ETSI TS 103 597-1 V1.1.1 (2021-01)

Methods for Testing and Specification (MTS);
Test Specification for MQTT;
Part 1: Conformance Tests

Reference
DTS/MTS-TSTMQTT-1

Keywords
conformance, TSS&TP

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

The present document is part 1 of a multi-part deliverable covering the MQTT protocol as identified below:

Part 1: "Conformance Tests";

Part 2: "Security Tests";

Part 3: "Performance Tests".

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

While the Internet of Things (IoT) is on the rise, the quality assurance of interconnected systems becomes an ever-increasing challenge. Within the last years, many different IoT protocols came to the fore. The MQ Telemetry Transport (MQTT) protocol is one of the most popular representatives as many surveys have shown.

Although many implementations for the MQTT protocol exist, it lacks in satisfying quality assurance. While many IoT components communicate over standardized protocols, communication protocols for IoT like MQTT or CoAP evolved over time without a holistic approach for quality assurance.

In the present document the conformance testing is presented. It provides a basis for interoperability testing and performance testing. The latter is presented in ETSI TS 103 597-3 [i.3].

1 Scope

The present document provides a test specification, i.e. an overall test suite structure and catalogue of test purposes for the MQ Telemetry Transport (MQTT). It will be a reference base for both client-side test campaigns and server-side test campaigns addressing the conformance issues.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] OASIS Standard: "MQTT Version 3.1.1".

[2] ETSI ES 203 119-4: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 4: Structured Test Objective Specification (Extension)".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ISO/IEC 9646-1: " Information technology -- Open Systems Interconnection -- Conformance testing methodology and framework -- Part 1: General concepts".

[i.2] ETSI ES 202 951: "Methods for Testing and Specification (MTS); Model-Based Testing (MBT); Requirements for Modelling Notations".

[i.3] ETSI TS 103 597-3: "Methods for Testing and Specification (MTS); Test Specification for MQTT; Part 3: Performance Tests".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

conformance: extent to which an implementation of a standard satisfies the requirements expressed in that standard

conformance testing: process to verify to what extent the IUT conforms to the standard

Implementation Under Test (IUT): implementation of one or more Open Systems Interconnection (OSI) protocols in an adjacent user/provider relationship, being the part of a real open system, which is to be studied by testing (ISO/IEC 9646-1 [i.1])

system under test: real open system in which the implementation under test resides (ETSI ES 202 951 [i.2])

test purpose: non-formal high-level description of a test, mainly using text

test suite structure: document defining (hierarchical) grouping of test cases according to some rules

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

IUT	Implementation Under Test
MQTT	MQ Telemetry Transport
SUT	System Under Test
TDL	Test Description Language
TDL-TO	Test Description Language - Test Objectives
TSS	Test Suite Structure

4 Test Suite Structure

4.0 Introduction

In the first one an MQTT server as SUT is considered and in the latter, an MQTT client as SUT is considered

The structure itself is partly derived from the MQTT specification [1] but changed due to overlapping functions that cannot be tested separately.

4.1 Broker as SUT

1) All mandatory message data fields
    a) CONNECT Control Packet
        i) Fixed Header
            1) Header Flags
        ii) Variable Header
            1) Protocol Name
            2) Protocol Level
            3) Reserved Flags
            4) Last Will Testament Flags
            5) Credentials Flags

iii) Payload

1) Client Identifier

2) Will Topic

3) Credentials

b) CONNACK Control Packet

i) Fixed Header

ii) Variable Header

1) Clean Session

2) Present Session

3) Return Codes

c) SUBSCRIBE Control Packet

i) Fixed Header

1) Header Flags

ii) Variable Header

1) Packet Identifier

iii) Payload

1) UTF-8 Encoding

2) Topic Filter

3) Requested QoS

d) SUBACK Control Packet

i) Fixed Header

1) Header Flags

ii) Variable Header

1) Packet Identifier

iii) Payload

1) Return Codes

e) UNSUBSCRIBE Control Packet

i) Fixed Header

1) Header Flags

ii) Variable Header

1) Packet Identifier

iii) Payload

1) UTF-8 Encoding

2) Topic Filters

f) UNSUBACK Control Packet

i) Fixed Header

ii) Variable Header

g) PINGREQ Control Packet

i) Fixed Header

h) PINGRESP Control Packet

i) Fixed Header

i) DISCONNECT Control Packet

i) Fixed Header

2) Protocol features

a) General

i) QoS levels

ii) Delivery retransmission

iii) Retained messages

iv) Message ordering

v) Anonymous client identifier

b) Connect/disconnect (session handling)

i) Credentials

ii) Session initiation

iii) Session states

c) Subscribe

d) Unsubscribe

e) Immediate publish (w/o awaiting for CONNACK)

f) Last Will and Testament message

g) Heartbeats: keepAlive values

h) Topic names/filters

i) Error handling

4.2 Client as SUT

1) All mandatory message data fields

a) CONNECT Control Packet

b) CONNACK Control Packet

c) PUBLISH Control Packet

d) PUBACK Control Packet

e) PUBREC Control Packet

f) UNSUBACK Control Packet

g) PUBREL Control Packet

h) PUBCOMP Control Packet

i) SUBSCRIBE Control Packet

j) UNSUBSCRIBE Control Packet

k) DISCONNECT Control Packet

2) Protocol features

a) keepAlive values

4.3 TP naming convention

TPs are numbered, starting at 001, within each main scope. The main scopes are organized according to the TSS. Some TPs may not have a second level scope.

Table 1: TP identifier naming convention scheme

Identifier: **TP_<protocol>_<iut>_<scope>_<2nd_lvl_scope>*_<number>***

TP	=	Test Purpose	Fixed to TP
<protocol>	=	Protocol name	Fixed to MQTT
<iut>	=	Type of IUT	Client or Broker
<scope>	=	Main scope	Scope of the protocol (feature)
			<CONTROL PACKET> Name of the scoped Control Packet
			FEAT Protocol Features
<2nd_lvl_scope>	=	Second level scope	RTND Retained Messages
<number>	=	Sequential number	From 001 to 999

*optional

4.4 TP structure

Each TP has been written in TDL-TO and thus in a structured manner which is consistent with all other TPs. The intention of this is to make the TPs more formal. In addition, a more readable format is provided by generating tables out of the TDL-TO format. The defined structure, that has been used, is illustrated in table 2. This table should be read in conjunction with any TP, i.e. please use a TP as an example to facilitate the full comprehension of table 2. All structures are defined formally in the TDL Specification ETSI ES 203 119-4 [2].

Table 2: Structure of a single TP

TP part	Text	Example
Header	<Identifier> <Test objective> <Reference> <PICS reference>	*see table 1* *"The IUT has to close network connect …"* *[MQTT-3.2.2-6]* *PIC_BROKER_BASIC*
Initial condition (optional)	Free text description of the condition that the IUT has reached before the test purpose applies.	*… the IUT entity having a present session for the CLIENT_ID entity …*
Start point	Describes the full logic of the test purpose. Includes trigger and expected behavior of the IUT.	*Expected behavior* *ensure that { … }*
Trigger	One or more actions that trigger an expected response of the IUT. Mostly a set of different messages the IUT receives.	*when {* *the IUT entity receives a CONNECT message containing header_flags indicating value '1111'B;* *}*
Expected behavior	Describes the response that the IUT sends after receiving a certain (set of) messages. This response describes the pass criteria	*then {* *the IUT entity closes the TCP_CONNECTION* *}*

5 Test Purposes for MQTT Broker

TP Id	TP_MQTT_BROKER_CONNECT_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in CONNECT Control Packet are invalid.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '1111'B;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_002
Test Objective	Verify that the IUT either disconnects the client or continues processing the CONNECT Control Packet if the protocol name does not correspond to 'MQTT'.
Reference	[MQTT-3.1.2-1], [MQTT-3.1.4-4]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
  when {
  the IUT receives a CONNECT message containing
  header_flags indicating value '0000'B,
  protocol_name corresponding to PROTOCOL_NAME_INVALID,
  protocol_level indicating value 0x04;
  }
  then {
  the IUT closes the TCP_CONNECTION
  // TODO: missing in TTCN-3 Implementation
  or the IUT sends a CONNACK message containing
  connect_return_code indicating value 0x00;
  }
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_003
Test Objective	Verify that the IUT responds to supported protocol levels (in scope: MQTT-3.1.1) with the return code 0x00.
Reference	[MQTT-3.1.2-2], [MQTT-3.1.4-4]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
  when {
  the IUT receives a CONNECT message containing
  header_flags indicating value '0000'B,
  protocol_name corresponding to PROTOCOL_NAME,
  protocol_level indicating value 0x04;
  }
  then {
  the IUT sends a CONNACK message containing
  connect_return_code indicating value 0x00;
  }
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_004
Test Objective	Verify that the IUT validates the reserved flags in the CONNECT Control Packet.
Reference	[MQTT-3.1.2-3], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
reserved_field indicating value '1'B;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_005
Test Objective	Verify that the IUT validates the will_topic and will_message fields if the will_flag is set to 1.
Reference	[MQTT-3.1.2-9], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_LWT
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
will_flag indicating value '1'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
will_topic indicating value omit,
will_message indicating value omit;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_006
Test Objective	Verify that the IUT validates the the will_topic and will_message fields to be omitted if the will_flag is set to 0.
Reference	[MQTT-3.1.2-11], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_LWT and PICS_BROKER_RTND
Initial Conditions	
Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a CONNECT message containing
         header_flags indicating value '0000'B,
         protocol_name corresponding to PROTOCOL_NAME,
         protocol_level indicating value 0x04,
         connect_flags containing
            will_flag indicating value '0'B,
            will_qos corresponding to AT_LEAST_ONCE,
            will_retain indicating value '1'B,
            reserved_field indicating value '0'B;
         ,
         payload containing
            will_topic corresponding to PX_WILL_TOPIC,
            will_message corresponding to PX_WILL_MESSAGE;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_007
Test Objective	Verify that the IUT validates the will_qos field to be set to 0 if the will_flag is set to 0.
Reference	[MQTT-3.1.2-13], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a CONNECT message containing
         header_flags indicating value '0000'B,
         protocol_name corresponding to PROTOCOL_NAME,
         protocol_level indicating value 0x04,
         connect_flags containing
            will_flag indicating value '0'B,
            will_qos corresponding to AT_LEAST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_008
Test Objective	Verify that the IUT validates the will_qos field and rejects connections with an invalid will_qos value.
Reference	[MQTT-3.1.2-14], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_LWT
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
will_flag indicating value '1'B,
will_qos corresponding to INVALID_QOS,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_009
Test Objective	Verify that the IUT validates the will_qos field if the will_flag is set to 1.
Reference	[MQTT-3.1.2-14], [MQTT-3.1.4-4]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_LWT
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
will_flag indicating value '1'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
;
}
then {
the IUT sends a CONNACK message containing
connect_return_code indicating value 0x00;
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_010
Test Objective	Verify that the IUT validates the will_flag and will_retain flags to be set correctly.
Reference	[MQTT-3.1.2-15], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
will_flag indicating value '0'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '1'B,
reserved_field indicating value '0'B;
,
payload containing
will_topic indicating value omit,
will_message indicating value omit;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_011
Test Objective	Verify that the IUT validates the will_retain flag to be set to 0 if the will_flag is set to 0.
Reference	[MQTT-3.1.2-15], [MQTT-3.1.4-4]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
will_flag indicating value '0'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
will_topic indicating value omit,
will_message indicating value omit;
;
}
then {
the IUT sends a CONNACK message containing
connect_return_code indicating value 0x00;
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_012
Test Objective	Verify that the IUT validates the password flag to be set to 0 if the user_name_flag is set to 0.
Reference	[MQTT-3.1.2-22], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_AUTH
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
user_name_flag indicating value '0'B,
password_flag indicating value '1'B,
will_flag indicating value '0'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_013
Test Objective	Verify that the IUT validates the username field to be omitted if the user_name_flag is set to 0.
Reference	[MQTT-3.1.2-18], [MQTT-3.1.2-22], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_AUTH
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
user_name_flag indicating value '0'B,
password_flag indicating value '0'B,
will_flag indicating value '0'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
user_name corresponding to PX_MQTT_USER_NAME,
password indicating value omit;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_014
Test Objective	Verify that the IUT validates a payload is present if the user_name_flag is set to 1.
Reference	[MQTT-3.1.2-19], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_AUTH
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
user_name_flag indicating value '1'B,
password_flag indicating value '0'B,
will_flag indicating value '0'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
user_name indicating value omit,
password indicating value omit;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_015
Test Objective	Verify that the IUT validates the password field to be omitted if the password_flag is set to 0.
Reference	[MQTT-3.1.2-20], [MQTT-3.1.2-22], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_AUTH
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
user_name_flag indicating value '0'B,
password_flag indicating value '0'B,
will_flag indicating value '0'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
user_name indicating value omit,
password corresponding to PX_MQTT_PASSWORD;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_016
Test Objective	Verify that the IUT validates the password field to be present if the password_flag is set to 1.
Reference	[MQTT-3.1.2-21], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_AUTH
Initial Conditions	
Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a CONNECT message containing
         header_flags indicating value '0000'B,
         protocol_name corresponding to PROTOCOL_NAME,
         protocol_level indicating value 0x04,
         connect_flags containing
            user_name_flag indicating value '1'B,
            password_flag indicating value '1'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
         ,
         payload containing
            user_name corresponding to PX_MQTT_USER_NAME,
            password indicating value omit;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_017
Test Objective	Verify that the IUT validates the client_identifier to be between 1 and 23 UTF-8 encoded bytes in length.
Reference	[MQTT-3.1.3-5], [MQTT-3.1.4-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a CONNECT message containing
         header_flags indicating value '0000'B,
         protocol_name corresponding to PROTOCOL_NAME,
         protocol_level indicating value 0x04,
         connect_flags containing
            clean_session indicating value '1'B,
            user_name_flag indicating value '0'B,
            password_flag indicating value '0'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
         ,
         payload containing
            client_identifier corresponding to CLIENT_ID_24_BYTES;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_018
Test Objective	Verify that the IUT validates the client_identifier to contain only alphanumeric characters [0-9a-zA-Z].
Reference	[MQTT-3.1.3-5], [MQTT-3.1.4-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a CONNECT message containing
         header_flags indicating value '0000'B,
         protocol_name corresponding to PROTOCOL_NAME,
         protocol_level indicating value 0x04,
         connect_flags containing
            clean_session indicating value '1'B,
            user_name_flag indicating value '0'B,
            password_flag indicating value '0'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
         ,
         payload containing
            client_identifier corresponding to CLIENT_ID_NON_ALPHA_NUM;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_019
Test Objective	Verify that the IUT accepts client_identifiers of zero byte length.
Reference	[MQTT-3.1.3-6], [MQTT-3.1.3-7], [MQTT-3.1.4-4]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a CONNECT message containing
         header_flags indicating value '0000'B,
         protocol_name corresponding to PROTOCOL_NAME,
         protocol_level indicating value 0x04,
         connect_flags containing
            clean_session indicating value '1'B,
            user_name_flag indicating value '0'B,
            password_flag indicating value '0'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
         ,
         payload containing
            client_identifier corresponding to CLIENT_ID_ZERO_BYTES;
         ;
   }
   then {
         // TODO: Standards says: MAY allow
```

