 37 61 32 36 33 62 30 62 36 33 36 66 38 34 65 65 63 38 35 64 31 65 65
↪  32 62 30 65 32 33 34 35 39 35 38 32 62 35 34 33 65 63 61 33 30 62 66 36 33 66 39 33 39 64
↪  31 66 63 31 30 37 37 37 65 39 65 66 32 63 66 39 34 33 62 65 39 39 64 37 61 33 33 66 32 39
↪  32 65 39 62 62 66 37 31 22 7d 93 4c bc 2d
SP server: RX <- service: authentification request, data_id: key, status: okay, data:
↪  "'52cffa52636d4dc117438088e2df14d7d65557a263b0b636f84eec85d1ee2b0e23459582b543eca30bf63f9
↪  39d1fc10777e9ef2cf943be99d7a33f292e9bbf71'"
SP server: RX <- Executing callback __authentificate_check_key__ to process received data
SP server: TX <- service: authentification response, data_id: key, status: okay, data: "True"
Send data: (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 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 74 72
↪  75 65 7d 94 fe 74 32
Receive data (62): 7b 22 64 61 74 61 5f 69 64 22 3a 20 31 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 74 72
```

```
Result (Client connection status): True (<class 'bool'>)
Expectation (Client connection status): result = True (<class 'bool'>)
```

| | |
|---|---|
| **Success** | Server connection status is correct (Content True and Type is <class 'bool'>). |

```
Result (Server connection status): True (<class 'bool'>)
Expectation (Server connection status): result = True (<class 'bool'>)
```

### B.1.20 Is connected information

**Testresult**

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

| | |
|---|---|
| **Info** | Setting up communication |

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
Result (Client Communication instance connection status): False (<class 'bool'>)
Expectation (Client Communication instance connection status): result = False (<class 'bool'>)
```

**Success** Server Communication instance connection status is correct (Content False and Type is <class 'bool'>).

```
Result (Server Communication instance connection status): False (<class 'bool'>)
Expectation (Server Communication instance connection status): result = False (<class 'bool'>)
```

**Info** Connecting Client

```
SP client: Cleaning up receive-buffer
SP client: TX <- service: channel name request, data_id: name, status: okay, 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 38 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 53 5e 67 0b
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 38 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 53 5e 67 0b
SP server: RX <- service: channel name request, data_id: name, status: okay, data: "None"
SP server: RX <- Executing callback __channel_name_request__ to process received data
SP server: TX <- service: channel name response, data_id: name, status: okay, 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 39 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 30 59 be 2f
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 39 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 30 59 be 2f
SP client: RX <- service: channel name response, data_id: name, status: okay, data: "None"
SP client: Executing callback __channel_name_response__ to process received data
```

**Success** Client Communication instance connection status is correct (Content True and Type is <class 'bool'>).

```
Result (Client Communication instance connection status): True (<class 'bool'>)
Expectation (Client Communication instance connection status): result = True (<class 'bool'>)
```

**Success** Server Communication instance connection status is correct (Content False and Type is <class 'bool'>).

```
Result (Server Communication instance connection status): False (<class 'bool'>)
Expectation (Server Communication instance connection status): result = False (<class 'bool'>)
```

**Info** Connecting Server

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

**Success** Client Communication instance connection status is correct (Content True and Type is <class 'bool'>).

```
Result (Client Communication instance connection status): True (<class 'bool'>)
Expectation (Client Communication instance connection status): result = True (<class 'bool'>)
```

| **Success** | Server Communication instance connection status is correct (Content True and Type is <class 'bool'>). |
|---|---|

```
Result (Server Communication instance connection status): True (<class 'bool'>)
Expectation (Server Communication instance connection status): result = True (<class 'bool'>)
```

### B.1.21 Reconnect Method

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
Result (Reconnect executed marker): False (<class 'bool'>)
Expectation (Reconnect executed marker): result = False (<class 'bool'>)
```

| | |
|---|---|
| **Success** | Reconnect executed marker is correct (Content False and Type is <class 'bool'>). |

```
Result (Reconnect executed marker): False (<class 'bool'>)
Expectation (Reconnect executed marker): result = False (<class 'bool'>)
```

| | |
|---|---|
| **Info** | Executing reconnect for Server |

| | |
|---|---|
| **Success** | Reconnect executed marker is correct (Content True and Type is <class 'bool'>). |

```
Result (Reconnect executed marker): True (<class 'bool'>)
Expectation (Reconnect executed marker): result = True (<class 'bool'>)
```

| | |
|---|---|
| **Success** | Reconnect executed marker is correct (Content False and Type is <class 'bool'>). |

```
Result (Reconnect executed marker): False (<class 'bool'>)
Expectation (Reconnect executed marker): result = False (<class 'bool'>)
```

| | |
|---|---|
| **Info** | Executing reconnect for Client |

| | |
|---|---|
| **Success** | Reconnect executed marker is correct (Content True and Type is <class 'bool'>). |

```
Result (Reconnect executed marker): True (<class 'bool'>)
Expectation (Reconnect executed marker): result = True (<class 'bool'>)
```

| | |
|---|---|
| **Success** | Reconnect executed marker is correct (Content True and Type is <class 'bool'>). |

```
Result (Reconnect executed marker): True (<class 'bool'>)
Expectation (Reconnect executed marker): result = True (<class 'bool'>)
```

#### B.1.22 A full Message Object including the defined properties and data shall be transfered.

**Description**

Every Communication shall transfer a complete message with its content.

**Reason for the implementation**

See Reasons for every single information of the Message Object.

**Fitcriterion**

Send two different messages and compare the received message with each sent message.

**Testresult**

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

| **Info** | Setting up communication |
|---|---|

