OPEN Alliance
Automotive Ethernet ECU Test Specification

TC8 ECU Test

Author & Company	Thomas Kirchmeier (BMW AG) Georg Janker (Ruetz System Solutions GmbH) All Members of the OPEN ALLIANCE TC8 Working Group
Title	ECU Test, Test Specification ECU
Version	2.0
Date	August 23, 2017
Status	Final
Restriction Level	Public

OPEN Alliance: Public OPEN Specification

Contents

1 Introduction

1.1 Overview

This ECU and Network Test Specification is designed to determine if a product conforms to specifications defined in OPEN Specifications or related requirements. This specification is a collection of all test cases which are recommend to be considered for automotive use and should be referred by car manufacturers within their quality control processes.
Sucessful execution and passing all relevant tests gives a Device Under Test (DUT) a mimimum approval that the device's basic implementiations are done correctly.

This Test specification document is grouped in several chapters oriented on the scopes: "Automotive Ethernet", "TCP/IP Protocol Family" and "Automotive Protocols" which are described in chapter 1.3. Tests are organized and identified with distinct IDs that relate to their scopes, and a unique enumeration. For every scope introduction chapters explain common requirements on the Device under Test, the Test Setup and parameters used by the following tests.

1.2 Definition of Test Scopes

1.2.1 Test Scope Automotive Ethernet

Scope Automotive Ethernet includes the following ISO/OSI layers:

- Layer 1: Physical Layer OPEN Alliance BroadR-Reach (OABR)
- Layer 2: Data Link Layer, e.g IEEE Ethernet MAC + VLAN (802.1Q), ARP

1.2.2 Test Scope TCP/IP Protocol Family

Scope TCP/IP Protocol Family includes the following ISO/OSI layers:

- Layer 3: Network Layer, e.g. IP, ICMP
- Layer 4: Transport Layer, e.g. UDP, TCP, DHCP

1.2.3 Test Scope Automotive Protocols

Scope Automotive Protocols includes the following ISO/OSI layer:

- Layers 5-6-7: Application oriented layers, e.g. UDP-NM, SOME/IP, SD

1.3 Version Control of Document

Version	Author	Description	Date
1.0	TC8 members	First release	15.01.2016
1.1	T.Kirchmeier (BMW)	Improvments regarding IPv4 test cases, see change history	31.05.2016
1.2	T.Kirchmeier (BMW)	Improvments regarding UDP test cases, see change history	29.06.2016
1.3	T.Kirchmeier (BMW)	Improvments regarding ICMPv4 test cases, see change history	07.09.2016
1.4	Mathias Kleinwächter (Ruetz System Solutions GmbH)	Chapter 5.6 DHCPv4 Server deleted	19.05.2017
1.4	Mathias Kleinwächter (Ruetz System Solutions GmbH)	Improvements regarding TCP test cases, see change history	23.05.2017
1.4	Mathias Kleinwächter (Ruetz System Solutions GmbH)	Added chapters 6.1.4 Specification of the SOMEIP TestStub Enhanced Testability Service (ETS) 6.1.6 Test Cases ETS Improvements regarding ARP test cases, see change history	24.05.2017
1.4	Georg Janker	Update of Layer 1 and Layer2 Chapters	24.05.2017
1.5	Georg Janker	Update of AUTOSAR References for SOME/IP to 1.1.0	30.05.2017
1.5	Georg Janker	Inserted Chapter: 3.6 Referenced TC 11 Tests	30.05.2017
1.6	Martin Heinzinger (Ruetz System Solutions GmbH)	Removed Port Disabling test and referenced to the corresponding TC11 Test	07.06.2017

1.7	Mathias Kleinwächter (Ruetz System Solutions GmbH)	Deleted invalid or duplicate Test Cases. See change history	20.06.2017
1.8	Frederic Garraud	Update 1.3 References	22.06.2017
1.9	Martin Heinzinger (Ruetz System Solutions GmbH)	Updated change history for L2 Switching	23.06.2017

1.4 Change history between version 1 and 2

Test case ID	Change reason	Version 1	Version 2
IPv4_TTL_05	Renamed	ICMPv4 Echo Reply (TTL value check)	Packets with 0 or 1 TTL are not discarded by hosts
IPv4_TTL_01	Renamed	Ensure that the DUT generates an IPv4 Packet with TTL value greater than 0	A host MUST NOT send a datagram with a Time-to-Live (TTL) value of zero
IPv4_TTL_01	Changed synopsis	Ensure that when the DUT generates an IPv4 packet then the DUT sends an IPv4 packet containing an IPv4 Header containing a TTL indicating a value greater than 0.	A host MUST NOT send a datagram with a Time-to-Live (TTL) value of zero
IPv4_TTL_03	Deprecated		Test case only applicable for router
IPv4_TTL_04	Deprecated		Test case only applicable for router
IPv4_CHECKSUM_01	Deprecated		test case is duplicated to IPv4_CHECKSUM_05
IPv4_CHECKSUM_02	Renamed	Ensure that the DUT discards an IPv4 Packet with an invalid Header Checksum	IP Checksum method validation on receiving
IPv4_CHECKSUM_02	Changed synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a Header Checksum indicating an invalid <invalidChecksum>, then the DUT discards the IPv4 Packet silently.	If the header checksum fails, the internet datagram is discarded at once by the entity which detects the error.
IPv4_CHECKSUM_04	Changed test case description	1. TESTER: Cause DUT to send an ICMP Echo Request through <DIface-0>, containing: - IP Destination Address set to address of host-1 ... 3. DUT: Send ICMP Echo Request 4. TESTER: Verify that the	1. TESTER: Cause DUT to send a IP packet through <DIface-0>, containing: - IP Destination Address set to address of host-1 ... 3. DUT: Send a IP packet 4. TESTER: Verify that the received IP packet contains:

		received ICMP Echo RequestIP contains: …	
IPv4_CHECKSUM_04	Additional note		Request / Reply Mechnism has to be supported by DUT and a packet may be an ICMP Request or UDP datagram on a port DUT is listening.
IPv4_CHECKSUM_04	Renamed	ICMPv4 Echo Request (IP Checksum method validation on sending)	IP Checksum method validation on sending
IPv4_CHECKSUM_05	Renamed	ICMPv4 Echo Reply (IP Checksum method validation on receiving)	IP Checksum method validation
IPv4_FRAGMENTS_01	Renamed	ICMPv4 Echo Reply (IP Reconstruct fragments)	IP Reconstruct fragments validation
IPv4_FRAGMENTS_05	Changed test case description	Test step 1. TESTER: Cause DUT to send an ICMP Echo Request through <DIface-0>, containing: - IP Destination Address set to address of host-1	Test step 1. TESTER: Cause DUT to send a Message with <UDPDefaultData> from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> - Source UDP Port field set to <unusedUDPDstPort1> (20001) - Destination UDP Port field set to <unusedUDPSrcPort> (20000) Additional note: The message can be an ICMP Echo Request or a simple UDP message.
IPv4_FRAGMENTS_05	Renamed	"ICMPv4 Echo Request (IP Fragmentation check)	IP send unfragmented data validation
IPv4_REASSEMBLY_01	Deprecated		Test case is duplicated to IPv4_FRAGMENTS_01
IPv4_REASSEMBLY_02	Deprecated		Test case is duplicated to IPv4_FRAGMENTS_02
IPv4_REASSEMBLY_03	Deprecated		Test case is duplicated to IPv4_FRAGMENTS_04
IPv4_REASSEMBLY_05	Deprecated		Test case is duplicated to IPv4_REASSEMBLY_13
IPv4_REASSEMBLY_08	Deprecated		Test case is duplicated to IPv4_REASSEMBLY_10
IPv4_HEADER_05	Renamed	ICMPv4 Echo Reply	IP Maximum datagram length check

IPv4_HEADER_08	Renamed	ICMPv4 Echo Request (IP Header length validation)	IP Header length validation
IPv4_HEADER_09	Renamed	ICMPv4 Echo Request (IP Total Length validation)	IP Total Length validation”
IPv4_VERSION_04	Renamed	ICMPv4 Echo Reply (IP Version validation)	IP Version validation
IPv4_FRAGMENTS_02	Remamed	ICMPv4 Echo Reply (IP Reconstruct fragments, negative, id)	IP Reconstruct fragments, negative test on id
IPv4_FRAGMENTS_03	Renamed	ICMPv4 Echo Reply (IP Reconstruct fragments, negative, source)	IP Reconstruct fragments, negative test on source
IPv4_FRAGMENTS_04	Renamed	ICMPv4 Echo Reply (IP Reconstruct fragments, negative, protocol)	IP Reconstruct fragments, negative test on protocol
IPv4_REASSEMBLY_10	Renamed	ICMPv4 Echo Reply (IP Reassembly default time check)	IP Reassembly default time check
IPv4_REASSEMBLY_11	Renamed	ICMPv4 Echo Reply (check fragment with Large TTL value)	Check fragment with Large TTL value
IPv4_REASSEMBLY_12	Renamed	ICMPv4 Echo Reply (check fragment with Low TTL value)	Check fragment with Low TTL value
IPv4_REASSEMBLY_13	Renamed	ICMPv4 Echo Reply (IP Fragments overlap check)	IP Fragments overlap check
IPv4_OPTIONS_01	Renamed	ICMPv4 Echo Reply (Multiple End of Options check)	Multiple End of Options check
IPv4_OPTIONS_02	Renamed	ICMPv4 Echo Reply (Multiple No Operation option check)	Multiple No Operation option check
IPv4_OPTIONS_03	Renamed	ICMPv4 Echo Reply (Multiple No Operation and End of Option check)	Multiple No Operation and End of Option check
IPv4_OPTIONS_04	Renamed	ICMPv4 Echo Reply (Security Option check)	Security Option check
IPv4_OPTIONS_05	Renamed	ICMPv4 Echo Reply (Overflow Bit in timestamp option check)	Overflow Bit in timestamp option check
IPv4_OPTIONS_06	Renamed	ICMPv4 Echo Reply (timestamp value check)	Timestamp value check
IPv4_OPTIONS_07	Renamed	ICMPv4 Echo Reply (timestamp value, internet address and sequence check)	Timestamp value, internet address and sequence check
IPv4_OPTIONS_08	Renamed	ICMPv4 Echo Reply (relation between timestamp & its	Relation between timestamp & its own

		own address)	address
IPv4_OPTIONS_09	Renamed	ICMPv4 Echo Reply (relation between timestamp & its own address, negative)	Relation between timestamp & its own address, negative
IPv4_OPTIONS_10	Renamed	ICMPv4 Echo Reply (room for timestamp check)	Room for timestamp check
IPv4_OPTIONS_11	Renamed	ICMPv4 Echo Reply (overflow bits validation)	Overflow bits validation
IPv4_OPTIONS_12	Renamed	ICMPv4 Echo Reply (check room for timestamp)	Check room for timestamp
IPv4_OPTIONS_13	Renamed	ICMPv4 Echo Reply (validate overflow bits)	Validate overflow bits
IPv4_OPTIONS_14	Renamed	ICMPv4 Echo Reply (timestamp option occurrence check)	Timestamp option occurrence check
UDP_FIELDS_02	Additional note	[Note: In this test, we verify that DUT will respond back(if required) to source port of received message]	[Note: In this test, we verify that DUT can respond respond back (if required) to source port of received message. This has two parts: being able to read the source port of a received message, and being able to set the destination port of a sent message - these operations are available on the AUTOSAR testability protocol. In this case, the tester is basically implementing the application logic of issuing the reply by properly reading the source port and replying to that port.]
UDP_FIELDS_03	Renamed	"Fields - Source Port received and sent to zero"	"Fields - Accept Source Port set to zero"
UDP_FIELDS_11	Deprecated		Remove as it conflicts with UDP_DatagramLength_01
UDP_FIELDS_12	Additional note	Note: Check that the DUT accepts the received datagram in case the length value in the header is set to the maximum value	Note: Check that the DUT accepts the received datagram in case the length value in the header is set to the maximum allowed value. - IPv4: 65,507 bytes (65,535 − 8 byte UDP header − 20 byte IP header)