```
        the IUT sends a CONNACK message containing
        connect_return_code indicating value 0x00;
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_020
Test Objective	Verify that the IUT validates the client_identifier to be a well-formed UTF-8 encoded string.
Reference	[MQTT-1.5.3-1], [MQTT-3.1.3-4], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0x04,
        connect_flags containing
            clean_session indicating value '0'B,
            user_name_flag indicating value '0'B,
            password_flag indicating value '0'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
        ,
        payload containing
            client_identifier corresponding to CLIENT_ID_D800;
        ;
    }
    then {
        the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_021
Test Objective	Verify that the IUT validates the client_identifier to be a well-formed UTF-8 encoded string.
Reference	[MQTT-1.5.3-2], [MQTT-3.1.3-4], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0x04,
        connect_flags containing
            clean_session indicating value '0'B,
            user_name_flag indicating value '0'B,
            password_flag indicating value '0'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
        ,
        payload containing
            client_identifier corresponding to CLIENT_ID_0000;
        ;
```

```
   }
   then {
        the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_022
Test Objective	Verify that the IUT validates the will_topic to be a well-formed UTF-8 encoded string.
Reference	[MQTT-1.5.3-1], [MQTT-3.1.3-10], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_LWT

Initial Conditions

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0x04,
        connect_flags containing
           clean_session indicating value '0'B,
           user_name_flag indicating value '0'B,
           password_flag indicating value '0'B,
           will_flag indicating value '1'B,
           will_qos corresponding to AT_MOST_ONCE,
           will_retain indicating value '0'B,
           reserved_field indicating value '0'B;
        ,
        payload containing
           will_topic indicating value WILL_TOPIC_D800,
           will_message corresponding to PX_WILL_MESSAGE;
        ;
   }
   then {
        the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_023
Test Objective	Verify that the IUT validates the will_topic to be a well-formed UTF-8 encoded string.
Reference	[MQTT-1.5.3-2], [MQTT-3.1.3-10], [MQTT-3.1.4-1], [MQTT-3.2.2-6], [MQTT-4.7.3-2]
PICS Selection	PICS_BROKER_LWT

Initial Conditions

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0x04,
        connect_flags containing
           clean_session indicating value '0'B,
           user_name_flag indicating value '0'B,
           password_flag indicating value '0'B,
           will_flag indicating value '1'B,
           will_qos corresponding to AT_MOST_ONCE,
           will_retain indicating value '0'B,
           reserved_field indicating value '0'B;
        ,
        payload containing
```

will_topic indicating value WILL_TOPIC_0000,
will_message corresponding to PX_WILL_MESSAGE;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_024
Test Objective	Verify that the IUT validates the will_topic to not contain single-level topic filters.
Reference	[MQTT-4.7.1-1], [MQTT-3.1.4-1]
PICS Selection	PICS_BROKER_LWT
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
connect_flags containing
will_flag indicating value '1'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
will_topic indicating value TOPIC_FILTER_SINGLE_LEVEL,
will_message corresponding to PX_WILL_MESSAGE;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_025
Test Objective	Verify that the IUT validates the will_topic to not contain multi-level topic filters.
Reference	[MQTT-4.7.1-1], [MQTT-3.1.4-1]
PICS Selection	PICS_BROKER_LWT
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
connect_flags containing
will_flag indicating value '1'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
will_topic indicating value TOPIC_FILTER_MULTI_LEVEL,
will_message corresponding to PX_WILL_MESSAGE;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_026
Test Objective	Verify that the IUT validates the user_name to be a well-formed UTF-8 encoded string.
Reference	[MQTT-1.5.3-1], [MQTT-3.1.3-11], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_AUTH
Initial Conditions	
Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a CONNECT message containing
         header_flags indicating value '0000'B,
         protocol_name corresponding to PROTOCOL_NAME,
         protocol_level indicating value 0x04,
         connect_flags containing
            clean_session indicating value '0'B,
            user_name_flag indicating value '1'B,
            password_flag indicating value '1'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
         ,
         payload containing
            user_name corresponding to MQTT_USER_NAME_INVALID_UTF8,
            password corresponding to PX_MQTT_PASSWORD;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_027
Test Objective	Verify that the IUT validates the user_name to be a well-formed UTF-8 encoded string.
Reference	[MQTT-1.5.3-2], [MQTT-3.1.3-11], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PICS_BROKER_AUTH
Initial Conditions	
Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a CONNECT message containing
         header_flags indicating value '0000'B,
         protocol_name corresponding to PROTOCOL_NAME,
         protocol_level indicating value 0x04,
         connect_flags containing
            clean_session indicating value '0'B,
            user_name_flag indicating value '1'B,
            password_flag indicating value '1'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
         ,
         payload containing
            user_name corresponding to MQTT_USER_NAME_INVALID_UTF8,
            password corresponding to PX_MQTT_PASSWORD;
         ;
   }
   then {
```

the IUT closes the TCP_CONNECTION } }
Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_028
Test Objective	Verify that the IUT validates the first MQTT Control Packet sent from the client to the server after a TCP connection is a MQTT CONNECT.
Reference	[MQTT-3.1.0-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the TEST_SYSTEM having a TCP_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_029
Test Objective	Verify that the IUT detects multiple MQTT CONNECT Control Packets sent from a client within a single session as a protocol violation.
Reference	[MQTT-3.1.0-2]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_030
Test Objective	Verify that the IUT detects multiple clients with the same client_identifier and disconnects the existing client.
Reference	[MQTT-3.1.4-2]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_1 established the MQTT_CONNECTION containing
payload containing
client_identifier corresponding to PX_CLIENT_ID;
;
to the IUT
}	

Expected Behaviour

ensure that {
  when {
  the IUT receives a CONNECT message containing
  payload containing
  client_identifier corresponding to PX_CLIENT_ID;
  ;
  from the CLIENT_2
  }
  then {
  the IUT closes the TCP_CONNECT to the CLIENT_1
  and
  the IUT sends a CONNACK message to the CLIENT_2
  }
}

Final Conditions

TP Id	TP_MQTT_BROKER_CONNECT_031
Test Objective	Verify that the IUT validates all topic names to be at least one character long.
Reference	[MQTT-4.7.3-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
  when {
  the IUT receives a CONNECT message containing
  header_flags indicating value '0000'B,
  protocol_name corresponding to PROTOCOL_NAME,
  protocol_level indicating value 0x04,
  connect_flags containing
  will_flag indicating value '1'B,
  will_qos corresponding to AT_LEAST_ONCE,
  will_retain indicating value '0'B,
  reserved_field indicating value '0'B;
  ,
  payload containing
  will_topic corresponding to TOPIC_NAME_ZERO_CHARS,
  will_message corresponding to PX_WILL_MESSAGE;
  ;
  }
  then {
  the IUT closes the TCP_CONNECTION
  }
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNECT_032
Test Objective	Verify that the IUT does not process any data sent by the client after a rejected CONNECT Control Packet.
Reference	[MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_AUTH
Initial Conditions	

with {
  the IUT received a CONNECT message containing
  connect_flags containing
  user_name_flag indicating value '1'B,
  password_flag indicating value '1'B,
  payload containing
  user_name corresponding to MQTT_USER_NAME_INVALID,
  password corresponding to MQTT_PASSWORD_INVALID;;;
}	

Expected Behaviour

ensure that {
  when {
  the IUT receives a SUBSCRIBE message
  }
  then {
  the IUT sends a CONNACK message containing
  connect_return_code indicating value 0x05;
  and
  the IUT sends no SUBACK message
  and
  the IUT closes the TCP_CONNECTION
  }
}

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_001
Test Objective	Verify that the IUT replies with a CONNACK Control Packet with valid header flags.
Reference	[MQTT-2.2.2-1], [MQTT-3.1.4-4], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
  when {
  the IUT receives a CONNECT message containing
  header_flags indicating value '0000'B,
  protocol_name corresponding to PROTOCOL_NAME,
  protocol_level indicating value 0x04,
  connect_flags containing
  clean_session indicating value '0'B,
  user_name_flag indicating value '0'B,
  password_flag indicating value '0'B,
  will_flag indicating value '0'B,
  will_qos corresponding to AT_MOST_ONCE,
  will_retain indicating value '0'B,
  reserved_field indicating value '0'B;
  ;
  }
  then {
  the IUT sends a CONNACK message containing
  header_flags indicating value '0000'B,
  connect_return_code indicating value 0x00;
  }
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_CONNACK_002
Test Objective	Verify that the IUT responds to a CONNECT Control Packet with clean_session set to 1 with a CONNACK Control Packet with session_present_flag set to 0.
Reference	[MQTT-3.2.2-1], [MQTT-3.1.4-4], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

Expected Behaviour

ensure that {
  when {
  the IUT receives a CONNECT message containing
  header_flags indicating value '0000'B,
  protocol_name corresponding to PROTOCOL_NAME,
  protocol_level indicating value 0x04,
  connect_flags containing
  clean_session indicating value '1'B,
  user_name_flag indicating value '0'B,
  password_flag indicating value '0'B,
  will_flag indicating value '0'B,
  will_qos corresponding to AT_MOST_ONCE,
  will_retain indicating value '0'B,
  reserved_field indicating value '0'B;
  ,
  payload containing
  client_identifier corresponding to VALID_CLIENT_ID;
  ;
  }
  then {
  the IUT sends a CONNACK message containing
  header_flags indicating value '0000'B,
  session_present_flag indicating value '0'B,
  connect_return_code indicating value 0x00;
  }
}

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_003
Test Objective	Verify that the IUT responds to a CONNECT Control Packet for a present session with a CONNACK Control Packet with session_present_flag set to 1.
Reference	[MQTT-3.2.2-2], [MQTT-3.1.4-4], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_RTND
Initial Conditions	

with {
  the IUT having a present session for the CLIENT_ID
}	

Expected Behaviour	

ensure that {
  when {
  the IUT receives a CONNECT message containing
  header_flags indicating value '0000'B,
  protocol_name corresponding to PROTOCOL_NAME,
  protocol_level indicating value 0x04,
  connect_flags containing
  clean_session indicating value '0'B,
  user_name_flag indicating value '0'B,
  password_flag indicating value '0'B,
  will_flag indicating value '0'B,
  will_qos corresponding to AT_MOST_ONCE,
  will_retain indicating value '0'B,
  reserved_field indicating value '0'B;
  ,
  payload containing
  client_identifier corresponding to PX_CLIENT_ID;
  ;
  }
  then {
  the IUT sends a CONNACK message containing
  header_flags indicating value '0000'B,
  session_present_flag indicating value '1'B,
  connect_return_code indicating value 0x00;
  }
}	

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_004
Test Objective	Verify that the IUT responds to a CONNECT Control Packet with clean_session set to 1 but not having a present session for this client_identifier with a CONNACK Control Packet with session_present_flag set to 0.
Reference	[MQTT-3.2.2-3], [MQTT-3.1.4-4], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_RTND