```
SP server: Cleaning up receive-buffer
SP server: Adding Service with Request=authentification request and Response=authentification
↪  response
SP server: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP server: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP server: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP server: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP server: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP server: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP server: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP server: Adding Service with Request=channel name request and Response=channel name response
SP server: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP server: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP server: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP server: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP server: Adding Service with Request=read data request and Response=read data response
SP server: Adding Service with Request=write data request and Response=write data response
SP server: Adding Service with Request=execute request and Response=execute response
SP server: Initialisation finished.
SP client: Cleaning up receive-buffer
SP client: Adding Service with Request=authentification request and Response=authentification
↪  response
SP client: Adding Message (service: authentification request, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: seed) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification request, data_id: key) to the
↪  authentification whitelist
SP client: Adding Message (service: authentification response, data_id: key) to the
↪  authentification whitelist
SP client: Adding callback '__authentificate_create_seed__' for SID=0 and DID=0
SP client: Adding callback '__authentificate_create_key__' for SID=1 and DID=0
SP client: Adding callback '__authentificate_check_key__' for SID=0 and DID=1
SP client: Adding callback '__authentificate_process_feedback__' for SID=1 and DID=1
SP client: Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION
SP client: Adding Service with Request=channel name request and Response=channel name response
SP client: Adding Message (service: channel name request, data_id: name) to the
↪  authentification whitelist
SP client: Adding Message (service: channel name response, data_id: name) to the
↪  authentification whitelist
SP client: Adding callback '__channel_name_request__' for SID=8 and DID=0
SP client: Adding callback '__channel_name_response__' for SID=9 and DID=0
SP client: Adding Service with Request=read data request and Response=read data response
SP client: Adding Service with Request=write data request and Response=write data response
```

```
SP client: TX <- service: 17, data_id: 34, status: okay, data: "'msg1_data_to_be_transfered'"
Send data: (41): 00 00 00 00 00 00 00 11 00 00 00 22 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f
↪  5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7d
Receive data (41): 00 00 00 00 00 00 00 11 00 00 00 22 22 6d 73 67 31 5f 64 61 74 61 5f 74 6f
↪  5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7e
SP server: RX <- Received message has a wrong checksum. Message will be ignored.
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 34) not in buffer.
```

**Success** Returnvalue of Client send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Client send Method): True (<class 'bool'>)
Expectation (Returnvalue of Client send Method): result = True (<class 'bool'>)
```

**Success** Checksum Error → No message received by server is correct (Content None and Type is <class 'NoneType'>).

```
Result (Checksum Error -> No message received by server): None (<class 'NoneType'>)
Expectation (Checksum Error -> No message received by server): result = None (<class
↪  'NoneType'>)
```

**Info** Transfering a message server → client

```
SP server: TX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
Send data: (41): 00 00 00 04 00 00 00 11 00 00 00 23 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f
↪  5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7b
Receive data (41): 00 00 00 04 00 00 00 11 00 00 00 23 22 6d 73 67 32 5f 64 61 74 61 5f 74 6f
↪  5f 62 65 5f 74 72 61 6e 73 66 65 72 65 64 22 7b
SP client: RX <- service: 17, data_id: 35, status: service or data unknown, data:
↪  "'msg2_data_to_be_transfered'"
SP client: RX <- Message has a peculiar status: status: service or data unknown
SP client: Message data is stored in buffer and is now ready to be retrieved by receive method
SP server: TIMEOUT (0.25s): Requested data (service_id: 17; data_id: 35) not in buffer.
```

**Success** Returnvalue of Server send Method is correct (Content True and Type is <class 'bool'>).

```
Result (Returnvalue of Server send Method): True (<class 'bool'>)
Expectation (Returnvalue of Server send Method): result = True (<class 'bool'>)
```

**Success** Checksum Error → No message received by client is correct (Content None and Type is <class 'NoneType'>).

```
Result (Checksum Error -> No message received by client): None (<class 'NoneType'>)
Expectation (Checksum Error -> No message received by client): result = None (<class
↪  'NoneType'>)
```

# C Test-Coverage

## C.1 socket_protocol

The line coverage for socket_protocol was 100.0%
The branch coverage for socket_protocol was 100.0%