			- IPv6: 65,535 bytes
UDP_FIELDS_17	Deprecated		There is no use case for this test case in Automotive.
UDP_FIELDS_18	Deprecated		There is no use case for this test case in Automotive.
UDP_MessageFormat_01	Deprecated		Test case is duplicated to UDP_FIELDS_1, UDP_FIELDS_2, UDP_FIELDS_06, UDP_FIELDS_13, UDP_FIELDS_14
UDP_PortHandling_01	Deprecated		Test case is duplicated to UDP_MessageFormat_02
UDP_PortHandling_02	Deprecated		Test case is duplicated to UDP_FIELDS_03
UDP_PortHandling_03	Deprecated		Test case is duplicated to UDP_MessageFormat_02
UDP_PortHandling_04	Deprecated		There is no use case for this test case in Automotive.
UDP_DatagramLength_01	Changed synopsis and additional note	Ensure that when the DUT receives a truncated UDP packet then the DUT discards the UDP packet. Derived from RFC768, section Format	Ensure that when the DUT receives a truncated UDP packet (a packet with the length field smaller than the actual size of the data coming from the Ethernet frame) then the DUT discards the UDP packet. Derived from RFC768, section Format and automotive specific requirements on frame integrity
UDP_DatagramLength_02	Deprecated		Test case is duplicated to UDP_FIELDS_08
UDP_DatagramLength_03	Deprecated		Test case is duplicated to UDP_MessageFormat_02
UDP_DatagramLength_04	Deprecated		Test case is duplicated to UDP_FIELDS_09
UDP_DatagramLength_05	Deprecated		Test case is duplicated to UDP_FIELDS_10
UDP_DatagramLength_06	Deprecated		Test case is duplicated to UDP_DatagramLength_01
UDP_DatagramLength_07	Deprecated		Test case is invalid.
UDP_Padding_01	Deprecated		Test case is invalid.
UDP_Padding_02	Changed test case	2. DUT: Generates a UDP packet with padding.	2. DUT: Generates a UDP packet without padding.

	description		
ICMPv4_ERROR_01	Renamed	ICMPv4 Parameter Problem Message (avoid the infinite regress)	Avoid the infinite loop for ICMP message error
ICMPv4_ERROR_02	Renamed	ICMPv4 Parameter Problem Message (errornous fragmentation)	ICMP messages are only sent for fragment 0
ICMPv4_ERROR_03	Renamed	ICMPv4 Parameter Problem Message (non zero fragment)	ICMP messages are not sent when fragment not 0
ICMPv4_ERROR_04	Renamed	ICMPv4 Parameter Problem Message (broadcasting mechanism)	ICMP messages are not sent for broadcast address
ICMPv4_ERROR_05	Renamed	ICMPv4 Messages (unkown message type)	Uknown ICMP message types are ignored
ICMPv4_TYPE_16	Renamed, changed synopsis and changed test case description	<u>Titel</u>: Ensure that the DUT does not accept an ICMPv4 Information Request and does not generate a ICMPv4 Information Reply <u>Synopsis</u>: Ensure that when a DUT receives an ICMPv4 Packet containing a Type indicating a value of 15 (Information Request) and containing a Code indicating a value of 0 and containing a valid Checksum and containing an Identifier indicating a value of ID1 and containing a Sequence Number indicating a value of SEQ1 then the DUT discards the message and does not send an ICMPv4 Packet containing a Type indicating a value of 16 (Information Reply) <u>Description</u>: 1. TESTER: Send an ICMPv4 Information Request message 2. DUT: Does not send an	<u>Title</u>: Do not implement ICMP Information messages <u>Synopsis</u>: A host SHOULD NOT implement these messages. [Note: referring to Information Request or Information Reply Message] <u>Description</u>: 1) TESTER: Send an ICMP Information Request to <DIface-0>, containing: - IP Source Address field set to network portion of IP address of host-1 - IP Destination Address field set to zero - Identifier field set to <idfr> - Sequence Number field set to <seqno> 2) TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3) DUT: Do not send ICMP Information Reply Pass criteria: 3) DUT: Do not send ICMP Information Reply

		ICMPv4 Information Reply message	
ICMPv4_TYPE_17	Deprecated		Test case is duplicated to ICMPv4_TYPE_11, ICMPv4_TYPE_12 and ICMPv4_TYPE_17
ICMPv4_TYPE_18	Renamed	Ensure that the DUT generates an ICMPv4 Destination Unreachable message with Code Protocol Unreachable	Send ICMP Destination Unreachable for unknown protocol
ICMPv4_TYPE_21	Deprecated		Test case is duplicated to ICMPv4_TYPE_04
ICMPv4_TYPE_01	Renamed	ICMPv4 Destination Unreachable Message (unknown Destination port)	Send ICMP Destination Unreachable for unknown port
ICMPv4_TYPE_02	Renamed	ICMPv4 Echo Reply (missing fragments)	Discard packet with missing fragments
ICMPv4_TYPE_03	Renamed	ICMPv4 Time Exceeded message (fragment reassembly time expire)	Send ICMP Time Exceeded message on fragmentation error
ICMPv4_TYPE_04	Renamed	ICMPv4 Time Exceeded message (with out fragment zero)	Do no send ICMP Time Exceeded message if missing fragment 0
ICMPv4_TYPE_05	Renamed	Datagram receive (Discard those with header parameter problems)	Discard messages with header parameter problem
ICMPv4_TYPE_06	Renamed	ICMP send (Parameter Problem Message)	Sending of ICMP Prameter Problem message
ICMPv4_TYPE_07	Renamed	ICMP send (Parameter Problem Message contents)	ICMP Parameter Problem Message contents
ICMPv4_TYPE_08	Renamed	ICMPv4 Echo Reply message (data field)	ICMP Echo Reply message data field
ICMPv4_TYPE_09	Renamed	ICMPv4 Echo Reply message (id and sequence)	ICMP Echo Reply message id and sequence field
ICMPv4_TYPE_10	Renamed	ICMPv4 Echo Reply message (incorrect ICMP checksum)	ICMP checksum is checked
ICMPv4_TYPE_11	Renamed	ICMPv4 Timestamp Reply message (additional timestamp)	ICMP Timestamp Reply message content
ICMPv4_TYPE_12	Renamed	ICMPv4 Timestamp Reply message (id and sequence)	ICMP Timestamp Reply message id and sequence field
ICMPv4_TYPE_22	Renamed	ICMPv4 Echo Reply	Send ICMP Echo Reply on receiving ICMP Echo Request
TCP_BASICS_03	Changed	2. TESTER: Send a FIN	2. TESTER: Send a FIN ,ACK