Initial Conditions

```
with {
        the IUT having no present session for the CLIENT_ID
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0x04,
        connect_flags containing
           clean_session indicating value '0'B,
           user_name_flag indicating value '0'B,
           password_flag indicating value '0'B,
           will_flag indicating value '0'B,
           will_qos corresponding to AT_MOST_ONCE,
           will_retain indicating value '0'B,
           reserved_field indicating value '0'B;
        ,
        payload containing
           client_identifier corresponding to PX_CLIENT_ID;
        ;
   }
   then {
        the IUT sends a CONNACK message containing
        header_flags indicating value '0000'B,
        session_present_flag indicating value '0'B,
        connect_return_code indicating value 0x00;
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_005
Test Objective	Verify that the IUT responds to protocol levels which it does not support with return code 0x01.
Reference	[MQTT-3.1.2-2], [MQTT-3.2.2-4], [MQTT-3.2.2-5], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0xFF,
        connect_flags containing
           clean_session indicating value '0'B,
           user_name_flag indicating value '0'B,
           password_flag indicating value '0'B,
           will_flag indicating value '0'B,
           will_qos corresponding to AT_MOST_ONCE,
           will_retain indicating value '0'B,
           reserved_field indicating value '0'B;
        ,
```

```
        payload containing
            client_identifier corresponding to PX_CLIENT_ID;
        ;
    }
    then {
        the IUT sends a CONNACK message containing
        header_flags indicating value '0000'B,
        session_present_flag indicating value '0'B,
        connect_return_code indicating value 0x01;
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_006
Test Objective	Verify that the IUT responds to CONNECT Control Packets with a zero-byte client_identifier and clean_session set to 0 with CONNACK with connect_return_code set to 0x02 and close the network connection.
Reference	[MQTT-3.1.3-8], [MQTT-3.1.3-9], [MQTT-3.2.2-4], [MQTT-3.2.2-5], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_RTND

Initial Conditions

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0x04,
        connect_flags containing
            clean_session indicating value '0'B,
            user_name_flag indicating value '0'B,
            password_flag indicating value '0'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
        ,
        payload containing
            client_identifier corresponding to CLIENT_ID_ZERO_BYTES;
        ;
    }
    then {
        the IUT sends a CONNACK message containing
        header_flags indicating value '0000'B,
        session_present_flag indicating value '0'B,
        connect_return_code indicating value 0x02;
        and the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_007
Test Objective	Verify that the IUT responds to a CONNECT Control Packet with a malformed user_name with CONNACK with connect_return_code set to 0x04.
Reference	[MQTT-3.2.2-4], [MQTT-3.2.2-5], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_AUTH

Initial Conditions

Expected Behaviour

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
clean_session indicating value '0'B,
user_name_flag indicating value '1'B,
password_flag indicating value '1'B,
will_flag indicating value '0'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
client_identifier corresponding to PX_CLIENT_ID,
user_name corresponding to MQTT_USER_NAME_INVALID_UTF8,
password corresponding to PX_MQTT_PASSWORD;
;
}
then {
the IUT sends a CONNACK message containing
header_flags indicating value '0000'B,
session_present_flag indicating value '0'B,
connect_return_code indicating value 0x04;
and the IUT closes the TCP_CONNECTION
}
}

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_008
Test Objective	Verify that the IUT responds to a CONNECT Control Packet with a invalid user_name with CONNACK with connect_return_code set to 0x05.
Reference	[MQTT-3.2.2-4], [MQTT-3.2.2-5], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_AUTH
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT receives a CONNECT message containing
header_flags indicating value '0000'B,
protocol_name corresponding to PROTOCOL_NAME,
protocol_level indicating value 0x04,
connect_flags containing
clean_session indicating value '0'B,
user_name_flag indicating value '1'B,
password_flag indicating value '1'B,
will_flag indicating value '0'B,
will_qos corresponding to AT_MOST_ONCE,
will_retain indicating value '0'B,
reserved_field indicating value '0'B;
,
payload containing
client_identifier corresponding to PX_CLIENT_ID,
user_name corresponding to MQTT_USER_NAME_INVALID,
password corresponding to PX_MQTT_PASSWORD;
;
}
then {
the IUT sends a CONNACK message containing
header_flags indicating value '0000'B,
session_present_flag indicating value '0'B,
connect_return_code indicating value 0x05;	

```
        and the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_009
Test Objective	Verify that the IUT responds to a CONNECT Control Packet with a invalid password with CONNACK with connect_return_code set to 0x05.
Reference	[MQTT-3.2.2-4], [MQTT-3.2.2-5], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_AUTH

Initial Conditions

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0x04,
        connect_flags containing
            clean_session indicating value '0'B,
            user_name_flag indicating value '1'B,
            password_flag indicating value '1'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
        ,
        payload containing
            client_identifier corresponding to PX_CLIENT_ID,
            user_name corresponding to PX_MQTT_USER_NAME,
            password corresponding to MQTT_PASSWORD_INVALID;
        ;
    }
    then {
        the IUT sends a CONNACK message containing
        header_flags indicating value '0000'B,
        session_present_flag indicating value '0'B,
        connect_return_code indicating value 0x05;
        and the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_CONNACK_010
Test Objective	Verify that the IUT responds with CONNECT with connect_return_code set to 0x03 if the MQTT service is unavailable.
Reference	[MQTT-3.2.2-4], [MQTT-3.2.2-5], [MQTT-3.2.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the IUT having no available service for the MQTT_CONNECTION
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a CONNECT message containing
        header_flags indicating value '0000'B,
        protocol_name corresponding to PROTOCOL_NAME,
        protocol_level indicating value 0x04,
        connect_flags containing
            clean_session indicating value '0'B,
            user_name_flag indicating value '0'B,
```

```
            password_flag indicating value '0'B,
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B,
            reserved_field indicating value '0'B;
        ,
        payload containing
            client_identifier corresponding to PX_CLIENT_ID;
        ;
    }
    then {
        the IUT sends a CONNACK message containing
        header_flags indicating value '0000'B,
        session_present_flag indicating value '0'B,
        connect_return_code indicating value 0x03;
        and the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_PUBLISH_001
Test Objective	Verify that the IUT accepts only QoS 0 PUBLISH Control Packets with the dup_flag set to 0.
Reference	[MQTT-3.3.1-2]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a PUBLISH message containing
      header containing
        dup_flag indicating value '1'B,
        qos_level corresponding to AT_MOST_ONCE;
      ;
    }
    then {
        the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_PUBLISH_002
Test Objective	Verify that the IUT accepts only PUBLISH Control Packets with a valid QoS level.
Reference	[MQTT-2.2.2-1], [MQTT-3.3.1-4]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a PUBLISH message containing
      header containing
        qos_level corresponding to INVALID_QOS;
      ;
    }
    then {
        the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_PUBLISH_003
Test Objective	Verify that the IUT validates the topic_name in a PUBLISH Control Packet to be a well-formed UTF-8 encoded string.
Reference	[MQTT-3.3.2-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_INVALID_UTF8;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBLISH_004
Test Objective	Verify that the IUT validates the topic_name in a PUBLISH Control Packet to not contain multi-level wildcard characters.
Reference	[MQTT-3.3.2-2], [MQTT-4.7.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_WC_MULTI_LVL;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBLISH_005
Test Objective	Verify that the IUT validates the topic_name in a PUBLISH Control Packet to not contain single-level wildcard characters.
Reference	[MQTT-3.3.2-2], [MQTT-4.7.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_WC_SINGLE_LVL;
}
then {
the IUT closes the TCP_CONNECTION	

} }
Final Conditions

TP Id	TP_MQTT_BROKER_PUBLISH_006
Test Objective	Verify that the IUT validates the topic_name in a PUBLISH Control Packet to be at least on character long.
Reference	[MQTT-4.7.3-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_ZERO_CHARS;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBLISH_007
Test Objective	Verify that the IUT validates the topic_name in a PUBLISH Control Packet to not contain the null character (Unicode U+0000).
Reference	[MQTT-4.7.3-2], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_0000;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBLISH_008
Test Objective	Verify that the IUT rejects QoS 0 PUBLISH Control Packets with the dup_flag set to 1.
Reference	[MQTT-4.3.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour

ensure that {
  when {
  the IUT receives a PUBLISH message containing
  header containing
  dup_flag indicating value '1'B,
  qos_level corresponding to AT_MOST_ONCE;
  ;
  }
  then {
  the IUT closes the TCP_CONNECTION
  }
}

Final Conditions

TP Id	TP_MQTT_BROKER_PUBLISH_009
Test Objective	Verify that the IUT validates a QoS 0 PUBLISH Control Packet to not contain a packet_identifier.
Reference	[MQTT-2.3.1-5], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
  the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
  when {
  the IUT receives a PUBLISH message containing
  header containing
  qos_level corresponding to AT_MOST_ONCE,
  packet_identifier corresponding to PACKET_ID;
  ;
  }
  then {
  the IUT closes the TCP_CONNECTION
  }
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBLISH_010
Test Objective	Verify that the IUT validates a QoS 1 PUBLISH Control Packet to contain a non-zero 16-bit Packet Identifier.
Reference	[MQTT-2.3.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_QOS_1
Initial Conditions	

with {
  the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
  when {
  the IUT receives a PUBLISH message containing
  header containing
  qos_level corresponding to AT_LEAST_ONCE,
  packet_identifier corresponding to PACKET_ID_ZERO;
  ;
  }
  then {
  the IUT closes the TCP_CONNECTION
  }
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBLISH_011
Test Objective	Verify that the IUT validates a QoS 2 PUBLISH Control Packet to contain a non-zero 16-bit Packet Identifier.
Reference	[MQTT-2.3.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_QOS_2
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
packet_identifier corresponding to PACKET_ID_ZERO;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBLISH_012
Test Objective	Verify that the IUT handles not authorized PUBLISH Control Packets with either a positive PUBACK Control Packet or by closing the network connection.
Reference	[MQTT-3.3.5-2]
PICS Selection	PICS_BROKER_QOS_1
Initial Conditions	

with {
the TEST_SYSTEM having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to AT_LEAST_ONCE,
topic_name corresponding to TOPIC_NAME_RESTRICTED;
;
}
then {
the IUT sends a PUBACK message
or
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBACK_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in PUBACK Control Packet are invalid.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2]
PICS Selection	PICS_BROKER_QOS_1
Initial Conditions	
with { the CLIENT_1 having a MQTT_CONNECTION to the IUT and the CLIENT_2 having a MQTT_CONNECTION to the IUT and the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing qos_level corresponding to AT_LEAST_ONCE;	

```
      and
      the CLIENT_1 delivered a PUBLISH message containing
        qos_level corresponding to AT_LEAST_ONCE,
        topic_name corresponding to PX_PUBLISH_TOPIC;
      to the IUT
}
```

Expected Behaviour

```
ensure that {
  when {
      the IUT sends a PUBLISH message containing
    header containing
     qos_level corresponding to AT_LEAST_ONCE,
    topic_name corresponding to PX_PUBLISH_TOPIC;
    ;
    to the CLIENT_2
    and
    the IUT receives a PUBACK message containing
    header_flags indicating value '1111'B;
    from the CLIENT_2
  }
  then {
      the IUT closes the TCP_CONNECTION to the CLIENT_2
  }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_PUBACK_002
Test Objective	Verify that the IUT sends PUBACK Control Packets in the order in which the corresponding QoS 1 PUBLISH Control Packets were received.
Reference	[MQTT-4.6.0-2], [MQTT-3.3.4-1], [MQTT-4.6.0-6], [MQTT-2.3.1-6]
PICS Selection	PICS_BROKER_QOS_1

Initial Conditions

```
with {
      the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
  when {
      the IUT receives a PUBLISH message containing
    header containing
      qos_level corresponding to AT_LEAST_ONCE,
    topic_name corresponding to PX_PUBLISH_TOPIC,
    packet_identifier corresponding to PACKET_ID_1;
    ;
    and
    the IUT receives a PUBLISH message containing
    header containing
      qos_level corresponding to AT_LEAST_ONCE,
    topic_name corresponding to PX_PUBLISH_TOPIC,
    packet_identifier corresponding to PACKET_ID_2;
    ;
  }
  then {
      the IUT sends a PUBACK message containing
     packet_identifier corresponding to PACKET_ID_1;
    and
    the IUT sends a PUBACK message containing
     packet_identifier corresponding to PACKET_ID_2;
  }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_PUBREC_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in PUBREC Control Packet are invalid.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2]
PICS Selection	PICS_BROKER_QOS_2
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
qos_level corresponding to EXACTLY_ONCE;
and
the CLIENT_1 delivered a PUBLISH message containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC;
to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT sends a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC;
;
to the CLIENT_2
and
the IUT receives a PUBREC message containing
header_flags indicating value '1111'B;
from the CLIENT_2
}
then {
the IUT closes the TCP_CONNECTION to the CLIENT_2
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBREC_002
Test Objective	Verify that the IUT sends PUBREC Control Packets in the order in which the corresponding QoS 2 PUBLISH Control Packets were received.
Reference	[MQTT-4.6.0-3], [MQTT-3.3.4-1], [MQTT-4.6.0-6], [MQTT-2.3.1-6]
PICS Selection	PICS_BROKER_QOS_2
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC,
packet_identifier corresponding to PACKET_ID_1;
;
and
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC,
packet_identifier corresponding to PACKET_ID_2;
;
}
then {	

the IUT sends a PUBREC message containing packet_identifier corresponding to PACKET_ID_1; and the IUT sends a PUBREC message containing packet_identifier corresponding to PACKET_ID_2; } }
Final Conditions

TP Id	TP_MQTT_BROKER_PUBREL_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in PUBREL Control Packet are invalid.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2], [MQTT-3.6.1-1]
PICS Selection	PICS_BROKER_QOS_2
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC;;
from the CLIENT_1
and
the IUT sends a PUBREC message
from the CLIENT_1
and
the IUT receives a PUBREL message containing
header_flags indicating value '1101'B;
from the CLIENT_1
}
then {
the IUT closes the TCP_CONNECTION to the CLIENT_1
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_PUBREL_002
Test Objective	Verify that the IUT sends PUBREL Control Packets in the order in which the corresponding PUBREC Control Packets were received.
Reference	[MQTT-4.6.0-4], [MQTT-4.6.0-6], [MQTT-2.3.1-6]
PICS Selection	PICS_BROKER_QOS_2
Initial Conditions	
with { the CLIENT_1 having a MQTT_CONNECTION to the IUT and the CLIENT_2 having a MQTT_CONNECTION to the IUT and the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing qos_level corresponding to EXACTLY_ONCE; and the CLIENT_1 delivered a PUBLISH message containing qos_level corresponding to EXACTLY_ONCE, topic_name corresponding to PX_PUBLISH_TOPIC, packet_identifier corresponding to PACKET_ID_1; to the IUT and the CLIENT_1 delivered a PUBLISH message containing qos_level corresponding to EXACTLY_ONCE, topic_name corresponding to PX_PUBLISH_TOPIC, packet_identifier corresponding to PACKET_ID_2;	

to the IUT }
Expected Behaviour

ensure that {
when {
the IUT sends a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC,
packet_identifier corresponding to PACKET_ID_1;;
to the CLIENT_2
and
the IUT receives a PUBREC message containing
packet_identifier corresponding to PACKET_ID_1;
from the CLIENT_2
and
the IUT sends a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC,
packet_identifier corresponding to PACKET_ID_2;;
to the CLIENT_2
and
the IUT receives a PUBREC message containing
packet_identifier corresponding to PACKET_ID_2;
from the CLIENT_2
}
then {
the IUT sends a PUBREL message containing
packet_identifier corresponding to PACKET_ID_1;
and
the IUT sends a PUBREL message containing
packet_identifier corresponding to PACKET_ID_2;
}
}