### C.1.1 socket_protocol.__init__.py

The line coverage for socket_protocol.__init__.py was 100.0%
The branch coverage for socket_protocol.__init__.py was 100.0%

```
#!/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.data_storage`
* :class:`socket_protocol.pure_json_protocol`
* :class:`socket_protocol.struct_json_protocol`

**Unittest:**

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

**Module 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__)
```

```
48
49
50 __DESCRIPTION__ = """The Module {\\tt %s} is designed for point to point communication for client
       -server issues.
51 For more Information read the sphinx documentation.""" % __name__.replace('_', '\_')
52 """The Module Description"""
53 __INTERPRETER__ = (2, 3)
54 """The Tested Interpreter-Versions"""
55
56 SID_AUTH_REQUEST = 0
57 """SID for authentification request"""
58 SID_AUTH_RESPONSE = 1
59 """SID for authentification response"""
60 DID_AUTH_SEED = 0
61 """DID for authentification (seed)"""
62 DID_AUTH_KEY = 1
63 """DID for authentification (key)"""
64 SID_CHANNEL_NAME_REQUEST = 8
65 """SID for channel name exchange request """
66 SID_CHANNEL_NAME_RESPONSE = 9
67 """SID for channel name exchange response"""
68 DID_CHANNEL_NAME = 0
69 """DID for channel name """
70 SID_READ_REQUEST = 10
71 """SID for a read data request"""
72 SID_READ_RESPONSE = 11
73 """SID for read data response"""
74 SID_WRITE_REQUEST = 20
75 """SID for a write data request"""
76 SID_WRITE_RESPONSE = 21
77 """SID for a write data response"""
78 SID_EXECUTE_REQUEST = 30
79 """SID for a execute request"""
80 SID_EXECUTE_RESPONSE = 31
81 """SID for a execute response"""
82
83 STATUS_OKAY = 0
84 """Status for 'okay'"""
85 STATUS_BUFFERING_UNHANDLED_REQUEST = 1
86 """Status for 'unhandled request'"""
87 STATUS_CALLBACK_ERROR = 2
88 """Status for 'callback errors'"""
89 STATUS_AUTH_REQUIRED = 3
90 """Status for 'authentification is required'"""
91 STATUS_SERVICE_OR_DATA_UNKNOWN = 4
92 """Status for 'service or data unknown'"""
93 STATUS_CHECKSUM_ERROR = 5
94 """Status for 'checksum error'"""
95 STATUS_OPERATION_NOT_PERMITTED = 6
96 """Status for 'operation not permitted'"""
97
98 AUTH_STATE_UNTRUSTED_CONNECTION = 0
99 """Authentification Status for an 'Untrusted Connection'"""
100 AUTH_STATE_SEED_REQUESTED = 1
101 """Authentification Status for 'Seed was requested'"""
102 AUTH_STATE_SEED_TRANSFERRED = 2
103 """Authentification Status for 'Seed has been sent'"""
104 AUTH_STATE_KEY_TRANSFERRED = 3
105 """Authentification Status for 'Key has been sent'"""
106 AUTH_STATE_TRUSTED_CONNECTION = 4
107 """Authentification Status for a 'Trusted Connection'"""
108 AUTH_STATE__NAMES = {AUTH_STATE_UNTRUSTED_CONNECTION: 'Untrusted Connection',
```

```
109                         AUTH_STATE_SEED_REQUESTED: 'Seed was requested',
110                         AUTH_STATE_SEED_TRANSFERRED: 'Seed has been sent',
111                         AUTH_STATE_KEY_TRANSFERRED: 'Key has been sent',
112                         AUTH_STATE_TRUSTED_CONNECTION: 'Trusted Connection'}
113 """Authentification Status names for previous defined authentification states"""
114
115
116 class RequestSidExistsError(Exception):
117     pass
118
119
120 class ResponseSidExistsError(Exception):
121     pass
122
123
124 class _callback_storage(dict):
125     DEFAULT_CHANNEL_NAME = 'all_others'
126
127     def __init__(self, channel_name, log_prefix):
128         self.init_channel_name(channel_name)
129         self.__log_prefix__ = log_prefix
130         dict.__init__(self)
131
132     def init_channel_name(self, channel_name):
133         if channel_name is None:
134             self.logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__ + '.' + self.
    DEFAULT_CHANNEL_NAME)
135         else:
136             self.logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__ + '.' +
    channel_name)
137
138     def get(self, service_id, data_id):
139         if dict.get(self, service_id, {}).get(data_id, None) is not None:
140             return self[service_id][data_id]
141         elif dict.get(self, service_id, {}).get(None, None) is not None:
142             return self[service_id][None]
143         elif dict.get(self, None, {}).get(data_id, None) is not None:
144             return self[None][data_id]
145         elif dict.get(self, None, {}).get(None, None) is not None:
146             return self[None][None]
147         else:
148             return (None, None, None)
149
150     def add(self, service_id, data_id, callback, *args, **kwargs):
151         cb_data = self.get(service_id, data_id)
152         if dict.get(self, service_id, {}).get(data_id, None) is not None:
153             if callback is None:
154                 self.logger.warning("%s Deleting existing callback %s for service_id (%s) and
    data_id (%s)!", self.__log_prefix__(), repr(cb_data[0].__name__), repr(service_id), repr(
    data_id))
155                 del(self[service_id][data_id])
156                 return
157             else:
158                 self.logger.warning("%s Overwriting existing callback %s for service_id (%s) and
    data_id (%s) to %s!", self.__log_prefix__(), repr(cb_data[0].__name__), repr(service_id),
    repr(data_id), repr(callback.__name__))
159         else:
160             self.logger.debug("%s Adding callback %s for SID=%s and DID=%s", self.__log_prefix__
    (), repr(callback.__name__), repr(service_id), repr(data_id))
161         if service_id not in self:
162             self[service_id] = {}
163         self[service_id][data_id] = (callback, args, kwargs)
```

```
164
165
166 class data_storage(dict):
167     """
168     This is a storage object for socket_protocol messages.
169
170     :param status: The message status.
171     :type status: int
172     :param service_id: The Service-ID.
173     :type service_id: int
174     :param data_id: The Data-ID.
175     :type data_id: int
176     :param data: The transfered data.
177     :type data: any
178     """
179
180     KEY_STATUS = 'status'
181     KEY_SERVICE_ID = 'service_id'
182     KEY_DATA_ID = 'data_id'
183     KEY_DATA = 'data'
184     ALL_KEYS = [KEY_DATA, KEY_DATA_ID, KEY_SERVICE_ID, KEY_STATUS]
185
186     def __init__(self, *args, **kwargs):