	Test Procedure		
TCP_BASICS_14	Changed Pass Criteria	3. DUT: Send ACK (DUT moves to TIME-WAIT state) 4. DUT: Send ACK	4. DUT: Send ACK
TCP_BASICS_17	Changed Test Procedure	6. TESTER: Verify that the DUT moves on to ESTABLISHED state	6. DUT: Send ACK 7. TESTER: Verify that the DUT moves on to ESTABLISHED state
TCP_UNACCEPTABLE_14	Changed Test Procedure and Pass Criteria		6.TESTER: Verify that the DUT stays in CLOSE-WAIT state
TCP_CALL_OPEN_02	Changed Pass Criteria	2. DUT: Issue a passive open call 3. TESTER: Verify that the DUT refuses to create new connection indicating \"error: connection illegal for this process\" to the application	3. TESTER: Verify that the DUT refuses to create new connection indicating \"error: connection illegal for this process\" to the application
TCP_CALL_OPEN_03	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
TCP_CALL_OPEN_04	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
TCP_CALL_OPEN_05	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
TCP_CALL_OPEN_06	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
TCP_CALL_OPEN_07	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application

TCP_CALL_OPEN_08	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
TCP_CALL_OPEN_09	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
TCP_CALL_OPEN_10	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
TCP_CALL_OPEN_11	Changed Pass Criteria	3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
TCP_CALL_SEND_01	Changed Pass Criteria	2. DUT: Issue a SEND call 3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application	3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
TCP_CALL_SEND_02	Changed Pass Criteria	3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_SEND_03	Changed Pass Criteria	3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_SEND_04	Changed Pass Criteria	3. DUT: Issues a SEND call 4. TESTER: Verify that the DUT returns \"error: foreign socket unspecified\" to the application	4. TESTER: Verify that the DUT returns \"error: foreign socket unspecified\" to the application
TCP_CALL_SEND_05	Changed Pass Criteria	3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_SEND_06	Changed Pass Criteria	3. DUT: Issue a SEND call 4. TESTER: Verify that the	4. TESTER: Verify that the DUT returns \"error:

		DUT returns \"error: connection closing\" to the application	connection closing\" to the application
TCP_CALL_SEND_09	Changed Pass Criteria	3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_RECEIVE_01	Changed Pass Criteria	3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_RECEIVE_02	Changed Pass Criteria	2. DUT: Issues a RECEIVE call 3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application	3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
TCP_CALL_RECEIVE_03	Changed Pass Criteria	3. DUT: Queue the request 5. DUT: Send ACK 7. TESTER: Verify that the DUT returns the data in response to the queued RECEIVE call	5. DUT: Send ACK 7. TESTER: Verify that the DUT returns the data in response to the queued RECEIVE call
TCP_CALL_RECEIVE_04	Changed Pass Criteria	4. DUT: Issues a RECEIVE call 5. TESTER: Verify that the DUT returns the reassembled data to the receiving application	5. TESTER: Verify that the DUT returns the reassembled data to the receiving application
TCP_CALL_RECEIVE_05	Changed Pass Criteria	4. DUT: Issues a RECEIVE call 5. TESTER: Verify that the DUT returns the data to the receiving application	5. TESTER: Verify that the DUT returns the data to the receiving application
TCP_CALL_RECEIVE_06	Changed Pass Criteria	3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT responds \"error: connection closing\" to the receiving application	4. TESTER: Verify that the DUT responds \"error: connection closing\" to the receiving application
TCP_CALL_RECEIVE_07	Changed Pass Criteria	3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT returns \"error:	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application

		connection closing\" to the application	
TCP_CALL_RECEIVE_08	Changed Pass Criteria	3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_CLOSE_01	Changed Pass Criteria	2. DUT: Issues a CLOSE call 3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application	3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
TCP_CALL_CLOSE_02	Changed Pass Criteria	3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_CLOSE_03	Changed Pass Criteria	3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_CLOSE_04	Changed Pass Criteria	3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_CLOSE_05	Changed Pass Criteria	3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
TCP_CALL_ABORT_01	Changed Pass Criteria	2. DUT: Issues an ABORT call 3. TESTER: Verify that DUT MUST return \"error: connection does not exist\" to the application	3. TESTER: Verify that DUT MUST return \"error: connection does not exist\" to the application
TCP_CALL_ABORT_02	Changed Test Procedure	3. DUT: Send a RESET control message	3. DUT: Send a RST control message
TCP_CALL_ABORT_03	Changed Pass Criteria	3. DUT: Issues an ABORT call 4. TESTER: Verify that the DUT responds with \"ok\" to the application	4. TESTER: Verify that the DUT responds with \"ok\" to the application 5. TESTER: Verify that the DUT moves on to CLOSED

		5. TESTER: Verify that the DUT moves on to CLOSED state	state
TCP_FLAGS_INVALID_06	Changed Test Iterations	1. CASE: <stp> = ACK segment 2. CASE: <stp> = Data segment	1. CASE: <stp> = no data 2. CASE: <stp> = data
TCP_FLAGS_INVALID_07	Changed Test Procedure	2. TESTER: Send a segment with a flag set and with an unacceptable SEQ number	2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number
TCP_FLAGS_INVALID_08	Changed Test Procedure	2. TESTER: Send a segment with a flag set and with an unacceptable SEQ number	2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number
TCP_FLAGS_INVALID_09	Changed Test Procedure	2. TESTER: Send a segment with a flag set and with an unacceptable SEQ number	2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number
TCP_FLAGS_INVALID_10	Changed Test Procedure	2. TESTER: Send a segment with a flag set and with an unacceptable SEQ number	2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number
TCP_FLAGS_INVALID_11	Changed Test Procedure	2. TESTER: Send a segment with a flag set and with an unacceptable SEQ number	2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number
TCP_FLAGS_INVALID_12	Changed Test Procedure	2. TESTER: Send a segment with a flag set and with an unacceptable SEQ number	2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number
TCP_FLAGS_INVALID_13	Changed Test Procedure	2. TESTER: Send a segment with a flag set and with an unacceptable SEQ number	2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number
TCP_FLAGS_INVALID_14	Changed Test Procedure	2. TESTER: Send a segment with a flag set and with an unacceptable SEQ number	2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number
TCP_FLAGS_PROCESSING_03	Changed Pass Criteria	3. DUT: Goes to CLOSED state 4. TESTER: Verify that the DUT moves on to CLOSED state	4. TESTER: Verify that the DUT moves on to CLOSED state
TCP_FLAGS_PROCESSING_04	Changed Pass Criteria	3. DUT: Goes to CLOSED state 4. TESTER: Verify that the DUT moves on to CLOSED	4. TESTER: Verify that the DUT moves on to CLOSED state