Final Conditions

TP Id	TP_MQTT_BROKER_PUBCOMP_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in PUBCOMP Control Packet are invalid.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2]
PICS Selection	PICS_BROKER_QOS_2
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
qos_level corresponding to EXACTLY_ONCE;
and
the CLIENT_1 delivered a PUBLISH message containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC;
to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT sends a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC;;
to the CLIENT_2
and
the IUT receives a PUBREC message	

```
    from the CLIENT_2
    and
    the IUT sends a PUBREL message
    to the CLIENT_2
    and
    the IUT receives a PUBCOMP message containing
    header_flags indicating value '1111'B;
    from the CLIENT_2
  }
  then {
      the IUT closes the TCP_CONNECTION to the CLIENT_2
  }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in SUBSCRIBE Control Packet are invalid.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2], [MQTT-3.8.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
      the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
  when {
      the IUT receives a SUBSCRIBE message containing
      header_flags indicating value '1101'B;
  }
  then {
      the IUT closes the TCP_CONNECTION
  }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_002
Test Objective	Verify that the IUT validates a SUBSCRIBE Control Packet to contain a non-zero 16-bit Packet Identifier.
Reference	[MQTT-2.3.1-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
      the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
  when {
      the IUT receives a SUBSCRIBE message containing
      header_flags indicating value '0010'B,
      packet_identifier corresponding to PACKET_ID_ZERO,
      payload containing
        topic_filter corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
        requested_qos corresponding to AT_LEAST_ONCE;
      ;
  }
  then {
      the IUT closes the TCP_CONNECTION
  }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_003
Test Objective	Verify that the IUT validates the topic_filter in a SUBSCRIBE Control Packet to be a well-formed UTF-8 encoded string and do not contain code points between U+D800 and U+DFFF.
Reference	[MQTT-1.5.3-1], [MQTT-3.8.3-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a SUBSCRIBE message containing
header_flags indicating value '0010'B,
packet_identifier corresponding to PACKET_ID,
payload containing
topic_filter corresponding to TOPIC_FILTER_D800,
requested_qos corresponding to AT_LEAST_ONCE;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_SUBSCRIBE_004
Test Objective	Verify that the IUT validates the topic_filter in a SUBSCRIBE Control Packet to be a well-formed UTF-8 encoded string and do not contain the null character (Unicode U+0000).
Reference	[MQTT-1.5.3-2], [MQTT-3.8.3-1], [MQTT-4.7.3-2], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a SUBSCRIBE message containing
header_flags indicating value '0010'B,
packet_identifier corresponding to PACKET_ID,
payload containing
topic_filter corresponding to TOPIC_FILTER_0000,
requested_qos corresponding to AT_LEAST_ONCE;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_SUBSCRIBE_005
Test Objective	Verify that the IUT validates the topic_filter in a SUBSCRIBE Control Packet to be at least on character long.
Reference	[MQTT-4.7.3-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	
ensure that {	

```
   when {
         the IUT receives a SUBSCRIBE message containing
         header_flags indicating value '0010'B,
         packet_identifier corresponding to PACKET_ID,
         payload containing
            topic_filter corresponding to TOPIC_FILTER_ZERO_CHARS,
            requested_qos corresponding to AT_LEAST_ONCE;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_006
Test Objective	Verify that the IUT validates a SUBSCRIBE Control Packet to contain at least one topic filter/QoS pair.
Reference	[MQTT-3.8.3-3], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
      the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
         the IUT receives a SUBSCRIBE message containing
         header_flags indicating value '0010'B,
         packet_identifier corresponding to PACKET_ID,
         payload containing
            omit;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_007
Test Objective	Verify that the IUT validates in a SUBSCRIBE Control Packet the upper 6 bits of a requested QoS byte (reserved bits) to be set to 0.
Reference	[MQTT-3.8.3-4], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
      the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
         the IUT receives a SUBSCRIBE message containing
         header_flags indicating value '0010'B,
         packet_identifier corresponding to PACKET_ID,
         payload containing
            topic_filter corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
            requested_qos corresponding to AT_MOST_ONCE,
            requested_qos_flags indicating value '111111'B;
         ;
   }
   then {
         the IUT closes the TCP_CONNECTION
```

```
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_008
Test Objective	Verify that the IUT validates the requested_qos field to be a valid QoS level.
Reference	[MQTT-3.8.3-4], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a SUBSCRIBE message containing
        header_flags indicating value '0010'B,
        packet_identifier corresponding to PACKET_ID,
        payload containing
           topic_filter corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
           requested_qos corresponding to INVALID_QOS;
        ;
   }
   then {
        the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_009
Test Objective	Verify that the IUT validates topic_filter field to be a valid multi-level Topic Fiilter.
Reference	[MQTT-4.7.1-2], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a SUBSCRIBE message containing
        header_flags indicating value '0010'B,
        packet_identifier corresponding to PACKET_ID,
        payload containing
           topic_filter corresponding to TOPIC_FILTER_MULTI_LEVEL_INVALID,
           requested_qos corresponding to AT_MOST_ONCE;
        ;
   }
   then {
        the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_010
Test Objective	Verify that the IUT validates topic_filter field to be a valid single-level Topic Fiilter.
Reference	[MQTT-4.7.1-3], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT having a MQTT_CONNECTION to the IUT
```

}
Expected Behaviour

ensure that {
  when {
  the IUT receives a SUBSCRIBE message containing
  header_flags indicating value '0010'B,
  packet_identifier corresponding to PACKET_ID,
  payload containing
  topic_filter corresponding to TOPIC_FILTER_SINGLE_LEVEL_INVALID,
  requested_qos corresponding to AT_MOST_ONCE;
  ;
  }
  then {
  the IUT closes the TCP_CONNECTION
  }
}

Final Conditions

TP Id	TP_MQTT_BROKER_SUBSCRIBE_011
Test Objective	Verify that the IUT allows topic_filter field to include the 'zero width no-break space character'
Reference	[MQTT-1.5.3-3]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
  the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
  when {
  the IUT receives a SUBSCRIBE message containing
  packet_identifier corresponding to PACKET_ID,
  payload containing
  topic_filter corresponding to TOPIC_FILTER_WITH_ZWNBS,
  requested_qos corresponding to AT_MOST_ONCE;
  ;
  }
  then {
  the IUT sends a SUBACK message containing
  packet_identifier corresponding to PACKET_ID;
  }
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_SUBACK_001
Test Objective	Verify that the IUT replies with a SUBACK Control Packet with valid header flags.
Reference	[MQTT-2.2.2-1], [MQTT-3.8.1-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
  the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
  when {
  the IUT receives a SUBSCRIBE message containing
  header_flags indicating value '0010'B;
  }
  then {
  the IUT sends a SUBACK message containing
  header_flags indicating value '0000'B;
  }
}	

Final Conditions

TP Id	TP_MQTT_BROKER_SUBACK_002
Test Objective	Verify that the IUT replies with a SUBACK Control Packet containing a packet identifier corresponding to the SUBSCRIBE Control Packet.
Reference	[MQTT-2.3.1-1], [MQTT-2.3.1-7], [MQTT-3.8.4-1], [MQTT-3.8.4-2]
PICS Selection	PICS_BROKER_QOS_1

Initial Conditions

```
with {
        the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a SUBSCRIBE message containing
        header_flags indicating value '0010'B,
        packet_identifier corresponding to PACKET_ID,
        payload containing
           topic_filter corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
           requested_qos corresponding to AT_LEAST_ONCE;
        ;
   }
   then {
        the IUT sends a SUBACK message containing
        header_flags indicating value '0000'B,
        packet_identifier corresponding to PACKET_ID;
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBACK_003
Test Objective	Verify that the IUT replies with a SUBACK Control Packet with a valid maximum QoS level.
Reference	[MQTT-3.9.3-1], [MQTT-3.9.3-2]
PICS Selection	PICS_BROKER_QOS_2

Initial Conditions

```
with {
        the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a SUBSCRIBE message containing
        header_flags indicating value '0010'B,
        packet_identifier corresponding to PACKET_ID,
        payload containing
           topic_filter corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
           requested_qos corresponding to AT_MOST_ONCE;
        ;
   }
   then {
        the IUT sends a SUBACK message containing
        header_flags indicating value '0000'B,
        packet_identifier corresponding to PACKET_ID,
        return_code indicating value 0x00;
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_SUBACK_004
Test Objective	Verify that the IUT replies with a SUBACK Control Packet with a valid maximum QoS level.
Reference	[MQTT-3.9.3-1], [MQTT-3.9.3-2]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_QOS_1
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a SUBSCRIBE message containing
header_flags indicating value '0010'B,
packet_identifier corresponding to PACKET_ID,
payload containing
topic_filter corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
requested_qos corresponding to AT_LEAST_ONCE;
;
}
then {
the IUT sends a SUBACK message containing
header_flags indicating value '0000'B,
packet_identifier corresponding to PACKET_ID,
return_code indicating value 0x01;
or
the IUT sends a SUBACK message containing // Note: if the IUT supports only QoS 0
header_flags indicating value '0000'B,
packet_identifier corresponding to PACKET_ID,
return_code indicating value 0x00;
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_SUBACK_005
Test Objective	Verify that the IUT replies with a SUBACK Control Packet with a valid maximum QoS level.
Reference	[MQTT-3.9.3-1], [MQTT-3.9.3-2]
PICS Selection	PICS_BROKER_BASIC and PICS_BROKER_QOS_1 and PICS_BROKER_QOS_2
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a SUBSCRIBE message containing
header_flags indicating value '0010'B,
packet_identifier corresponding to PACKET_ID,
payload containing
topic_filter corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
requested_qos corresponding to EXACTLY_ONCE;
;
}
then {
the IUT sends a SUBACK message containing
header_flags indicating value '0000'B,
packet_identifier corresponding to PACKET_ID,
return_code indicating value 0x02;
or
the IUT sends a SUBACK message containing // Note: if the IUT supports only up to QoS 1
header_flags indicating value '0000'B,
packet_identifier corresponding to PACKET_ID,
return_code indicating value 0x01;
or
the IUT sends a SUBACK message containing // Note: if the IUT supports only QoS 0
header_flags indicating value '0000'B,
packet_identifier corresponding to PACKET_ID,	

return_code indicating value 0x00; } }
Final Conditions

TP Id	TP_MQTT_BROKER_SUBACK_006
Test Objective	Verify that the IUT replies to a failed subscription with a SUBACK Control Packet with the return code 0x80.
Reference	[MQTT-3.9.3-1], [MQTT-3.9.3-2]
PICS Selection	PICS_BROKER_BASIC or PICS_BROKER_QOS_2
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a SUBSCRIBE message containing
header_flags indicating value '0010'B,
packet_identifier corresponding to PACKET_ID,
payload containing
topic_filter corresponding to TOPIC_FILTER_INVALID,
requested_qos corresponding to AT_MOST_ONCE;
;
}
then {
the IUT sends a SUBACK message containing
header_flags indicating value '0000'B,
packet_identifier corresponding to PACKET_ID,
return_code indicating value 0x80;
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_UNSUBSCRIBE_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in UNSUBSCRIBE Control Packet are invalid.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2], [MQTT-3.10.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a UNSUBSCRIBE message containing
header_flags indicating value '1101'B;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_UNSUBSCRIBE_002
Test Objective	Verify that the IUT validates a UNSUBSCRIBE Control Packet to contain a non-zero 16-bit Packet Identifier.
Reference	[MQTT-2.3.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a UNSUBSCRIBE message containing
header_flags indicating value '0010'B,
packet_identifier corresponding to PACKET_ID_ZERO;
}
then {
the IUT closes the TCP_CONNECTION
or
the IUT sends a UNSUBACK message containing
packet_identifier corresponding to PACKET_ID_ZERO;
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_UNSUBSCRIBE_003
Test Objective	Verify that the IUT validates the topic_filter in a UNSUBSCRIBE Control Packet to be a well-formed UTF-8 encoded string and do not contain code points between U+D800 and U+DFFF.
Reference	[MQTT-1.5.3-1], [MQTT-3.10.3-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a UNSUBSCRIBE message containing
header_flags indicating value '0010'B,
packet_identifier corresponding to PACKET_ID,
payload containing
topic_filter corresponding to TOPIC_FILTER_D800;
;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_UNSUBSCRIBE_004
Test Objective	Verify that the IUT validates all topic filters to be at least one character long.
Reference	[MQTT-4.7.3-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a UNSUBSCRIBE message containing
header_flags indicating value '0010'B,	

```
        packet_identifier corresponding to PACKET_ID,
        payload containing
            topic_filter corresponding to TOPIC_FILTER_ZERO_CHARS;
        ;
    }
    then {
        the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_UNSUBSCRIBE_005
Test Objective	Verify that the IUT validates the topic filter in a UNSUBSCRIBE Control Packet not to contain the null character (Unicode U+0000).
Reference	[MQTT-4.7.3-2], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a UNSUBSCRIBE message containing
        header_flags indicating value '0010'B,
        packet_identifier corresponding to PACKET_ID,
        payload containing
            topic_filter corresponding to TOPIC_FILTER_0000;
        ;
    }
    then {
        the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_UNSUBSCRIBE_006
Test Objective	Verify that the IUT validates a UNSUBSCRIBE Control Packet to contain at least on topic filter.
Reference	[MQTT-3.10.3-2], [MQTT-4.8.0-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a UNSUBSCRIBE message containing
        header_flags indicating value '0010'B,
        packet_identifier corresponding to PACKET_ID,
        payload containing
            omit;
        ;
    }
    then {
        the IUT closes the TCP_CONNECTION
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_UNSUBACK_001
Test Objective	Verify that the IUT replies with an UNSUBACK Control Packet with valid header flags.
Reference	[MQTT-2.2.2-1], [MQTT-3.10.1-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a UNSUBSCRIBE message containing
header_flags indicating value '0010'B;
}
then {
the IUT sends a UNSUBACK message containing
header_flags indicating value '0000'B;
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_UNSUBACK_002
Test Objective	Verify that the IUT replies with a UNSUBACK Control Packet containing a packet identifier corresponding to the UNSUBSCRIBE Control Packet.
Reference	[MQTT-3.10.4-4], [MQTT-2.3.1-7]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
and
the CLIENT subscribed the PX_PUBLISH_TOPIC containing
qos_level corresponding to AT_MOST_ONCE;
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a UNSUBSCRIBE message containing
header_flags indicating value '0010'B,
packet_identifier corresponding to PACKET_ID,
payload containing
topic_filter corresponding to PX_PUBLISH_TOPIC;
;
}
then {
the IUT sends a UNSUBACK message containing
header_flags indicating value '0000'B,
packet_identifier corresponding to PACKET_ID;
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_UNSUBACK_003
Test Objective	Verify that the IUT replies with an UNSUBACK Control Packet even if no topic subscriptions are deleted.
Reference	[MQTT-3.10.4-5]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
and
the CLIENT not subscribed the PX_PUBLISH_TOPIC
}	