187         dict.__init__(self, *args, **kwargs)
188         for key in self.ALL_KEYS:
189             if key not in self:
190                 self[key] = None
191
192     def get_status(self, default=None):
193         """
194         This Method returns the message status.
195
196         :param default: The default value, if no data is available.
197         """
198         return self.get(self.KEY_STATUS, default)
199
200     def get_service_id(self, default=None):
201         """
202         This Method returns the message Service-ID.
203
204         :param default: The default value, if no data is available.
205         """
206         return self.get(self.KEY_SERVICE_ID, default)
207
208     def get_data_id(self, default=None):
209         """
210         This Method returns the message Data-ID.
211
212         :param default: The default value, if no data is available.
213         """
214         return self.get(self.KEY_DATA_ID, default)
215
216     def get_data(self, default=None):
217         """
218         This Method returns the message data.
219
220         :param default: The default value, if no data is available.
221         """
222         return self.get(self.KEY_DATA, default)
223
224
225 class pure_json_protocol(object):
```

```
226     """
227     This `class` supports to transfer a message and it's data.
228
229     :param comm_instance: A communication instance.
230     :type comm_instance: instance
231     :param secret: An optinal secret (e.g. created by ``binascii.hexlify(os.urandom(24))``).
232     :type secret: str
233     :param auto_auth: An optional parameter to enable (True) automatic authentification,
        otherwise you need to do it manually, if needed.
234     :type auto_auth: bool
235     :param channel_name: An optional parameter to set a channel name for logging of the
        communication.
236     :type channel_name: str
237
238     .. hint::
239
240         * The Service-ID is designed to identify the type of the communication (e.g. :const:`
        READ_REQUEST`, :const:`WRITE_REQUEST`, :const:`READ_RESPONSE`, :const:`WRITE_RESPONSE`, ...)
241         * The Data-ID is designed to identify the requests / responses using the same Service_ID.
242
243     .. note:: The :class:`comm_instance` needs to have at least the following interface:
244
245         * A Method :func:`comm_instance.init_channel_name` to set the channel name.
246         * A Constant :const:`comm_instance.IS_CLIENT` to identify that the :class:`comm_instance`
         is a client (True) or a server (False).
247         * A Method :func:`comm_instance.is_connected` to identify if the instance is connected (
        True) or not (False).
248         * A Method :func:`comm_instance.reconnect` to initiate a reconnect.
249         * A Method :func:`comm_instance.register_callback` to register a data available callback.
250         * A Method :func:`comm_instance.register_connect_callback` to register a connect callback
        .
251         * A Method :func:`comm_instance.register_disconnect_callback` to register a disconnect
        callback.
252         * A Method :func:`comm_instance.send` to send data via the :class:`comm_instance`.
253
254     .. note::  The parameter :const:`auto_auth` is only relevant, if a secret is given and the :
        class:`comm_instance` is a client. The authentification is initiated directly after the
        connection is established.
255
256     .. note::  The :const:`channel_name`-exchange will be initiated by the client directly after
        the the connection is established.
257
258         * If a channel_name is given at both communication sides and they are different, the
        client name is taken over and the server will log a warning message.
259     """
260     DEFAULT_CHANNEL_NAME = 'all_others'
261
262     def __init__(self, comm_instance, secret=None, auto_auth=False, channel_name=None):
263         self.__comm_inst__ = comm_instance
264         self.__secret__ = secret
265         self.__auto_auth__ = auto_auth
266         #
267         self.__auth_whitelist__ = {}
268         self.__sid_response_dict__ = {}
269         self.__sid_name_dict__ = {}
270         self.__did_name_dict__ = {}
271         #
272         self.__status_name_dict = {}
273         self.add_status(STATUS_OKAY, 'okay')
274         self.add_status(STATUS_BUFFERING_UNHANDLED_REQUEST, 'no callback for service, data
        buffered.')
```

```
275         self.add_status(STATUS_CALLBACK_ERROR, 'callback error.')
276         self.add_status(STATUS_AUTH_REQUIRED, 'authentification required')
277         self.add_status(STATUS_SERVICE_OR_DATA_UNKNOWN, 'service or data unknown')
278         self.add_status(STATUS_CHECKSUM_ERROR, 'checksum error')
279         self.add_status(STATUS_OPERATION_NOT_PERMITTED, 'operation not permitted')
280         #
281         self.__callbacks__ = _callback_storage(channel_name, self.__log_prefix__)
282         self.__init_channel_name__(channel_name)
283         #
284         self.__clean_receive_buffer__()
285 
286         self.add_service(SID_AUTH_REQUEST, SID_AUTH_RESPONSE, 'authentification request', '
    authentification response')
287         self.add_data((SID_AUTH_REQUEST, SID_AUTH_RESPONSE), DID_AUTH_SEED, 'seed')
288         self.add_data(SID_AUTH_REQUEST, DID_AUTH_KEY, 'key')
289         self.add_data(SID_AUTH_RESPONSE, DID_AUTH_KEY, 'key')
290         self.add_msg_to_auth_whitelist_(SID_AUTH_REQUEST, DID_AUTH_SEED)
291         self.add_msg_to_auth_whitelist_(SID_AUTH_RESPONSE, DID_AUTH_SEED)
292         self.add_msg_to_auth_whitelist_(SID_AUTH_REQUEST, DID_AUTH_KEY)
293         self.add_msg_to_auth_whitelist_(SID_AUTH_RESPONSE, DID_AUTH_KEY)
294         self.__callbacks__.add(SID_AUTH_REQUEST, DID_AUTH_SEED, self.
    __authentificate_create_seed__)
295         self.__callbacks__.add(SID_AUTH_RESPONSE, DID_AUTH_SEED, self.
    __authentificate_create_key__)
296         self.__callbacks__.add(SID_AUTH_REQUEST, DID_AUTH_KEY, self.__authentificate_check_key__)
297         self.__callbacks__.add(SID_AUTH_RESPONSE, DID_AUTH_KEY, self.
    __authentificate_process_feedback__)
298         self.__authentification_state_reset__()