		state	
TCP_FLAGS_PROCESSING_05	Changed Pass Criteria	3. DUT: Send a RST segment 3. TESTER: Verify that the DUT moves on to CLOSED state	3. DUT: Send a RST segment 4.TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site and the DUT moves on to CLOSED state
TCP_FLAGS_PROCESSING_07	Changed Test Procedure	2. TESTER: Send a data segment with URG flag set	2. TESTER: Send a data segment with only URG flag set
TCP_FLAGS_PROCESSING_08	Changed Test Procedure	3. DUT: Do not send any response or retransmit SYN in SYN-SENT or LISTEN state, and send a RST segment in CLOSED state	3. DUT: in SYN-SENT or LISTEN state : Do not send any response or retransmit SYN In CLOSED state : send a RST segment
TCP_FLAGS_PROCESSING_09	Changed Pass Criteria	3. DUT: Do not change state 4. TESTER: Verify that the DUT remains in <wst> state	4. TESTER: Verify that the DUT remains in <wst> state
TCP_FLAGS_PROCESSING_11	Changed Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state at a <wnp> 2. TESTER: Cause DUT to move to ESTABLISHED state 3. TESTER: Send the last ACK once more 4. DUT: Do not send any response 5. TESTER: Verify that the DUT remains in ESTABLISHED state	1. TESTER: Cause DUT to move to ESTABLISHED state 2. TESTER: Send the last ACK once more 3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in ESTABLISHED state
TCP_CLOSING_01	Changed Pass Criteria	1. DUT: Move on to SYN-RCVD state starting with an active OPEN call 3. DUT: Go to CLOSED state 4. TESTER: Verify that DUT goes to CLOSED state	4. TESTER: Verify that DUT goes to CLOSED state
TCP_CLOSING_02	Changed Pass Criteria	1. DUT: Move on to ESTABLISHED state 4. TESTER: Verify that the receiving application receives the signal	4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the

		that \"connection reset\" has occurred from the remote site	remote site
TCP_CLOSING_03	Changed Test Procedure	2. TESTER: Send a RST segment containing some ASCII-data	2. TESTER: Send a RST segment containing some data
TCP_CLOSING_05	Changed Test Procedure	2. TESTER: Send a SYN on the same port	2. TESTER: Send a SYN
TCP_CLOSING_05	Changed Pass Criteria	3. DUT: Send SYN,ACK	3. DUT: Send SYN,ACK 4. TESTER: Verify that the DUT moves on to ESTABLISHED state
TCP_CLOSING_07	Changed Pass Criteria	3. DUT: DUT sends a FIN and Reaches FIN-WAIT-1 State 5. DUT: Issue a RECEIVE call 7. DUT: Send ACK of the received data 8. TESTER: Check that DUT receives proper data 9. TESTER: Check that DUT remains in FIN-WAIT-1 state	3. DUT: DUT sends a FIN 7. DUT: Send ACK of the received data 9. TESTER: Check that DUT remains in FIN-WAIT-1 state
TCP_CLOSING_09	Changed Pass Criteria	3. DUT: Send ACK for the received FIN 5. DUT: Send data 6. TESTER: Check that DUT sent proper data 7. TESTER: Check that DUT remains in CLOSE_WAIT state	3. DUT: Send ACK for the received FIN 6. TESTER: Check that DUT sent proper data 7. TESTER: Check that DUT remains in CLOSE_WAIT state
TCP_CLOSING_10	Changed Pass Criteria	1. DUT: Move on to FINWAIT-1 state 4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site	4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
TCP_CLOSING_11	Changed Pass Criteria	1. DUT: Move on to FINWAIT-2 state 4. TESTER: Verify that the	4. TESTER: Verify that the receiving application receives the signal that

		receiving application receives the signal that \"connection reset\" has occurred from the remote site	\"connection reset\" has occurred from the remote site
TCP_CLOSING_12	Changed Pass Criteria	1. DUT: Move on to CLOSE-WAIT state 4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site	4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
TCP_MSS_OPTIONS_06	Changed Test Procedure	5. TESTER: Cause the application to issue a SEND request for data with size at least max(transport_layer_max_send_msg_size-20, MSS) 7. TESTER: Verify that the first received segment has size that is min(transport_layer_max_send_msg_size-20, MSS)	5. TESTER: Cause the application to issue a SEND request for data with size at least max (MSS) 7. TESTER: Verify that the first received segment has size that is min (MSS)
TCP_MSS_OPTIONS_06	Changed Test Iterations	1. CASE: Mv is smaller than (transport_layer_max_send_msg_size-20) of DUT 2. CASE: Mv is larger than (transport_layer_max_send_msg_size-20) of DUT	1. CASE: Mv is smaller than MSS of DUT 2. CASE: Mv is larger than MSS of DUT
TCP_MSS_OPTIONS_08	Changed Pass Criteria	1. DUT: Move on to ESTABLISHED state 4. DUT: Send a data segment 5. TESTER: Verify that the received segment has size equal to min(transport_layer_max_send_msg_size-20, 536)	4. DUT: Send a data segment 5. TESTER: Verify that the received segment has size equal to min(transport_layer_max_send_msg_size-20, 536)
TCP_AVOIDANCE_02	Changes Synopsis	TCP MUST include an SWS avoidance algorithm in the receiver (Note:read some data of size greater or equal to MSS. TESTER follow the third step in a sequence. TESTER cause DUT to	TCP MUST include an SWS avoidance algorithm in the receiver (Note:read some data of size greater or equal to MSS. TESTER follow the third step in a sequence. TESTER cause DUT to

		receive multiple of 360 byte data until DUT read data of size greater or equal to MSS.)	receive data until DUT read data of size greater or equal to MSS.)
TCP_RETRANSMISSION_TO_04	Changed Pass Criteria	3. DUT: Send the data segment 5. DUT: Retransmit the data segment after a timeout	2. DUT: Send a SYN 4. DUT: Retransmit the SYN segment after a timeout 5. TESTER: Verify that RTO retransmission interval is increasing at a fast (at least more than linear) rate
TCP_RETRANSMISSION_TO_05	Changed Pass Criteria	2. DUT: Send a SYN 4. DUT: Retransmit the SYN segment after a timeout	2. DUT: Send a SYN 4. DUT: Retransmit the SYN segment after a timeout 5. TESTER: Verify that RTO retransmission interval is increasing at a fast (at least more than linear) rate
TCP_RETRANSMISSION_TO_08	Changed Test Procedure	3. TESTER: Do not send ACK 4. DUT: Retransmit the data segment 5. TESTER: Do not send any ACK and verify that the DUT repeatedly retransmits with increasing delays till the retransmit timeout reaches 2*MSL after which the RTO gets fixed at that value	3. DUT : Send the data segment 4. TESTER: Do not send ACK 5. DUT: Retransmit the data segment 6. TESTER: Do not send any ACK and verify that the DUT repeatedly retransmits with increasing delays till the retransmit timeout reaches 2*MSL after which the RTO gets fixed at that value
TCP_RETRANSMISSION_TO_08	Changed Pass Criteria	4. DUT: Retransmit the data segment	3. DUT : Send the data segment 5. DUT: Retransmit the data segment 6. TESTER: Verify that the DUT repeatedly retransmits with increasing delays till the retransmit timeout reaches 2*MSL after which the RTO gets fixed at that value
TCP_RETRANSMISSION_TO_09	Changed Pass Criteria	2. DUT: Send a SYN 4. DUT: Retransmit the SYN segment	4. DUT: Send only <cwnd-ini> segments
TCP_SLOWSTART_CONGESTION_01	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segments and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-	4. DUT: Send only <cwnd-ini> segments