Expected Behaviour	

```
ensure that {
   when {
         the IUT receives a UNSUBSCRIBE message containing
         header_flags indicating value '0010'B,
         packet_identifier corresponding to PACKET_ID,
         payload containing
           topic_filter corresponding to PX_PUBLISH_TOPIC;
         ;
   }
   then {
         the IUT sends a UNSUBACK message containing
         header_flags indicating value '0000'B;
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_UNSUBACK_004
Test Objective	Verify that the IUT replies to UNSUBSCRIBE Control Packets with multiple topic filters with one single UNSUBACK Control Packet.
Reference	[MQTT-3.10.4-6]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
         the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
         the IUT receives a UNSUBSCRIBE message containing
         header_flags indicating value '0010'B,
         packet_identifier corresponding to PACKET_ID,
         payload containing
           topic_filter corresponding to PX_PUBLISH_TOPIC,
           topic_filter corresponding to TOPIC_FILTER_VALID; // second topic filter
         ;
   }
   then {
         the IUT sends a UNSUBACK message
         and
         the IUT sends no second UNSUBACK message
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_PINGREQ_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in PINGREQ Control Packet are invalid.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
         the CLIENT having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
         the IUT receives a PINGREQ message containing
         header_flags indicating value '1111'B;
   }
   then {
         the IUT closes the TCP_CONNECTION
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_PINGRESP_001
Test Objective	Verify that the IUT replies with a PINGRESP Control Packet with valid header flags.
Reference	[MQTT-2.2.2-1], [MQTT-3.12.4-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PINGREQ message containing
header_flags indicating value '0000'B;
}
then {
the IUT sends a PINGRESP message containing
header_flags indicating value '0000'B;
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_DISCONNECT_001
Test Objective	Verify that the IUT closes the network connection if fixed header flags in DISCONNECT Control Packet are valid.
Reference	[MQTT-2.2.2-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a DISCONNECT message containing
header_flags indicating value '0000'B;
}
then {
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_DISCONNECT_002
Test Objective	Verify that the IUT closes the network connection if fixed header flags in DISCONNECT Control Packet are valid.
Reference	[MQTT-2.2.2-2], [MQTT-3.14.1-1], [MQTT-4.8.0-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a DISCONNECT message containing
header_flags indicating value '1111'B;
}
then {
the IUT closes the TCP_CONNECTION	

```
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_REMLEN_001
Test Objective	Verify that the IUT forwards PUBLISH Control Packets with Remaining Length fields encoded in one byte.
Reference	MQTT 2.2.3
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
       and
       the CLIENT_2 having a MQTT_CONNECTION to the IUT
       and
       the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
          qos_level corresponding to AT_MOST_ONCE;
       to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a PUBLISH message containing
        publish_header containing
           dup_flag indicating value '0'B,
           qos_level corresponding to AT_MOST_ONCE,
        topic_name corresponding to PX_PUBLISH_TOPIC,
        packet_identifier indicating value omit,
        payload corresponding to PAYLOAD;
        ;
        from the CLIENT_1
   }
   then {
        the IUT sends a PUBLISH message containing
        publish_header containing
           dup_flag indicating value '0'B,
           qos_level corresponding to AT_MOST_ONCE,
        topic_name corresponding to PX_PUBLISH_TOPIC,
        packet_identifier indicating value omit,
        payload corresponding to PAYLOAD;
        ;
        to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_REMLEN_002
Test Objective	Verify that the IUT forwards PUBLISH Control Packets with Remaining Length fields encoded in two bytes.
Reference	MQTT 2.2.3
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
       and
       the CLIENT_2 having a MQTT_CONNECTION to the IUT
       and
       the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
          qos_level corresponding to AT_MOST_ONCE;
       to the IUT
}
```

Expected Behaviour

```
ensure that {
```

```
   when {
         the IUT receives a PUBLISH message containing
         publish_header containing
            dup_flag indicating value '0'B,
            qos_level corresponding to AT_MOST_ONCE,
         topic_name corresponding to PX_PUBLISH_TOPIC,
         packet_identifier indicating value omit,
         payload corresponding to PAYLOAD_REM_LEN_2;
         ;
         from the CLIENT_1
   }
   then {
         the IUT sends a PUBLISH message containing
         publish_header containing
            dup_flag indicating value '0'B,
            qos_level corresponding to AT_MOST_ONCE,
         topic_name corresponding to PX_PUBLISH_TOPIC,
         packet_identifier indicating value omit,
         payload corresponding to PAYLOAD_REM_LEN_2;
         ;
         to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_REMLEN_003
Test Objective	Verify that the IUT forwards PUBLISH Control Packets with Remaining Length fields encoded in three bytes.
Reference	MQTT 2.2.3
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
       and
       the CLIENT_2 having a MQTT_CONNECTION to the IUT
       and
       the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
          qos_level corresponding to AT_MOST_ONCE;
       to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
         the IUT receives a PUBLISH message containing
         publish_header containing
            dup_flag indicating value '0'B,
            qos_level corresponding to AT_MOST_ONCE,
         topic_name corresponding to PX_PUBLISH_TOPIC,
         packet_identifier indicating value omit,
         payload corresponding to PAYLOAD_REM_LEN_3;
         ;
         from the CLIENT_1
   }
   then {
         the IUT sends a PUBLISH message containing
         publish_header containing
            dup_flag indicating value '0'B,
            qos_level corresponding to AT_MOST_ONCE,
         topic_name corresponding to PX_PUBLISH_TOPIC,
         packet_identifier indicating value omit,
         payload corresponding to PAYLOAD_REM_LEN_3;
         ;
         to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_REMLEN_004
Test Objective	Verify that the IUT forwards PUBLISH Control Packets with Remaining Length fields encoded in four bytes.
Reference	MQTT 2.2.3
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
        and
        the CLIENT_2 having a MQTT_CONNECTION to the IUT
        and
        the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
          qos_level corresponding to AT_MOST_ONCE;
        to the IUT
}
```

Expected Behaviour

```
ensure that {
  when {
        the IUT receives a PUBLISH message containing
        publish_header containing
          dup_flag indicating value '0'B,
          qos_level corresponding to AT_MOST_ONCE,
        topic_name corresponding to PX_PUBLISH_TOPIC,
        packet_identifier indicating value omit,
        payload corresponding to PAYLOAD_REM_LEN_4;
        ;
        from the CLIENT_1
  }
  then {
        the IUT sends a PUBLISH message containing
        publish_header containing
          dup_flag indicating value '0'B,
          qos_level corresponding to AT_MOST_ONCE,
        topic_name corresponding to PX_PUBLISH_TOPIC,
        packet_identifier indicating value omit,
        payload corresponding to PAYLOAD_REM_LEN_4;
        ;
        to the CLIENT_2
  }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_KEEPALIVE_001
Test Objective	Verify that the IUT disconnects a client if it does not receive a Control Packet from it within one and a half times of the given Keep Alive time period.
Reference	[MQTT-3.1.2-24]
PICS Selection	PICS_CLIENT_BASIC

Initial Conditions

```
with {
        the CLIENT having a MQTT_CONNECTION to the IUT
      and
      the CLIENT established the MQTT_CONNECTION containing
      keep_alive corresponding to PX_KEEP_ALIVE;
}
```

Expected Behaviour

```
ensure that {
  when {
        the IUT times_out
  }
  then {
        the IUT closes the TCP_CONNECTION to the CLIENT
```

```
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_RTND_001
Test Objective	Verify that the IUT does not delete Retained Messages when a session with the corresponding client ends.
Reference	[MQTT-3.1.2-7]
PICS Selection	PICS_BROKER_RTND

Initial Conditions

```
with {
        the IUT having a UTF8_MESSAGE_VALID in the RETAIN_TOPIC
      and
      the CLIENT having a MQTT_CONNECTION to the IUT
      and
      the CLIENT having a CLEAN_SESSION
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a SUBSCRIBE message containing
        packet_identifier corresponding to PACKET_ID,
        payload containing
           topic_filter corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
           requested_qos corresponding to AT_MOST_ONCE;
        ;
   }
   then {
        the IUT sends a SUBACK message containing
        return_code indicating value 0x00;
        and the IUT sends a PUBLISH message containing
        topic_name corresponding to PX_SUBSCRIBE_TOPIC_FILTER,
        payload corresponding to UTF8_MESSAGE_VALID;
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_RTND_002
Test Objective	Verify that the IUT stores Retained Messages for future deliveries.
Reference	[MQTT-3.3.1-5], [MQTT-3.3.1-6], [MQTT-3.3.1-8]
PICS Selection	PICS_BROKER_RTND

Initial Conditions

```
with {
        the CLIENT_1 published a Message containing
          dup_flag indicating value '0'B,
          qos_level corresponding to AT_LEAST_ONCE,
          retain_flag indicating value '1'B,
          topic_name corresponding to PX_PUBLISH_TOPIC,
        packet_identifier corresponding to PACKET_ID_1,
        payload corresponding to PAYLOAD;
       to the IUT
       and
       the CLIENT_2 having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a SUBSCRIBE message containing
        packet_identifier corresponding to PACKET_ID_2,
        payload containing
           topic_filter corresponding to PX_PUBLISH_TOPIC,
           requested_qos corresponding to AT_MOST_ONCE;
        ;
```

```
        from the CLIENT_2
    }
    then {
        the IUT sends a SUBACK message containing
        packet_identifier corresponding to PACKET_ID_2,
        payload containing
            return_code indicating value 0x00;
        ;
        to the CLIENT_2
        and
        the IUT sends a PUBLISH message containing
        publish_header containing
            dup_flag indicating value '0'B,
            qos_level corresponding to AT_MOST_ONCE,
            retain_flag indicating value '1'B,
        topic_name corresponding to PX_PUBLISH_TOPIC,
        packet_identifier corresponding to PACKET_ID_3,
        payload corresponding to PAYLOAD;
        ;
        to the CLIENT_2
    }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_RTND_003
Test Objective	Verify that the IUT sets the retain_flag to 0 when Retained Messages are delivered directly to existing subscriptions.
Reference	[MQTT-3.3.1-9]
PICS Selection	PICS_BROKER_RTND

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
      and
      the CLIENT_2 having a MQTT_CONNECTION to the IUT
      and
      the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
        qos_level corresponding to AT_MOST_ONCE;
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a PUBLISH message containing
        publish_header containing
            dup_flag indicating value '0'B,
            qos_level corresponding to AT_LEAST_ONCE,
            retain_flag indicating value '1'B,
        topic_name corresponding to PX_PUBLISH_TOPIC,
        packet_identifier corresponding to PACKET_ID_1,
        payload corresponding to PAYLOAD;
        ;
        from the CLIENT_1
    }
    then {
        the IUT sends a PUBACK message containing
        packet_identifier corresponding to PACKET_ID_1;
        to the CLIENT_1
        and
        the IUT sends a PUBLISH message containing
        publish_header containing
            dup_flag indicating value '0'B,
            qos_level corresponding to AT_MOST_ONCE,
            retain_flag indicating value '0'B,
        topic_name corresponding to PX_PUBLISH_TOPIC,
        packet_identifier corresponding to PACKET_ID_2,
        payload corresponding to PAYLOAD;
        ;
```