299 
300         self.add_service(SID_CHANNEL_NAME_REQUEST, SID_CHANNEL_NAME_RESPONSE, 'channel name
    request', 'channel name response')
301         self.add_data((SID_CHANNEL_NAME_REQUEST, SID_CHANNEL_NAME_RESPONSE), DID_CHANNEL_NAME, '
    name')
302         self.add_msg_to_auth_whitelist_(SID_CHANNEL_NAME_REQUEST, DID_CHANNEL_NAME)
303         self.add_msg_to_auth_whitelist_(SID_CHANNEL_NAME_RESPONSE, DID_CHANNEL_NAME)
304         self.__callbacks__.add(SID_CHANNEL_NAME_REQUEST, DID_CHANNEL_NAME, self.
    __channel_name_request__)
305         self.__callbacks__.add(SID_CHANNEL_NAME_RESPONSE, DID_CHANNEL_NAME, self.
    __channel_name_response__)
306 
307         self.add_service(SID_READ_REQUEST, SID_READ_RESPONSE, 'read data request', 'read data
    response')
308         self.add_service(SID_WRITE_REQUEST, SID_WRITE_RESPONSE, 'write data request', 'write data
     response')
309         self.add_service(SID_EXECUTE_REQUEST, SID_EXECUTE_RESPONSE, 'execute request', 'execute
    response')
310 
311         self.__seed__ = None
312         self.__comm_inst__.register_callback(self.__data_available_callback__)
313         self.__comm_inst__.register_connect_callback(self.__connection_established__)
314         self.__comm_inst__.register_disconnect_callback(self.__authentification_state_reset__)
315         logger.info('%s Initialisation finished.', self.__log_prefix__())
316 
317     def __analyse_frame__(self, frame):
318         if sys.version_info >= (3, 0):
319             return data_storage(json.loads(frame[:-4].decode('utf-8')))
320         else:
321             return data_storage(json.loads(frame[:-4]))
322 
323     def __authentificate_check_key__(self, msg):
324         key = msg.get_data()
325         if key == self.__authentificate_salt_and_hash__(self.__seed__):
326             self.__authentification_state__ = AUTH_STATE_TRUSTED_CONNECTION
327             return STATUS_OKAY, True
```

```
328         else:
329             self.__authentification_state__ = AUTH_STATE_UNTRUSTED_CONNECTION
330             return STATUS_OKAY, False
331 
332     def __authenticate_create_key__(self, msg):
333         self.__authentification_state__ = AUTH_STATE_KEY_TRANSFERRED
334         seed = msg.get_data()
335         key = self.__authenticate_salt_and_hash__(seed)
336         self.send(SID_AUTH_REQUEST, DID_AUTH_KEY, key)
337 
338     def __authenticate_create_seed__(self, msg):
339         self.__authentification_state__ = AUTH_STATE_SEED_TRANSFERRED
340         if sys.version_info >= (3, 0):
341             self.__seed__ = binascii.hexlify(os.urandom(32)).decode('utf-8')
342         else:
343             self.__seed__ = binascii.hexlify(os.urandom(32))
344         return STATUS_OKAY, self.__seed__
345 
346     def __authenticate_process_feedback__(self, msg):
347         feedback = msg.get_data()
348         if feedback:
349             self.__authentification_state__ = AUTH_STATE_TRUSTED_CONNECTION
350             self.logger.info("%s Got positive authentification feedback", self.__log_prefix__())
351         else:
352             self.__authentification_state__ = AUTH_STATE_UNTRUSTED_CONNECTION
353             self.logger.warning("%s Got negative authentification feedback", self.__log_prefix__
    ())
354         return STATUS_OKAY, None
355 
356     def __authenticate_salt_and_hash__(self, seed):
357         if sys.version_info >= (3, 0):
358             return hashlib.sha512(bytes(seed, 'utf-8') + self.__secret__).hexdigest()
359         else:
360             return hashlib.sha512(seed.encode('utf-8') + self.__secret__.encode('utf-8')).
    hexdigest()
361 
362     def __authentification_state_reset__(self):
363         self.logger.info("%s Resetting authentification state to AUTH_STATE_UNTRUSTED_CONNECTION"
    , self.__log_prefix__())
364         self.__authentification_state__ = AUTH_STATE_UNTRUSTED_CONNECTION
365 
366     def __authentification_required__(self, service_id, data_id):
367         return data_id not in self.__auth_whitelist__.get(service_id, [])
368 
369     def __buffer_received_data__(self, msg):
370         if not msg.get_service_id() in self.__msg_buffer__:
371             self.__msg_buffer__[msg.get_service_id()] = {}
372         if not msg.get_data_id() in self.__msg_buffer__[msg.get_service_id()]:
373             self.__msg_buffer__[msg.get_service_id()][msg.get_data_id()] = []
374         self.__msg_buffer__[msg.get_service_id()][msg.get_data_id()].append(msg)
375         self.logger.debug("%s Message data is stored in buffer and is now ready to be retrieved
    by receive method", self.__log_prefix__())
376 
377     def __build_frame__(self, service_id, data_id, data, status=STATUS_OKAY):
378         data_frame = json.dumps(self.__mk_msg__(status, service_id, data_id, data))
379         if sys.version_info >= (3, 0):
380             data_frame = bytes(data_frame, 'utf-8')
381         checksum = self.__calc_chksum__(data_frame)
382         return data_frame + checksum
383 
384     def __calc_chksum__(self, raw_data):
385         return struct.pack('>I', binascii.crc32(raw_data) & 0xffffffff)
386 
```

```
387     @property
388     def __channel_name__(self):
389         cn = self.logger.name.split('.')[-1]
390         if cn != self.DEFAULT_CHANNEL_NAME:
391             return cn
392 
393     def __channel_name_response__(self, msg):
394         data = msg.get_data()
395         if self.__channel_name__ is None and data is not None:
396             self.__init_channel_name__(data)
397             self.logger.info('%s channel name is now %s', self.__log_prefix__(), repr(self.
    __channel_name__))
398         return STATUS_OKAY, None
399 
400     def __channel_name_request__(self, msg):
401         data = msg.get_data()
402         if data is None:
403             return STATUS_OKAY, self.__channel_name__
404         else:
405             prev_channel_name = self.__channel_name__
406             self.__init_channel_name__(data)
407             if prev_channel_name is not None and prev_channel_name != data:
408                 self.logger.warning('%s overwriting user defined channel name from %s to %s',
    self.__log_prefix__(), repr(prev_channel_name), repr(data))