		ini> segments	
TCP_SLOWSTART_CONGESTION_02	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Retransmit data packet	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Retransmit data packet
TCP_SLOWSTART_CONGESTION_03	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segments and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Set cwnd to 4*<cwnd-ini>, send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: Set cwnd to 8*<cwnd-ini>, send 8*<cwnd-ini> segments and wait for their ACK 12. DUT: Retransmit data packet	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: send 8*<cwnd-ini> segments and wait for their ACK 12. DUT: Retransmit data packet
TCP_SLOWSTART_CONGESTION_04	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Set cwnd to 4*<cwnd-ini>, send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: Set cwnd to 8*<cwnd-ini>, send 4 segments and wait for their ACK 12. DUT: Retransmit data	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: send 4 segments and wait for their ACK 12. DUT: Retransmit data packet

		packet	
TCP_SLOWSTART_CONGESTION_05	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Set cwnd to 4*<cwnd-ini> segment. Send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: Set ssthresh to (cwnd/2)= (2*<cwnd-ini>) and cwnd to <LW> 11. DUT: Retransmit <cwnd> segment 13. DUT: Send only <cwnd-ini> segments 15. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 17. DUT: Start congestion avoidance. Send (2*<cwnd-ini> + 1) segments because the DUT is performing Congestion Avoidance, and not (4*<cwnd-ini>) segments as would be the case with Slow Start.	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Send 4*<cwnd-ini> segments and wait for their ACK 11. DUT: Retransmit <cwnd> segment 13. DUT: Send only <cwnd-ini> segments 15. DUT: send 2*<cwnd-ini> segments and wait for their ACK 17. DUT: Start congestion avoidance. Send (2*<cwnd-ini> + 1) segments because the DUT is performing Congestion Avoidance, and not (4*<cwnd-ini>) segments as would be the case with Slow Start.
TCP_SLOWSTART_CONGESTION_06	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segment 8. DUT: Set ssthresh to 2 and cwnd to <LW> 9. DUT: Send only <cwnd> segment 13. DUT: Start congestion avoidance. Send 3 segments because the DUT is performing Congestion	4. DUT: Send only <cwnd-ini> segment 9. DUT: Send only <cwnd> segment 13. DUT: Start congestion avoidance. Send 3 segments because the DUT is performing Congestion Avoidance, and not 4 segments as would be the case with Slow Start.

		Avoidance, and not 4 segments as would be the case with Slow Start.	
TCP_SLOWSTART_CONGESTION_07	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Set cwnd to 4*<cwnd-ini> segment. Send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: Set ssthresh to 2 and cwnd to <LW> 11. DUT: Send only <cwnd> segment	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Send 4*<cwnd-ini> segments and wait for their ACK 11. DUT: Send only <cwnd> segment
TCP_SLOWSTART_CONGESTION_08	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Set cwnd to 4*<cwnd-ini> segment. Send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: Set ssthresh to (cwnd/2)=2*<cwnd-ini> and cwnd to <LW> 11. DUT: Send only <cwnd> segments 13. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 15. DUT: Start congestion avoidance by sending 2*<cwnd-ini>+1	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Send 4*<cwnd-ini> segments and wait for their ACK 11. DUT: Send only <cwnd> segments 13. DUT: send 2*<cwnd-ini> segments and wait for their ACK 15. DUT: Start congestion avoidance by sending 2*<cwnd-ini>+1 segments but not 4*<cwnd-ini>

		segments but not 4*<cwnd-ini>	
TCP_SLOWSTART_CONGESTION_09	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 9. DUT: Set cwnd to 4 segment. Send 4 segments and wait for their ACK 11. DUT: Set ssthresh = (cwnd/2)=2 and cwnd to <LW> 12. DUT: Send only <cwnd> segment 15. DUT: Set cwnd to 2 (2*SMALL_MAX_SEG_SIZE in bytes) send 2 segments and wait for their ACK 17. DUT: Start congestion avoidance. Calculate cwnd (in bytes) as cwnd + { (segsize * segsize) / cwnd) } * no_of_ACK_received i.e. (2*SMALL_MAX_SEG_SIZE) + {(SMALL_MAX_SEG_SIZE* SMALL_MAX_SEG_SIZE)/(2* SMALL_MAX_SEG_SIZE)}*2 = (2*SMALL_MAX_SEG_SIZE) +SMALL_MAX_SEG_SIZE (bytes)= 3 segments 18. DUT: Send 3 segments	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 9. DUT: Send 4 segments and wait for their ACK 12. DUT: Send only <cwnd> segment 15. DUT: send 2 segments and wait for their ACK 18. DUT: Send 3 segments
TCP_SLOWSTART_CONGESTION_10	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 9. DUT: Send 4 segments and wait for