to the CLIENT_2 } }
Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_RTND_004
Test Objective	Verify that the IUT forwards Retained Messages with a zero-bytes payload.
Reference	[MQTT-3.3.1-10]
PICS Selection	PICS_BROKER_RTND
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
qos_level corresponding to AT_MOST_ONCE;
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
publish_header containing
qos_level corresponding to AT_MOST_ONCE,
retain_flag indicating value '1'B,
topic_name corresponding to PX_PUBLISH_TOPIC,
payload corresponding to PAYLOAD_ZERO_BYTE;
;
from the CLIENT_1
}
then {
the IUT sends a PUBLISH message containing
publish_header containing
qos_level corresponding to AT_MOST_ONCE,
retain_flag indicating value '0'B,
payload corresponding to PAYLOAD_ZERO_BYTE;
;
to the CLIENT_2
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_FEAT_RTND_005
Test Objective	Verify that the IUT deletes a stored Retained Messages if it receives a new Retained Message with a zero-bytes payload.
Reference	[MQTT-3.3.1-10], [MQTT-3.3.1-11]
PICS Selection	PICS_BROKER_RTND
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
publish_header containing
qos_level corresponding to AT_MOST_ONCE,
retain_flag indicating value '1'B,
topic_name corresponding to PX_PUBLISH_TOPIC,
payload corresponding to PAYLOAD;
;	

```
      from the CLIENT_1
      and
      the IUT receives a PUBLISH message containing
      publish_header containing
        qos_level corresponding to AT_MOST_ONCE,
        retain_flag indicating value '1'B,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      payload corresponding to PAYLOAD_ZERO_BYTE;
      ;
      from the CLIENT_1
      and
      the IUT receives a SUBSCRIBE message containing
      payload containing
        topic_filter corresponding to PX_PUBLISH_TOPIC,
        requested_qos corresponding to AT_MOST_ONCE;
      ;
   }
   then {
         the IUT sends no PUBLISH message to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_RTND_006
Test Objective	Verify that the IUT does neither store a Retained Message nor removes or replaces any existing Retained Messages if the retained_flag is set to 0.
Reference	[MQTT-3.3.1-12]
PICS Selection	PICS_BROKER_RTND

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
     and
     the CLIENT_2 having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
         the IUT receives a PUBLISH message containing
      publish_header containing
        qos_level corresponding to AT_MOST_ONCE,
        retain_flag indicating value '1'B,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      payload corresponding to PAYLOAD;
      ;
      from the CLIENT_1
      and
      the IUT receives a PUBLISH message containing
      publish_header containing
        qos_level corresponding to AT_MOST_ONCE,
        retain_flag indicating value '0'B,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      payload corresponding to PAYLOAD_2;
      ;
      from the CLIENT_1
      and
      the IUT receives a SUBSCRIBE message containing
      payload containing
        topic_filter corresponding to PX_PUBLISH_TOPIC,
        requested_qos corresponding to AT_MOST_ONCE;
      ;
      from the CLIENT_2
   }
   then {
         the IUT sends a PUBLISH message containing
      publish_header containing
        qos_level corresponding to AT_MOST_ONCE,
```

```
        retain_flag indicating value '1'B,
      payload corresponding to PAYLOAD;
      ;
      to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_RTND_007
Test Objective	Verify that the IUT stores Retained Messages with a QoS level of 0 for future deliveries. However, the IUT may choose to discard Retained Messages with a QoS level of 0 at any time.
Reference	[MQTT-3.3.1-7]
PICS Selection	PICS_BROKER_RTND

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_2 having a MQTT_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a PUBLISH message containing
      publish_header containing
        qos_level corresponding to AT_LEAST_ONCE,
        retain_flag indicating value '1'B,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      payload corresponding to PAYLOAD;
      ;
      from the CLIENT_1
      and
      the IUT receives a PUBLISH message containing
      publish_header containing
        qos_level corresponding to AT_MOST_ONCE,
        retain_flag indicating value '1'B,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      payload corresponding to PAYLOAD_2;
      ;
      from the CLIENT_1
      and
      the IUT receives a SUBSCRIBE message containing
      payload containing
        topic_filter corresponding to PX_PUBLISH_TOPIC,
        requested_qos corresponding to AT_MOST_ONCE;
      ;
      from the CLIENT_2
   }
   then {
        the IUT sends a PUBLISH message containing
      publish_header containing
        qos_level corresponding to AT_MOST_ONCE,
        retain_flag indicating value '1'B,
      payload corresponding to PAYLOAD_2;
      ;
      to the CLIENT_2
      or
      the IUT sends no PUBLISH message to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_LWT_001
Test Objective	Verify that the IUT sends a Will Messages to subscribes if a client with LWT disconnects unexpectedly.
Reference	[MQTT-3.1.2-8]
PICS Selection	PICS_BROKER_LWT
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_1 established the MQTT_CONNECTION containing
flags containing
will_flag indicating value '1'B,
payload containing
will_topic corresponding to PX_PUBLISH_TOPIC,
will_message corresponding to PX_WILL_MESSAGE;
;;
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 subscribed the PX_PUBLISH_TOPIC
}	

Expected Behaviour	

ensure that {
when {
the CLIENT_1 closes the TCP_CONNECTION to the IUT
}
then {
the IUT sends a PUBLISH message containing
topic_name corresponding to PX_PUBLISH_TOPIC,
payload corresponding to PX_WILL_MESSAGE;
to the CLIENT_2
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_FEAT_LWT_002
Test Objective	Verify that the IUT deletes a Will Messages if the client with LWT disconnects correctly with a DISCONNECT Control Packet.
Reference	[MQTT-3.1.2-8], [MQTT-3.1.2-10], [MQTT-3.14.4-3]
PICS Selection	PICS_BROKER_LWT
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_1 established the MQTT_CONNECTION containing
flags containing
will_flag indicating value '1'B,
payload containing
will_topic corresponding to PX_PUBLISH_TOPIC,
will_message corresponding to PX_WILL_MESSAGE;
;;
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 subscribed the PX_PUBLISH_TOPIC
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a DISCONNECT message from the CLIENT_1
}
then {
the IUT sends no PUBLISH message to the CLIENT_2
}
}	

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_LWT_003
Test Objective	Verify that the IUT sends no Will Message if a client without LWT disconnects unexpectedly.
Reference	[MQTT-3.1.2-12]
PICS Selection	PICS_BROKER_LWT

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_1 established the MQTT_CONNECTION containing
    flags containing
      will_flag indicating value '0'B;
    ;
    and
    the CLIENT_2 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_2 subscribed the PX_PUBLISH_TOPIC
}
```

Expected Behaviour

```
ensure that {
  when {
        the CLIENT_1 closes the TCP_CONNECTION to the IUT
  }
  then {
        the IUT sends no PUBLISH message to the CLIENT_2
  }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_LWT_004
Test Objective	Verify that the IUT handles Will Messages with will_retain set to 1 as Retained Will Messages.
Reference	[MQTT-3.1.2-17]
PICS Selection	PICS_BROKER_LWT

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_1 established the MQTT_CONNECTION containing
    flags containing
      will_retain indicating value '1'B,
      will_flag indicating value '1'B,
    payload containing
      will_topic corresponding to PX_PUBLISH_TOPIC,
      will_message corresponding to PX_WILL_MESSAGE;
    ;;
    and
    the CLIENT_2 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_1 closed the TCP_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
  when {
        the IUT receives a SUBSCRIBE message containing
      payload containing
        topic_filter corresponding to PX_PUBLISH_TOPIC;
      ;
      from the CLIENT_2
  }
  then {
        the IUT sends a PUBLISH message containing
      topic_name corresponding to PX_PUBLISH_TOPIC,
```

```
      payload corresponding to PX_WILL_MESSAGE;
      to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_LWT_005
Test Objective	Verify that the IUT handles Will Messages with will_retain set to 1 as non-retained Will Messages.
Reference	[MQTT-3.1.2-16]
PICS Selection	PICS_BROKER_LWT

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_1 established the MQTT_CONNECTION containing
    flags containing
      will_retain indicating value '0'B,
      will_flag indicating value '1'B,
    payload containing
      will_topic corresponding to PX_PUBLISH_TOPIC,
      will_message corresponding to PX_WILL_MESSAGE;
    ;;
    and
    the CLIENT_2 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_1 closed the TCP_CONNECTION to the IUT
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a SUBSCRIBE message containing
      payload containing
        topic_filter corresponding to PX_PUBLISH_TOPIC;
      ;
      from the CLIENT_2
   }
   then {
        the IUT sends no PUBLISH message to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_PUBSUB_001
Test Objective	Verify that the IUT validates the UTF-8 encoded sequence 0xEF 0xBB 0xBF as Unicode U+FEFF ('ZERO WIDTH NO-BREAK SPACE') within the topic name of a PUBLISH Control Packet.
Reference	[MQTT-1.5.3-3], [MQTT-4.7.3-4]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_2 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_2 subscribed the PX_PUBLISH_TOPIC
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a PUBLISH message containing
    publish_header containing
      qos_level corresponding to AT_MOST_ONCE,
    topic_name corresponding to TOPIC_FILTER_WITH_ZWNBS;;
    from the CLIENT_1
```

```
   }
   then {
        the IUT sends no PUBLISH to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_PUBSUB_002
Test Objective	Verify that the IUT validates the UTF-8 encoded sequence 0xEF 0xBB 0xBF as Unicode U+FEFF ('ZERO WIDTH NO-BREAK SPACE') within the topic filter of a SUBSCRIBE Control Packet.
Reference	[MQTT-1.5.3-3], [MQTT-4.7.3-4]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
     and
     the CLIENT_2 having a MQTT_CONNECTION to the IUT
     and
     the CLIENT_2 subscribed the TOPIC_FILTER_WITH_ZWNBS
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a SUBSCRIBE message containing
     payload containing
        topic_filter corresponding to PX_PUBLISH_TOPIC,
        requested_qos corresponding to AT_MOST_ONCE;
     ;
     from the CLIENT_1
   }
   then {
        the IUT sends no PUBLISH to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_PUBSUB_003
Test Objective	Verify that the IUT does not match topic filters starting with a multi-level wildcard character (#) with topic names beginning with a $ character
Reference	[MQTT-4.7.2-1]
PICS Selection	PICS_BROKER_BASIC

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
     and
     the CLIENT_2 having a MQTT_CONNECTION to the IUT
     and
     the CLIENT_2 subscribed the TOPIC_FILTER_MULTI_LEVEL_ALL
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a PUBLISH message containing
      topic_name corresponding to TOPIC_NAME_SYS;
      from the CLIENT_1
   }
   then {
        the IUT sends no PUBLISH message to the CLIENT_2
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_PUBSUB_004
Test Objective	Verify that the IUT does not match topic filters starting with a single-level wildcard character (+) with topic names beginning with a $ character
Reference	[MQTT-4.7.2-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 subscribed the TOPIC_FILTER_SINGLE_LEVEL
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_SYS;
from the CLIENT_1
}
then {
the IUT sends no PUBLISH message to the CLIENT_2
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_FEAT_PUBSUB_005
Test Objective	Verify that the IUT does match topic names and filters beginning with a $ character.
Reference	[MQTT-4.7.2-1]
PICS Selection	PICS_BROKER_BASIC
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 subscribed the TOPIC_FILTER_MULTI_LEVEL_SYS_ALL
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_SYS;
from the CLIENT_1
}
then {
the IUT sends a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_SYS;
to the CLIENT_2
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_FEAT_PUBSUB_006
Test Objective	Verify that the IUT resends PUBLISH Control Packets in the order in which the original PUBLISH Control Packets were sent.
Reference	[MQTT-4.6.0-1], [MQTT-4.6.0-6], [MQTT-4.4.0-1]
PICS Selection	PICS_BROKER_QOS_1
Initial Conditions	
with { the CLIENT_1 having a MQTT_CONNECTION to the IUT and the CLIENT_2 having a MQTT_CONNECTION to the IUT	

```
    and
    the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
      requested_qos corresponding to AT_LEAST_ONCE;
    to the IUT
    and
    the CLIENT_1 sent a PUBLISH message containing
      publish_header containing
          qos_level corresponding to AT_LEAST_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_1;
    ;
    to the IUT
    and
    the CLIENT_1 sent a PUBLISH message containing
      publish_header containing
          qos_level corresponding to AT_LEAST_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_2;
    ;
    to the IUT
    and
    the IUT sent a PUBLISH message containing
      publish_header containing
          qos_level corresponding to AT_LEAST_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_1;
    ;
    to the CLIENT_2
    and
    the IUT sent a PUBLISH message containing
      publish_header containing
          qos_level corresponding to AT_LEAST_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_2;
    ;
    to the CLIENT_2
}
```

Expected Behaviour

```
ensure that {
  when {
        the IUT received no PUBACK message containing
      packet_identifier corresponding to PACKET_ID_1;
    and
    the IUT received no PUBACK message containing
      packet_identifier corresponding to PACKET_ID_2;
  }
  then {
        the IUT sends a PUBLISH message containing
      publish_header containing
          qos_level corresponding to AT_LEAST_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_1;
    ;
    and
    the IUT sends a PUBLISH message containing
      publish_header containing
          qos_level corresponding to AT_LEAST_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_2;
    ;
  }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_PUBSUB_007
Test Objective	Verify that the IUT resends PUBLISH Control Packets in the order in which the original PUBLISH Control Packets were sent.
Reference	[MQTT-4.6.0-1], [MQTT-4.6.0-6], [MQTT-4.4.0-1]
PICS Selection	PICS_BROKER_QOS_2

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_2 having a MQTT_CONNECTION to the IUT
    and
    the CLIENT_2 subscribed the PX_PUBLISH_TOPIC containing
      requested_qos corresponding to EXACTLY_ONCE;
    to the IUT
    and
    the CLIENT_1 sent a PUBLISH message containing
      publish_header containing
        qos_level corresponding to EXACTLY_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_1;
    ;
    to the IUT
    and
    the CLIENT_1 sent a PUBLISH message containing
      publish_header containing
        qos_level corresponding to EXACTLY_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_2;
    ;
    to the IUT
    and
    the IUT sent a PUBLISH message containing
      publish_header containing
        qos_level corresponding to EXACTLY_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_1;
    ;
    to the CLIENT_2
    and
    the IUT sent a PUBLISH message containing
      publish_header containing
        qos_level corresponding to EXACTLY_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_2;
    ;
    to the CLIENT_2
}
```

Expected Behaviour

```
ensure that {
  when {
        the IUT not received a PUBREC message containing
      packet_identifier corresponding to PACKET_ID_1;
    and
    the IUT not received a PUBREC message containing
      packet_identifier corresponding to PACKET_ID_2;
  }
  then {
        the IUT sends a PUBLISH message containing
      publish_header containing
          qos_level corresponding to EXACTLY_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
      packet_identifier corresponding to PACKET_ID_1;
    ;
    and
    the IUT sends a PUBLISH message containing
      publish_header containing
          qos_level corresponding to EXACTLY_ONCE,
      topic_name corresponding to PX_PUBLISH_TOPIC,
```

packet_identifier corresponding to PACKET_ID_2; ; } }
Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_QOS_001
Test Objective	Verify that the IUT delivers PUBLISH Control Packets (in case of overlapping topic filter) respecting the maximum QoS level of all matching subscriptions.
Reference	[MQTT-3.3.5-1]
PICS Selection	PICS_BROKER_QOS_1
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_1 subscribed the TOPIC_FILTER_VALID containing
qos_level corresponding to AT_MOST_ONCE;
and
the CLIENT_1 subscribed the TOPIC_FILTER_VALID_OVERLAP containing
qos_level corresponding to AT_LEAST_ONCE;
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to AT_MOST_ONCE,
topic_name corresponding to TOPIC_NAME_VALID_OVERLAP;
;
from the CLIENT_2
}
then {
the IUT sends a PUBLISH message containing
header containing
qos_level corresponding to AT_LEAST_ONCE;
;
to the CLIENT_1
or
the IUT sends a PUBLISH message containing
header containing
qos_level corresponding to AT_MOST_ONCE;
;
to the CLIENT_1
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_FEAT_QOS_002
Test Objective	Verify that the IUT delivers PUBLISH Control Packets (in case of overlapping topic filter) respecting the maximum QoS level of all matching subscriptions.
Reference	[MQTT-3.3.5-1]
PICS Selection	PICS_BROKER_QOS_2
Initial Conditions	
with { the CLIENT_1 having a MQTT_CONNECTION to the IUT and the CLIENT_2 having a MQTT_CONNECTION to the IUT and the CLIENT_1 subscribed the TOPIC_FILTER_VALID containing qos_level corresponding to AT_MOST_ONCE; and the CLIENT_1 subscribed the TOPIC_FILTER_VALID_OVERLAP containing	