409             elif prev_channel_name is None:
410                 self.logger.info('%s channel name is now %s', self.__log_prefix__(), repr(self.
    __channel_name__))
411             return STATUS_OKAY, None
412 
413     def __check_frame_checksum__(self, frame):
414         return self.__calc_chksum__(frame[:-4]) == frame[-4:]
415 
416     def __clean_receive_buffer__(self):
417         self.logger.debug("%s Cleaning up receive-buffer", self.__log_prefix__())
418         self.__msg_buffer__ = {}
419 
420     def __connection_established__(self):
421         self.__clean_receive_buffer__()
422         if self.__comm_inst__.IS_CLIENT:
423             self.send(SID_CHANNEL_NAME_REQUEST, 0, self.__channel_name__)
424         if self.__auto_auth__ and self.__comm_inst__.IS_CLIENT and self.__secret__ is not None:
425             self.authentificate()
426 
427     def __data_available_callback__(self, comm_inst):
428         frame = comm_inst.receive()
429         msg = self.__analyse_frame__(frame)
430         if not self.__check_frame_checksum__(frame):
431             # Wrong Checksum
432             self.logger.warning("%s RX <- Received message has a wrong checksum. Message will be
    ignored.", self.__log_prefix__())
433             return  # No response needed
434         elif not self.check_authentification_state() and self.__authentification_required__(msg.
    get_service_id(), msg.get_data_id()):
435             # Authentification required
436             if msg.get_service_id() in self.__sid_response_dict__.keys():
437                 self.logger.warning("%s RX <- Authentification is required. Just sending negative
     response.", self.__log_prefix__())
438                 status = STATUS_AUTH_REQUIRED
439                 data = None
440             else:
441                 self.logger.warning("%s RX <- Authentification is required. Message will be
    ignored.", self.__log_prefix__())
442                 return  # No response needed
```

```
443         else:
444             # Valid message
445             self.logger.info(
446                 '%s RX <- %s, %s, data: "%s"',
447                 self.__log_prefix__(),
448                 self.__get_message_name__(msg.get_service_id(), msg.get_data_id()),
449                 self.__get_status_name__(msg.get_status()),
450                 repr(msg.get_data())
451             )
452             if msg.get_status() not in [STATUS_OKAY]:
453                 self.logger.warning("%s RX <- Message has a peculiar status: %s", self.
    __log_prefix__(), self.__get_status_name__(msg.get_status()))
454             callback, args, kwargs = self.__callbacks__.get(msg.get_service_id(), msg.get_data_id
    ())
455             if msg.get_service_id() in self.__sid_response_dict__.keys():
456                 #
457                 # REQUEST RECEIVED
458                 #
459                 if callback is None:
460                     self.logger.warning("%s RX <- Message with no registered callback. Sending
    negative response.", self.__log_prefix__())
461                     status = STATUS_BUFFERING_UNHANDLED_REQUEST
462                     data = None
463                 else:
464                     try:
465                         self.logger.debug("%s RX <- Executing callback %s to process received
    data", self.__log_prefix__(), callback.__name__)
466                         status, data = callback(msg, *args, **kwargs)
467                     except Exception:
468                         logger.error('{lp} RX <- Exception raised. Check callback {callback_name}
     and it\'s return values for service_id  {service_id} and data_id {data_id}'.format(lp=self.
    __log_prefix__(), callback_name=callback.__name__, service_id=repr(msg.get_service_id()),
    data_id=repr(msg.get_data_id())))
469                         status = STATUS_CALLBACK_ERROR
470                         data = None
471             else:
472                 #
473                 # RESPONSE RECEIVED
474                 #
475                 if callback is None:
476                     self.__buffer_received_data__(msg)
477                 else:
478                     self.logger.debug("%s Executing callback %s to process received data", self.
    __log_prefix__(), callback.__name__)
479                     callback(msg, *args, **kwargs)
480                 return  # No response needed
481         self.send(self.__sid_response_dict__[msg.get_service_id()], msg.get_data_id(), data,
    status=status)
482
483     def __get_message_name__(self, service_id, data_id):
484         return 'service: %s, data_id: %s' % (
485             self.__sid_name_dict__.get(service_id, repr(service_id)),
486             self.__did_name_dict__.get(service_id, {}).get(data_id, repr(data_id)),
487         )
488
489     def __get_status_name__(self, status):
490         return 'status: %s' % (self.__status_name_dict.get(status, 'unknown status: %s' % repr(
    status)))
491
```

```
492     def __init_channel_name__(self, channel_name):
493         self.__comm_inst__.init_channel_name(channel_name)
494         self.__callbacks__.init_channel_name(channel_name)
495         if channel_name is None:
496             self.logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__ + '.' + self.
    DEFAULT_CHANNEL_NAME)
497         else:
498             self.logger = logging.getLogger(ROOT_LOGGER_NAME).getChild(__name__ + '.' +
    channel_name)
499 
500     def __log_prefix__(self):
501         return 'SP client:' if self.__comm_inst__.IS_CLIENT else 'SP server:'
502 
503     def __mk_msg__(self, status, service_id, data_id, data):
504         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})
505 
506     def add_data(self, service_id, data_id, name):
507         """
508         Method to add a name for a specific message.
509 
510         :param service_id: The Service-ID of the message. See class definitions starting with ``
    SID_``.
511         :type service_id: int or list of ints
512         :param data_id: The Data-ID of the message.
513         :type data_id: int
514         :param name: The Name for the transfered message.
515         :type name: str
516         """
517         try:
518             iter(service_id)