		2*<cwnd-ini> segments and wait for their ACK 9. DUT: Set cwnd to 4 segment. Send 4 segments and wait for their ACK 11. DUT: Set ssthresh = (cwnd/2)=2 and cwnd to <LW> 12. DUT: Send only <cwnd> segment 16. DUT: Start congestion avoidance, send 3 segments 18. DUT: Send 4 segments 20. DUT: Send 5 segments	their ACK 12. DUT: Send only <cwnd> segment 16. DUT: Start congestion avoidance, send 3 segments 18. DUT: Send 4 segments 20. DUT: Send 5 segments
TCP_FAST_RETRANSMIT_01	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Set cwnd to 4*<cwnd-ini>, send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: On receiving the 3rd duplicate ACK, retransmit the ACKd segment 11. TESTER: Check that DUT retransmits the correct packet without waiting for retransmission timer to expire	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments 10. DUT: Retransmit the segment indicated by the last received ACK 12. DUT: Send 4 segments
TCP_FAST_RETRANSMIT_02	Changed Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. DUT: Send only <cwnd-ini> segments 6. DUT: Set cwnd to 2*<cwnd-ini> and send 2*<cwnd-ini> segments 10. DUT: On receivng the 3 duplicate ACKs,	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments 10. DUT: Retransmit the segment indicated by the last received ACK 12. DUT: Send 4 segments 14. DUT: Send 5 segments

		set ssthresh to one-half the current cwnd. Retransmit the segment indicated by the last received ACK and set cwnd to ssthresh plus 3 (in segments) 12. DUT: Send 4 segments 14. DUT: Send 5 segments	
TCP_HOST_SPEC_02	Changed Test Procedure	10. TESTER: Check whether: DF bit is not set (IPv4) OR Fragmentation Header is not present (IPv6) OR length does not exceed (SMALL_PATH_MTU - 40) if: DF bit is set (IPv4) OR: Fragmentation Header is present (IPv6)	10. TESTER: Check whether: A) DF bit is not set (IPv4) OR Fragmentation Header is not present (IPv6) OR B) length does not exceed (SMALL_PATH_MTU - 40) if: DF bit is set (IPv4) OR: Fragmentation Header is present (IPv6)
TCP_HOST_SPEC_03	Changed Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Send an ICMP packet with code field set to DATAGRAM_TOO_BIG and Next-Hop MTU field set to SMALL_PATH_MTU. 8. DUT: Send data packet of size less than (SMALL_PATH_MTU - 40) 9. TESTER: Send ACK for the received data 10. DUT: Continue sending data of size	1. Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to send large data 3. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 4. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0,

		(SMALL_PATH_MTU - 40) 11. TESTER: Send ACK for the received data, and observe for 5 minutes that DUT never sends data more than (SMALL_PATH_MTU - 40) bytes	DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 5. DUT: Send data packet of size less than (SMALL_PATH_MTU - 40) 6. TESTER: Send ACK for the received data 7. DUT: Continue sending data of size (SMALL_PATH_MTU - 40) 8. TESTER: Send ACK for the received data, and observe for 5 minutes that DUT never sends data more than (SMALL_PATH_MTU - 40) bytes
TCP_HOST_SPEC_04	Changed Test Procedure	7. TESTER: Send an ICMP packet with code field set to DATAGRAM_TOO_BIG and Next-Hop MTU field set to SMALL_PATH_MTU.	7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU
TCP_HOST_SPEC_09	Changed Test Procedure	7. TESTER: Send ICMP packet with code field set to DATAGRAM_TOO_BIG with - next hop MTU field set to 0	7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to 0 OR

			TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to 0
TCP_HOST_SPEC_10	Changed Test Procedure	7. TESTER: Send ICMP packet with code field set to DATAGRAM_TOO_BIG with - next hop MTU field set to 0	7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to 0 OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to 0
TCP_LAYER_ACTIONS_01	Changed Test Procedure	7. TESTER: Send an ICMP packet with code field set to DATAGRAM_TOO_BIG and Next-Hop MTU field set to SMALL_PATH_MTU.	7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU
TCP_LAYER_ACTIONS_02	Changed Test	4. TESTER: Send an ICMP packet with code field set to	4. TESTER: Send an ICMP packet with (IPv4)

	Procedure	DATAGRAM_TOO_BIG and Next-Hop MTU field set to SMALL_PATH_MTU.	- type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU
TCP_LAYER_ACTIONS_03	Changed Test Procedure	7. TESTER: Send an ICMP packet with code field set to DATAGRAM_TOO_BIG and Next-Hop MTU field set to SMALL_PATH_MTU.	7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU
TCP_LAYER_ACTIONS_05	Changed Test Procedure	10. TESTER: Send an ICMP packet with code field set to DATAGRAM_TOO_BIG and Next-Hop MTU field set to SMALL_PATH_MTU.	10. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6

			packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU
TCP_LAYER_ACTIONS_07	Changed Test Procedure	7. TESTER: Send an ICMP packet with code field set to DATAGRAM_TOO_BIG and Next-Hop MTU field set to SMALL_PATH_MTU.	7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU
TCP_IMPROVED_WINDOWING_02	Changed Test Procedure and Pass Criteria	5. DUT: Calculate usable window and it'll be less than 25% of offered window It'll not send any new data segments	5. DUT: Usable window is smaller than 25%, DUT does not send a new data segment
TCP_IMPROVED_WINDOWING_04	Changed Test Procedure and Pass Criteria	5. DUT: Calculate usable window and it'll be of Maximum Segment Size It'll send a new data segment	5. DUT: Send a new data segment as the usable window is bigger than the Maximum Segment Size
TCP_IMPROVED_WINDOWING_05	Changed Test Procedure and Pass Criteria	5. DUT: Calculate usable window and it'll be less than Maximum Segment Size It'll not send a new data segment	5. DUT: Does not send a new data segment as the usable window is smaller than the Maximum Segment Size It'll not send a new data segment
TCP_CONNECTION_ESTAB_12	Changed Test Procedure	2. TESTER: Send a ACK packet on a nother <wnp>	2. TESTER: Send a Keep-Alive ACK packet with SEG.SEQ = SND.NXT - 1 (Note: on a quiet

			connection SND.NXT = RCV.NXT) so that the Sequence-Number is out of the receive window
TCP_SEQUENCE_04	Changed Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state. 2. TESTER: Send a TCP packet with Sequence Number equal to <SeqMaxVal>. 3. DUT: Send an ACK with Ack Number equal to 0.	1. TESTER: Send a SYN to DUT at <wnp> with Initial Sequence Number equal to <SeqMaxVal>. 2. DUT: Send SYN,ACK with Ack Number equal to 0.
TCP_CONTROL_FLAGS_06	Test Case deleted		
ARP_01	Changed Test Procedure	3. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	3. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_02	Changed Test Procedure	3. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	3. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_03	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_04	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_05	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_06	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_07	Changed Test Procedure	2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_08	Changed Test Procedure	2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_09	Changed Test	2. DUT CONFIGURE: Configure DUT to send an	2. DUT CONFIGURE: Configure DUT to send a

	Procedure	ICMP Echo Request Message…	UDP Request Message…
ARP_10	Changed Test Procedure	2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_11	Changed Test Procedure	2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_12	Changed Test Procedure	2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_13	Changed Test Procedure	2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_14	Changed Test Procedure	2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_15	Changed Test Procedure	2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_22	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_25	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_28	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_31	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_32	Changed Test Procedure	5. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	5. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_33	Changed Test	5. DUT CONFIGURE: Configure DUT to send an	5. DUT CONFIGURE: Configure DUT to send a