```
        qos_level corresponding to EXACTLY_ONCE;
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a PUBLISH message containing
      header containing
        qos_level corresponding to AT_MOST_ONCE,
      topic_name corresponding to TOPIC_NAME_VALID_OVERLAP;
      ;
      from the CLIENT_2
   }
   then {
        the IUT sends a PUBLISH message containing
      header containing
        qos_level corresponding to EXACTLY_ONCE;
      ;
      to the CLIENT_1
      or
      the IUT sends a PUBLISH message containing
      header containing
        qos_level corresponding to AT_MOST_ONCE;
      ;
      to the CLIENT_1
   }
}
```

Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_QOS_003
Test Objective	Verify that the IUT delivers PUBLISH Control Packets (in case of overlapping topic filter) respecting the maximum QoS level of all matching subscriptions.
Reference	[MQTT-3.3.5-1]
PICS Selection	PICS_BROKER_QOS_2

Initial Conditions

```
with {
        the CLIENT_1 having a MQTT_CONNECTION to the IUT
      and
      the CLIENT_2 having a MQTT_CONNECTION to the IUT
      and
      the CLIENT_1 subscribed the TOPIC_FILTER_VALID containing
        qos_level corresponding to AT_LEAST_ONCE;
      and
      the CLIENT_1 subscribed the TOPIC_FILTER_VALID_OVERLAP containing
        qos_level corresponding to EXACTLY_ONCE;
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a PUBLISH message containing
      header containing
        qos_level corresponding to AT_LEAST_ONCE,
      topic_name corresponding to TOPIC_NAME_VALID_OVERLAP;
      ;
      from the CLIENT_2
   }
   then {
        the IUT sends a PUBLISH message containing
      header containing
        qos_level corresponding to EXACTLY_ONCE;
      ;
      to the CLIENT_1
      or
      the IUT sends a PUBLISH message containing
      header containing
        qos_level corresponding to AT_LEAST_ONCE;
      ;
```

to the CLIENT_1 } }
Final Conditions

TP Id	TP_MQTT_BROKER_FEAT_QOS_004
Test Objective	Verify that the IUT delivers PUBLISH Control Packets (in case of overlapping topic filter) respecting the maximum QoS level of all matching subscriptions.
Reference	[MQTT-3.3.5-1]
PICS Selection	PICS_BROKER_QOS_1
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_1 subscribed the TOPIC_FILTER_VALID containing
qos_level corresponding to AT_LEAST_ONCE;
and
the CLIENT_1 subscribed the TOPIC_FILTER_VALID_OVERLAP containing
qos_level corresponding to AT_MOST_ONCE;
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to AT_LEAST_ONCE,
topic_name corresponding to TOPIC_NAME_VALID_OVERLAP;
;
from the CLIENT_2
}
then {
the IUT sends a PUBLISH message containing
header containing
qos_level corresponding to AT_MOST_ONCE;
;
to the CLIENT_1
}
}	

Final Conditions	

TP Id	TP_MQTT_BROKER_FEAT_QOS_005
Test Objective	Verify that the IUT delivers PUBLISH Control Packets (in case of overlapping topic filter) respecting the maximum QoS level of all matching subscriptions.
Reference	[MQTT-3.3.5-1]
PICS Selection	PICS_BROKER_QOS_2
Initial Conditions	

with {
the CLIENT_1 having a MQTT_CONNECTION to the IUT
and
the CLIENT_2 having a MQTT_CONNECTION to the IUT
and
the CLIENT_1 subscribed the TOPIC_FILTER_VALID containing
qos_level corresponding to EXACTLY_ONCE;
and
the CLIENT_1 subscribed the TOPIC_FILTER_VALID_OVERLAP containing
qos_level corresponding to AT_LEAST_ONCE;
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing	

```
      header containing
        qos_level corresponding to EXACTLY_ONCE,
      topic_name corresponding to TOPIC_NAME_VALID_OVERLAP;
      ;
      from the CLIENT_2
   }
   then {
         the IUT sends a PUBLISH message containing
      header containing
        qos_level corresponding to AT_LEAST_ONCE;
      ;
      to the CLIENT_1
   }
}
```

Final Conditions

6 Test Purposes for MQTT Client

TP Id	TP_MQTT_CLIENT_CONNECT_001
Test Objective	Verify that the IUT is able to send CONNECT Control Packets with valid Header Flags.
Reference	[MQTT-2.2.2-1]
PICS Selection	PICS_CLIENT_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
   when {
         the IUT is triggered to send a CONNECT message
   }
   then {
         the IUT sends a CONNECT message containing
            header_flags indicating value '0000'B;
   }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_002
Test Objective	The protocol name representing the protocol is a UTF-8 encoded 'MQTT' string. Verify that the IUT is able to send CONNECT Control Packets with valid protocol name represented by a UTF-8 encoded 'MQTT' string.
Reference	[MQTT-3.1.2-1]
PICS Selection	PICS_CLIENT_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
   when {
         the IUT is triggered to send a CONNECT message
   }
   then {
         the IUT sends a CONNECT message containing
            protocol_name corresponding to PROTOCOL_NAME;
   }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_003
Test Objective	Verify that the IUT is able to send CONNECT Control Packets with Protocol Level 0x04 for MQTT 3.1.1
Reference	[MQTT-3.1.2-2]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a CONNECT message
}
then {
the IUT sends a CONNECT message containing
protocol_level indicating value 0x04;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_CONNECT_004
Test Objective	Verify that the IUT is able to send CONNECT Control Packets with valid reserved flag.
Reference	[MQTT-3.1.2-3]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a CONNECT message
}
then {
the IUT sends a CONNECT message containing
connect_flags containing
reserved_field indicating value '0'B;
;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_CONNECT_005
Test Objective	Verify that the IUT is able to send CONNECT Control Packets with valid Last Will Testament settings.
Reference	[MQTT-3.1.2-9], [MQTT-3.1.2-14]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a CONNECT message containing
connect_flags containing
will_flag indicating value '1'B;
;
}
then {
the IUT sends a CONNECT message containing
connect_flags containing
will_flag indicating value '1'B,
will_qos corresponding to VALID_QOS;
,
payload containing
will_topic indicating value not omit,	

```
            will_message indicating value not omit;
          ;
    }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_006
Test Objective	Verify that the IUT is able to send CONNECT Control Packets without Last Will Testament.
Reference	[MQTT-3.1.2-11], [MQTT-3.1.2-13], [MQTT-3.1.2-15]
PICS Selection	PICS_CLIENT_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
    when {
        the IUT is triggered to send a CONNECT message containing
          connect_flags containing
            will_flag indicating value '0'B;
          ;
    }
    then {
        the IUT sends a CONNECT message containing
          connect_flags containing
            will_flag indicating value '0'B,
            will_qos corresponding to AT_MOST_ONCE,
            will_retain indicating value '0'B;
            ,
          payload containing
            will_topic indicating value omit,
            will_message indicating value omit;
          ;
    }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_007
Test Objective	Verify that the IUT is able to send CONNECT Control Packets with valid settings for a connection without authentication.
Reference	[MQTT-3.1.2-18], [MQTT-3.1.2-20], [MQTT-3.1.2-22]
PICS Selection	PICS_CLIENT_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
    when {
        the IUT is triggered to send a CONNECT message containing
          connect_flags containing
            user_name_flag indicating value '0'B;
          ;
    }
    then {
        the IUT sends a CONNECT message containing
          connect_flags containing
            user_name_flag indicating value '0'B,
            password_flag indicating value '0'B;
            ,
          payload containing
            user_name indicating value omit,
            password indicating value omit;
          ;
    }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_008
Test Objective	Verify that the IUT is able to send CONNECT Control Packets with only a User Name.
Reference	[MQTT-3.1.2-19]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	
Expected Behaviour	

```
ensure that {
  when {
    the IUT is triggered to send a CONNECT message containing
      connect_flags containing
        user_name_flag indicating value '1'B,
        password_flag indicating value '0'B;
      ;
  }
  then {
    the IUT sends a CONNECT message containing
      connect_flags containing
        user_name_flag indicating value '1'B,
        password_flag indicating value '0'B;
      ,
      payload containing
        user_name indicating value not omit,
        password indicating value omit;
      ;
  }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_009
Test Objective	Verify that the IUT is able to send CONNECT Control Packets with a User Name and Password.
Reference	[MQTT-3.1.2-21], [MQTT-3.1.2-21]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	
Expected Behaviour	

```
ensure that {
  when {
    the IUT is triggered to send a CONNECT message containing
      connect_flags containing
        user_name_flag indicating value '1'B,
        password_flag indicating value '1'B;
      ;
  }
  then {
    the IUT sends a CONNECT message containing
      connect_flags containing
        user_name_flag indicating value '1'B,
        password_flag indicating value '1'B;
      ,
      payload containing
        user_name indicating value not omit,
        password indicating value not omit;
      ;
  }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_010
Test Objective	Verify that the IUT sends CONNECT Control Packets with Payload fields appearing in a correct order.
Reference	[MQTT-3.1.3-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	
Expected Behaviour	

```
ensure that {
  when {
    the IUT is triggered to send a CONNECT message containing
      connect_flags containing
        user_name_flag indicating value '1'B,
        password_flag indicating value '1'B,
        will_flag indicating value '1'B,
        will_qos corresponding to AT_MOST_ONCE;
      ,
      payload containing
        client_identifier corresponding to VALID_CLIENT_ID,
        will_topic corresponding to PX_WILL_TOPIC,
        will_message corresponding to PX_WILL_MESSAGE,
        user_name corresponding to PX_MQTT_USER_NAME,
        password corresponding to PX_MQTT_PASSWORD;
      ;
  }
  then {
    // Assumption: by comparing each field with the sent values, the order is checked implicitly.
      // Wrong order would silently swap the fields during decoding
      the IUT sends a CONNECT message containing
      connect_flags containing
        user_name_flag indicating value '1'B,
        password_flag indicating value '1'B,
        will_flag indicating value '1'B;
      ,
      payload containing
        client_identifier corresponding to VALID_CLIENT_ID,
        will_topic corresponding to PX_WILL_TOPIC,
        will_message corresponding to PX_WILL_MESSAGE,
        user_name corresponding to PX_MQTT_USER_NAME,
        password corresponding to PX_MQTT_PASSWORD;
      ;
  }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_011
Test Objective	Verify that the IUT is able to send CONNECT Control Packets with a well-formed UTF-8 encoded client identifier.
Reference	[MQTT-3.1.3-4], [MQTT-1.5.3-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	
Expected Behaviour	

```
ensure that {
  when {
    the IUT is triggered to send a CONNECT message containing
      payload containing
        client_identifier corresponding to VALID_CLIENT_ID;   // TODO: required to trigger a concrete Client ID?
      ;
  }
  then {
    // TODO: sufficient for [MQTT-3.1.3-4] ?
      the IUT sends a CONNECT message containing
      payload containing
        client_identifier corresponding to VALID_CLIENT_ID;
```

```
        ;
  }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_012
Test Objective	Verify that the IUT sets the clean_session flag to 1 if it connects with a zero-byte client identifier.
Reference	[MQTT-3.1.3-7]
PICS Selection	PICS_CLIENT_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
  when {
      the IUT is triggered to send a CONNECT message containing
        payload containing
          client_identifier corresponding to CLIENT_ID_ZERO_BYTES;
        ;
  }
  then {
      the IUT sends a CONNECT message containing
        connect_flags containing
          clean_session indicating value '1'B;
        ,
        payload containing
          client_identifier corresponding to CLIENT_ID_ZERO_BYTES;
        ;
  }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_013
Test Objective	Verify that IUT encodes the Will Topic to well-formed UTF-8 encoded string.
Reference	[MQTT-3.1.3-10]
PICS Selection	PICS_CLIENT_BASIC

Initial Conditions

Expected Behaviour

```
ensure that {
  when {
      the IUT is triggered to send a CONNECT message containing
        connect_flags containing
          will_flag indicating value '1'B;
        ,
        payload containing
          will_topic corresponding to PX_WILL_TOPIC;    // TODO: required to trigger a concrete topic?
        ;
  }
  then {
      // TODO: sufficient for [MQTT-3.1.3-10] ?
        the IUT sends a CONNECT message containing
        connect_flags containing
          will_flag indicating value '1'B;
        ,
        payload containing
          will_topic corresponding to TOPIC_NAME_VALID;
        ;
  }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_CONNECT_014
Test Objective	Verify that IUT encodes the User Name to well-formed UTF-8 encoded string.
Reference	[MQTT-3.1.3-11]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	
Expected Behaviour	

ensure that {
  when {
  the IUT is triggered to send a CONNECT message containing
  connect_flags containing
  user_name_flag indicating value '1'B;
  ,
  payload containing
  user_name corresponding to PX_MQTT_USER_NAME; // TODO: required to trigger a concrete username?
  ;
  }
  then {
  // TODO: sufficient for [MQTT-3.1.3-11] ?
  the IUT sends a CONNECT message containing
  connect_flags containing
  user_name_flag indicating value '1'B;
  ,
  payload containing
  user_name corresponding to USER_NAME_VALID_UTF8;
  ;
  }
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_CONNACK_001
Test Objective	Verify that the IUT closes the network connection on receiption of a CONNACK Control Packet with invalid fixed header.
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2], [MQTT-4.8.0-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
  the IUT is triggered to send a CONNECT message
}	

Expected Behaviour	

ensure that {
  when {
  the IUT receives a CONNACK message containing
  header_flags indicating value '1111'B;
  }
  then {
  the IUT closes the TCP_CONNECTION
  }
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_PUBLISH_001
Test Objective	Verify that the IUT sets the DUP flag to 0 for all QoS 0 PUBLISH Control Packets.
Reference	[MQTT-3.3.1-2]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
  the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	
ensure that { when {	