519         except Exception:
520             service_id = (service_id, )
521 
522         for sid in service_id:
523             if sid not in self.__did_name_dict__:
524                 self.__did_name_dict__[sid] = {}
525             self.__did_name_dict__[sid][data_id] = name
526 
527     def add_msg_to_auth_whitelist_(self, service_id, data_id):
528         """
529         Method to add a specific message to the list, where no authentification is required.
530 
531         :param service_id: The Service-ID of the message. See class definitions starting with ``
    SID_``.
532         :type service_id: int
533         :param data_id: The Data-ID of the message.
534         :type data_id: int
535         """
536         if service_id not in self.__auth_whitelist__:
537             self.__auth_whitelist__[service_id] = []
538         self.__auth_whitelist__[service_id].append(data_id)
539         logger.debug('%s Adding Message (%s) to the authentification whitelist', self.
    __log_prefix__(), self.__get_message_name__(service_id, data_id))
540 
541     def add_service(self, req_sid, resp_sid, req_name=None, resp_name=None):
542         """
543         Method to add a Service defined by Request- and Response Serivce-ID.
544 
545         :param req_sid: The Request Service-ID.
546         :type req_sid: int
547         :param resp_sid: The Response Service-ID.
548         :type resp_sid: int
549         """
```

```
550         if req_sid in self.__sid_response_dict__:
551             logger.error('%s Service with Request-SID=%d and Response-SID=%d not added, because request SID is already registered', self.__log_prefix__(), req_sid, resp_sid)
552             raise RequestSidExistsError("Request for this Service is already registered")
553         elif resp_sid in self.__sid_response_dict__.values():
554             logger.error('%s Service with Request-SID=%d and Response-SID=%d not added, because response SID is already registered', self.__log_prefix__(), req_sid, resp_sid)
555             raise ResponseSidExistsError("Response for this Service is already registered")
556         else:
557             self.__sid_response_dict__[req_sid] = resp_sid
558             if req_name is not None:
559                 self.__sid_name_dict__[req_sid] = req_name
560             if resp_name is not None:
561                 self.__sid_name_dict__[resp_sid] = resp_name
562             logger.debug('%s Adding Service with Request=%s and Response=%s', self.__log_prefix__(), req_name or repr(req_sid), resp_name or repr(resp_sid))
563 
564     def add_status(self, status, name):
565         """
566         Method to add a name for a status.
567 
568         :param status: The Status. See class definitions starting with ``STATUS_``.
569         :type status: int
570         :param name: The Name for the Status.
571         :type name: str
572         """
573         self.__status_name_dict[status] = name
574 
575     def authentificate(self, timeout=2):
576         """
577         This method authetificates the client at the server.
578 
579         :param timeout: The timeout for the authentification (requesting seed, sending key and getting authentification_feedback).
580         :type timeout: float
581         :returns: True, if authentification was successfull; False, if not.
582         :rtype: bool
583 
584         .. note:: An authentification will only processed, if a secret had been given on initialisation.
585 
586         .. note:: Client and Server needs to use the same secret.
587         """
588         if self.__secret__ is not None:
589             self.__authentification_state__ = AUTH_STATE_SEED_REQUESTED
590             self.send(SID_AUTH_REQUEST, DID_AUTH_SEED, None)
591             cnt = 0
592             while cnt < timeout * 10:
593                 time.sleep(0.1)
594                 if self.__authentification_state__ == AUTH_STATE_TRUSTED_CONNECTION:
595                     return True
596                 elif self.__authentification_state__ == AUTH_STATE_UNTRUSTED_CONNECTION:
597                     break
598                 cnt += 1
599         return False
600 
601     def check_authentification_state(self):
602         """
603         This Method return the Authitification State as boolean value.
604 
605         :return: True, if authentification state is okay, otherwise False
606         :rtype: bool
607         """
```

```
608         return self.__secret__ is None or self.__authentification_state__ ==
    AUTH_STATE_TRUSTED_CONNECTION
609
610     def connection_established(self):
611         """
612         This Method returns the Connection state including authentification as a boolean value.
613
614         :return: True, if the connection is established (incl. authentification, if a secret has
    been given)
615         :rtype: bool
616         """
617         return self.is_connected() and (self.__secret__ is None or self.
    check_authentification_state())
618
619     def is_connected(self):
620         """
621         This Methods returns Connection state of the Communication Instance :func:`comm_instance.
    is_connected`.
622
623         :return: True if the :class:`comm_instance` is connected, otherwise False..
624         :rtype: bool
625         """
626         return self.__comm_inst__.is_connected()
627
628     def receive(self, service_id, data_id, timeout=1):
629         """
630         This Method returns a message object for a defined message or None, if this message is
    not available after the given timout.
631
632         :param service_id: The Service-ID for the message. See class definitions starting with ``
    SID_``.
633         :type service_id: int
634         :param data_id: The Data-ID for the message.
635         :type data_id: int
636         :param timeout: The timeout for receiving.
637         :type timeout: float
638         :returns: The received data storage object or None, if no data was received.
639         :rtype: data_storage
640         """
641         data = None
642         cnt = 0
643         while data is None and cnt < timeout * 10:
644             try:
645                 data = self.__msg_buffer__.get(service_id, {}).get(data_id, []).pop(0)