```
        the IUT is triggered to send a PUBLISH message containing
        header containing
          qos_level corresponding to AT_MOST_ONCE;
        ;
    }
    then {
        the IUT sends a PUBLISH message containing
        header containing
          qos_level corresponding to AT_MOST_ONCE,
          dup_flag indicating value '0'B;
        ;
    }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBLISH_002
Test Objective	Verify that the IUT sets the DUP flag to 0 for all QoS 1 PUBLISH Control Packets.
Reference	[MQTT-3.3.1-1]
PICS Selection	PICS_CLIENT_QOS_1

Initial Conditions

```
with {
        the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT is triggered to send a PUBLISH message containing
        header containing
          qos_level corresponding to AT_LEAST_ONCE;
        ;
    }
    then {
        the IUT is triggered to send a PUBLISH message containing
        header containing
            qos_level corresponding to AT_LEAST_ONCE,
            dup_flag indicating value '0'B;
        ;
    }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBLISH_003
Test Objective	Verify that the IUT sets the DUP flag to 0 for all QoS 2 PUBLISH Control Packets.
Reference	[MQTT-3.3.1-1]
PICS Selection	PICS_CLIENT_QOS_2

Initial Conditions

```
with {
        the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT is triggered to send a PUBLISH message containing
            qos_level corresponding to EXACTLY_ONCE;
    }
    then {
        the IUT sends a PUBLISH message containing header containing
            qos_level corresponding to EXACTLY_ONCE,
            dup_flag indicating value '0'B;
        ;
    }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBLISH_004
Test Objective	Verify that IUT encodes the topic name to a well-formed UTF-8 encoded string.
Reference	[MQTT-3.3.2-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a PUBLISH message
}
then {
the IUT sends a PUBLISH message containing
topic_name not corresponding to TOPIC_NAME_INVALID_UTF8;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_PUBLISH_005
Test Objective	Verify that the IUT does not send PUBLISH Control Packets which contain only valid topic names without wildcard characters.
Reference	[MQTT-3.3.2-2]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a PUBLISH message
}
then {
the IUT sends a PUBLISH message containing
topic_name corresponding to TOPIC_NAME_VALID;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_PUBLISH_006
Test Objective	Verify that the IUT does not send any response on reception of a QoS level 0 PUBLISH Control Packet.
Reference	[MQTT-3.3.4-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
and
the IUT subscribed the PX_PUBLISH_TOPIC to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to AT_MOST_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC;;	

```
   }
   then {
        the IUT sends no response message
   }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBLISH_007
Test Objective	Verify that the IUT responds to a QoS level 1 PUBLISH Control Packet with a PUBACK Control Packet.
Reference	[MQTT-3.3.4-1]
PICS Selection	PICS_CLIENT_QOS_1

Initial Conditions

```
with {
        the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
       and
       the IUT subscribed the PX_PUBLISH_TOPIC to the TEST_SYSTEM
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a PUBLISH message containing
           header containing
              qos_level corresponding to AT_LEAST_ONCE,
           packet_identifier corresponding to PACKET_ID,
           topic_name corresponding to PX_PUBLISH_TOPIC;;
   }
   then {
        the IUT sends a PUBACK message containing
           packet_identifier corresponding to PACKET_ID;
   }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBLISH_008
Test Objective	Verify that the IUT responds to a QoS level 2 PUBLISH Control Packet with a PUBREC Control Packet.
Reference	[MQTT-3.3.4-1]
PICS Selection	PICS_CLIENT_QOS_2

Initial Conditions

```
with {
        the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
       and
       the IUT subscribed the PX_PUBLISH_TOPIC to the TEST_SYSTEM
}
```

Expected Behaviour

```
ensure that {
   when {
        the IUT receives a PUBLISH message containing
           header containing
              qos_level corresponding to EXACTLY_ONCE,
           packet_identifier corresponding to PACKET_ID,
           topic_name corresponding to PX_PUBLISH_TOPIC;;
   }
   then {
        the IUT sends a PUBREC message containing
           packet_identifier corresponding to PACKET_ID;
   }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBLISH_009
Test Objective	Verify that the IUT assigns a non-zero packet identifier on each new PUBLISH Control Packet with QoS level > 0
Reference	[MQTT-2.3.1-1]
PICS Selection	PICS_CLIENT_QOS_1
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a PUBLISH message containing
header containing
qos_level corresponding to AT_LEAST_ONCE;;
}
then {
the IUT sends a PUBLISH message containing
packet_identifier corresponding to PACKET_ID_NON_ZERO;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_PUBLISH_010
Test Objective	Verify that the IUT assigns a currently unused packet identifier on each new PUBLISH Control Packet with QoS level > 0
Reference	[MQTT-2.3.1-2]
PICS Selection	PICS_CLIENT_QOS_1
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a PUBLISH message containing
header containing
qos_level corresponding to AT_LEAST_ONCE;;
and
the IUT is triggered to send a PUBLISH message containing
header containing
qos_level corresponding to AT_LEAST_ONCE;;
}
then {
the IUT sends a PUBLISH message containing
packet_identifier corresponding to PACKET_ID_1;
and
the IUT sends a PUBLISH message containing
packet_identifier corresponding to PACKET_ID_2;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_PUBLISH_011
Test Objective	Verify that the IUT does not assign a packet identifier on PUBLISH Control Packet with QoS level equals 0
Reference	[MQTT-2.3.1-5]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

```
ensure that {
   when {
         the IUT is triggered to send a PUBLISH message containing
        header containing
           qos_level corresponding to AT_MOST_ONCE;;
   }
   then {
         the IUT sends a PUBLISH message containing
           packet_identifier indicating value omit;
   }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBACK_001
Test Objective	Verify that the IUT is able to send PUBACK Control Packets with valid Header Flags.
Reference	[MQTT-2.2.2-1]
PICS Selection	PICS_CLIENT_BASIC

Initial Conditions

```
with {
         the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
       and
       the IUT subscribed the PX_PUBLISH_TOPIC to the TEST_SYSTEM
}
```

Expected Behaviour

```
ensure that {
   when {
         the IUT receives a PUBLISH message containing
           header containing
              qos_level corresponding to AT_LEAST_ONCE,
           topic_name corresponding to PX_PUBLISH_TOPIC;;
   }
   then {
         the IUT sends a PUBACK message containing
           header_flags indicating value '0000'B;
   }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBACK_002
Test Objective	Verify that the IUT acknowledges a PUBLISH Control Packet with the correct packet identifier.
Reference	[MQTT-2.3.1-6]
PICS Selection	PICS_CLIENT_QOS_1

Initial Conditions

```
with {
         the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
       and
       the IUT subscribed the PX_PUBLISH_TOPIC to the TEST_SYSTEM
}
```

Expected Behaviour

```
ensure that {
   when {
         the IUT receives a PUBLISH message containing
           header containing
              qos_level corresponding to AT_LEAST_ONCE,
           topic_name corresponding to PX_PUBLISH_TOPIC,
           packet_identifier corresponding to PACKET_ID;;
   }
   then {
         the IUT sends a PUBACK message containing
           packet_identifier corresponding to PACKET_ID;
   }
}
```

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBREC_001
Test Objective	Verify that the IUT is able to send PUBREC Control Packets with valid Header Flags.
Reference	[MQTT-2.2.2-1]
PICS Selection	PICS_CLIENT_QOS_2
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
and
the IUT subscribed the PX_PUBLISH_TOPIC to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC;;
}
then {
the IUT sends a PUBREC message containing
header_flags indicating value '0000'B;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_PUBREC_002
Test Objective	Verify that the IUT acknowledges a PUBLISH Control Packet with the correct packet identifier.
Reference	[MQTT-2.3.1-6]
PICS Selection	PICS_CLIENT_QOS_2
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
and
the IUT subscribed the PX_PUBLISH_TOPIC to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT receives a PUBLISH message containing
header containing
qos_level corresponding to EXACTLY_ONCE,
topic_name corresponding to PX_PUBLISH_TOPIC,
packet_identifier corresponding to PACKET_ID;;
}
then {
the IUT sends a PUBREC message containing
acket_identifier corresponding to PACKET_ID;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_PUBREL_001
Test Objective	Verify that the IUT is able to send PUBREL Control Packets with valid Header Flags.
Reference	[MQTT-2.2.2-1]
PICS Selection	PICS_CLIENT_QOS_2
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour

ensure that {
  when {
  the IUT receives a PUBREC message
  }
  then {
  the IUT sends a PUBREL message containing
  header_flags indicating value '0000'B;
  }
}

Final Conditions

TP Id	TP_MQTT_CLIENT_PUBREL_002
Test Objective	Verify that the IUT acknowledges a PUBREC Control Packet with the correct packet identifier with a PUBREL Control Packet.
Reference	[MQTT-2.3.1-6]
PICS Selection	PICS_CLIENT_QOS_2
Initial Conditions	

with {
  the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
  when {
  the IUT receives a PUBREC message containing
  packet_identifier corresponding to PACKET_ID;
  }
  then {
  the IUT sends a PUBREL message containing
  packet_identifier corresponding to PACKET_ID;
  }
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_PUBCOMP_001
Test Objective	Verify that the IUT is able to send PUBCOMP Control Packets with valid Header Flags.
Reference	[MQTT-2.2.2-1]
PICS Selection	PICS_CLIENT_QOS_2
Initial Conditions	

with {
  the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
  and
  the IUT subscribed the PX_PUBLISH_TOPIC to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
  when {
  the IUT receives a PUBREL message
  }
  then {
  the IUT sends a PUBCOMP message containing
  header_flags indicating value '0000'B;
  }
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_SUBSCRIBE_001
Test Objective	Verify that the IUT is able to send SUBSCRIBE Control Packets with valid Header Flags.
Reference	[MQTT-2.2.2-1], [MQTT-3.8.1-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a SUBSCRIBE message
}
then {
the IUT sends a SUBSCRIBE message containing
header_flags indicating value '0010'B;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_SUBSCRIBE_002
Test Objective	Verify that the IUT assigns a non-zero packet identifier on each new SUBSCRIBE Control Packet
Reference	[MQTT-2.3.1-2]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a SUBSCRIBE message
}
then {
the IUT sends a SUBSCRIBE message containing
packet_identifier corresponding to PACKET_ID_1;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_SUBSCRIBE_003
Test Objective	Verify that the IUT assigns a currently unsued packet identifier on each new SUBSCRIBE Control Packet
Reference	[MQTT-2.3.1-2]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a SUBSCRIBE message
and
the IUT is triggered to send a SUBSCRIBE message
}
then {
the IUT sends a SUBSCRIBE message containing
packet_identifier corresponding to PACKET_ID_1;
and
the IUT sends a SUBSCRIBE message containing
packet_identifier corresponding to PACKET_ID_2;
}	

}
Final Conditions

TP Id	TP_MQTT_CLIENT_UNSUBSCRIBE_001
Test Objective	Verify that the IUT is able to send UNSUBSCRIBE Control Packets with valid Header Flags.
Reference	[MQTT-2.2.2-1], [MQTT-3.10.1-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a UNSUBSCRIBE message
}
then {
the IUT sends a UNSUBSCRIBE message containing
header_flags indicating value '0010'B;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_UNSUBSCRIBE_002
Test Objective	Verify that IUT encodes the topic filter to a well-formed UTF-8 encoded string.
Reference	[MQTT-3.10.3-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a UNSUBSCRIBE message
}
then {
the IUT sends a UNSUBSCRIBE message containing
payload containing
topic_filter not corresponding to TOPIC_FILTER_INVALID_UTF8;;
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_DISCONNECT_001
Test Objective	Verify that the IUT is able to send DISCONNECT Control Packets with valid Header Flags.
Reference	[MQTT-2.2.2-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a DISCONNECT message
}
then {
the IUT sends a DISCONNECT message containing
header_flags indicating value '0000'B;
}	

}
Final Conditions

TP Id	TP_MQTT_CLIENT_DISCONNECT_002
Test Objective	Verify that the IUT closes the network connection after sending a DISCONNECT Control Packet.
Reference	[MQTT-3.14.4-1]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION to the TEST_SYSTEM
}	

Expected Behaviour	

ensure that {
when {
the IUT is triggered to send a DISCONNECT message
}
then {
the IUT sends a DISCONNECT message
and
the IUT closes the TCP_CONNECTION
}
}	

Final Conditions	

TP Id	TP_MQTT_CLIENT_FEAT_KEEPALIVE_001
Test Objective	Verify that the IUT ensures that the interval between Control Packets being sent does not exceed the Keep Alive value.
Reference	[MQTT-3.1.2-23]
PICS Selection	PICS_CLIENT_BASIC
Initial Conditions	

with {
the IUT having a MQTT_CONNECTION containing
keep_alive corresponding to PX_KEEP_ALIVE;
}	

Expected Behaviour	

ensure that {
when {
the MQTT_CONNECTION times_out
}
then {
the IUT sends a PINGREQ message
}
}	

Final Conditions	

Annex A (normative): MQTT Test Purposes (TPs)

A.0 Introduction

This Test purpose catalogue has been produced using the Test Description Language (TDL-TO) according to ETSI ES 203 119-4 [2]. The TDL-TO library modules corresponding to the Test purpose catalogue are contained in archive ts_10359701v010101p0.zip which accompanies the present document.

History

Document history		
V1.1.1	January 2021	Publication