646             except IndexError:
647                 data = None
648             cnt += 1
649             time.sleep(0.1)
650         if data is None and cnt >= timeout * 10:
651             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))
652         return data
653
654     def reconnect(self):
655         """
656         This methods initiates a reconnect by calling :func:`comm_instance.reconnect`.
657         """
658         return self.__comm_inst__.reconnect()
659
660     def register_callback(self, service_id, data_id, callback, *args, **kwargs):
```

```
661         """
662         This method registers a callback for the given parameters. Giving ``None`` means, that all Service-IDs or all Data-IDs are used.
663         If a message hitting these parameters has been received, the callback will be executed.
664 
665         :param service_id: The Service-ID for the message. See class definitions starting with ``SID_``.
666         :type service_id: int
667         :param data_id: The Data-ID for the message.
668         :type data_id: int
669 
670         .. note:: The :func:`callback` is priorised in the following order:
671 
672             * Callbacks with defined Service-ID and Data-ID.
673             * Callbacks with a defined Service-ID and all Data-IDs.
674             * Callbacks with a defined Data-ID and all Service-IDs.
675             * Unspecific Callbacks.
676 
677         .. note:: The :func:`callback` is executed with these arguments:
678 
679             **Parameters given at the callback call:**
680 
681             * The first Arguments is the received message as :class:`data_storage` object.
682             * Further arguments given at registration.
683             * Further keyword arguments given at registration.
684 
685             **Return value of the callback:**
686 
687             If the Callback is a Request Callback for a registered Service, the return value has to be a tuple or list with
688 
689             * :const:`response_status`: The response status (see class definitions starting with :const:`STA_*`.
690             * :const:`response_data`: A JSON iterable object to be used as data for the response.
691 
692         .. note:: Only registered services will respond via the callbacks return values with the same data_id.
693         """
694         self.__callbacks__.add(service_id, data_id, callback, *args, **kwargs)
695 
696     def send(self, service_id, data_id, data, status=STATUS_OKAY, timeout=2):
697         """
698         This methods sends out a message with the given content.
699 
700         :param service_id: The Service-ID for the message. See class definitions starting with ``SERVICE_``.
701         :type service_id: int
702         :param data_id: The Data-ID for the message.
703         :type data_id: int
704         :param data: The data to be transfered. The data needs to be json compatible.
705         :type data: str
706         :param status: The Status for the message. All requests should have ``STATUS_OKAY``.
707         :type status: int
708         :param timeout: The timeout for sending data (e.g. time to establish new connection).
709         :type timeout: float
710         :return: True if data had been sent, otherwise False.
711         :rtype: bool
712         """
713         if (self.check_authentification_state() or not self.__authentification_required__(service_id, data_id)) or (service_id in self.__sid_response_dict__.values() and status == STATUS_AUTH_REQUIRED and data is None):
714             self.logger.info(
```

```
715                 '%s TX <- %s, %s, data: "%s"',
716                 self.__log_prefix__(),
717                 self.__get_message_name__(service_id, data_id),
718                 self.__get_status_name__(status),
719                 repr(data)
720             )
721             return self.__comm_inst__.send(self.__build_frame__(service_id, data_id, data, status), timeout=timeout, log_lvl=logging.DEBUG)
722         else:
723             # Authentification required
724             self.logger.warning("%s TX -> Authentification is required. Message %s, %s, data: %s will be ignored.", self.__log_prefix__(), self.__get_message_name__(service_id, data_id), self.__get_status_name__(status), repr(data))
725             return False
726
727
728 class struct_json_protocol(pure_json_protocol):
729     """
730     This Class has the same functionality like :class:`pure_json_protocol`. The message length is less than for :class:`pure_json_protocol`, but the functionality and compatibility is reduced.
731
732     .. note::
733         This class is depreceated and here for compatibility reasons (to support old clients or servers). Usage of :class:`pure_json_protocol` is recommended.
734     """
735     def __init__(self, *args, **kwargs):
736         pure_json_protocol.__init__(self, *args, **kwargs)
737
738     def __analyse_frame__(self, frame):
739         status, service_id, data_id = struct.unpack('>III', frame[0:12])
740         if sys.version_info >= (3, 0):
741             data = json.loads(frame[12:-1].decode('utf-8'))
742         else:
743             data = json.loads(frame[12:-1])
744         return self.__mk_msg__(status, service_id, data_id, data)
745
746     def __build_frame__(self, service_id, data_id, data, status=STATUS_OKAY):
747         frame = struct.pack('>III', status, service_id, data_id)
748         if sys.version_info >= (3, 0):
749             frame += bytes(json.dumps(data), 'utf-8')
750             frame += self.__calc_chksum__(frame)
751         else:
752             frame += json.dumps(data)
753             frame += self.__calc_chksum__(frame)
754         return frame
755
756     def __calc_chksum__(self, raw_data):
757         chksum = 0
758         for b in raw_data:
759             if sys.version_info >= (3, 0):
760                 chksum ^= b
761             else:
762                 chksum ^= ord(b)
763         if sys.version_info >= (3, 0):
764             return bytes([chksum])
765         else:
766             return chr(chksum)
767
768     def __check_frame_checksum__(self, frame):
769         return self.__calc_chksum__(frame[:-1]) == frame[-1:]
```