	Procedure	ICMP Echo Request Message…	UDP Request Message…
ARP_34	Changed Test Procedure	5. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	5. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_35	Changed Test Procedure	5. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	5. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_38	Changed Test Procedure	4. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_39	Changed Test Procedure	2. and 7. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. and 7. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
ARP_40	Changed Test Procedure	2. and 7. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message…	2. and 7. DUT CONFIGURE: Configure DUT to send a UDP Request Message…
TCP_OUT_OF_ORDER_06 TCP_OUT_OF_ORDER_07 TCP_OUT_OF_ORDER_09 TCP_MD5_01 - TCP_MD5_11 TCP_CUBIC_01 - TCP_CUBIC_06 TCP_IMPROVED_WINDOWING_06 TCP_IMPROVED_WINDOWING_07 TCP_CONNECTION_ESTAB_04 TCP_CONNECTION_ESTAB_05 TCP_CONNECTION_ESTAB_06 TCP_CONNECTION_ESTAB_08 - TCP_CONNECTION_ESTAB_11 TCP_HEADER_03 TCP_SEQUENCE_06 TCP_ACKNOWLEDGEMENT_01 TCP_CONTROL_FLAGS_02	Invalid Test Case		Deleted Test Case

TCP_CONTROL_FLAGS_03 **TCP_CONTROL_FLAGS_04** **TCP_CONTROL_FLAGS_06** **TCP_CONTROL_FLAGS_09** **TCP_CONTROL_FLAGS_10** **TCP_CONTROL_FLAGS_11** **TCP_CONTROL_FLAGS_12** **TCP_WINDOW_SIZE_01** **TCP_WINDOW_SIZE_02** **TCP_WINDOW_SIZE_03**			
SWITCH_VLAN_01	Changed Test Procedure	1. Create the test topology as shown in the Setup. 2. On each test port, iterate through each VLAN, source MAC, and destination MAC, and transmit Echo Request packets for each combination. 3. Verify that for each valid combination of destination MAC and VLAN, an Echo Reply is received on the test device port that sent the Echo-Request packet. 4. Verify that for each invalid VLAN or each invalid MAC/VLAN combination, the Echo-Request packet is dropped on the receiving ECU Switch Port. 5. Verify that for each combination of invalid destination MAC and valid VLAN, the Echo-Request is forwarded to each test device port belonging to that VLAN except the source port of Echo-Request.	1. Create the test topology as shown in the Setup. 2. On each test port, iterate through each source MAC and destination MAC address and transmit untagged Echo Request packets for each combination. 3. Verify that for each valid destination MAC, the Echo Request is received from each external DUT port except from the port the Echo Request packet has been sent to. 4. Verify that for each valid combination of destination MAC and VID, an Echo Reply from each internal port is received from the DUT port the Echo Request packet has been sent to. 5. Verify that for each invalid VLAN or each invalid MAC/VLAN combination, no Echo Request or Echo Reply has been received on any port. 6. On each test port, iterate through each VLAN ID, source MAC and destination MAC address and transmit single-tagged Echo Request packets for each combination. 7. Verify that for each valid destination MAC, the Echo

			Request is received from each external DUT port except from the port the Echo Request packet has been sent to. 8. Verify that for each valid combination of destination MAC and VID, an Echo Reply from each internal port is received from the DUT port the Echo Request packet has been sent to. 9. Verify that for each invalid VLAN or each invalid MAC/VLAN combination, no Echo Request or Echo Reply has been received on any port. 10. On each test port, iterate through each VLAN ID (as outer VLAN tag), source MAC and destination MAC address and transmit double-tagged Echo Request packets for each combination. 11. Verify that for each valid destination MAC, the Echo Request is received from each external DUT port except from the port the Echo Request packet has been sent to. 12. Verify that for each valid combination of destination MAC and VID, an Echo Reply from each internal port is received from the DUT port the Echo Request packet has been sent to. 13. Verify that for each invalid VLAN or each invalid MAC/VLAN combination, no Echo Request or Echo Reply has been received on any port.
	Changed	The DUT traffic counters	The DUT traffic counters

	Pass Criteria	match the expected test device traffic counters for each combination of VLAN, Source MAC and Destination MAC.	match the expected test device traffic counters for each combination of VLAN, Source MAC and Destination MAC. The traffic received by the test device is tagged or untagged in accordance to the VLAN configuration provided by the manufacturer.
SWITCH_VLAN_02	Changed Test Procedure	1. Create the test topology as shown in the Setup. 2. On each test port, iterate through each VLAN, source MAC, source IP and destination IP, and transmit ARP Request packets for each combination. 3. Verify that for each valid combination of destination IP and VLAN, an ARP Reply is received on the test device port that sent the ARP Request packet. 4. Verify that for each invalid VLAN, the ARP Request packet is dropped on the receiving ECU Switch Port. 5. Verify that for each combination of invalid destination IP and valid VLAN, the ARP Request is forwarded to each test device port belonging to that VLAN except the source port of ARP Request.	1. Create the test topology as shown in the Setup. 2. On each test port, iterate through each source MAC, source IP and destination IP and transmit untagged broadcast ARP Request packets for each combination. 3. Verify that for each valid destination IP, the ARP Request is received from each external DUT port except from the port the ARP Request packet has been sent to. 4. Verify that for each valid combination of destination IP and VID, an ARP Reply from each internal port is received from the DUT port the ARP Request packet has been sent to. 5. Verify that for each invalid VLAN, no ARP Request or ARP Reply has been received on any port. 6. On each test port, iterate through each VLAN, source MAC, source IP and destination IP address and transmit single-tagged broadcast ARP Request packets for each combination. 7. Verify that for each valid destination MAC, the ARP

			Request is received from each external DUT port except from the port the ARP Request packet has been sent to. 8. Verify that for each valid combination of destination MAC and VID, an ARP Reply from each internal port is received from the DUT port the ARP Request packet has been sent to. 9. Verify that for each invalid VLAN, no ARP Request or ARP Reply has been received on any port. 10. On each test port, iterate through each VLAN (as outer VLAN tag), source MAC, source IP and destination IP address and transmit double-tagged broadcast ARP Request packets for each combination. 11. Verify that for each valid destination MAC, the ARP Request is received from each external DUT port except from the port the ARP Request packet has been sent to. 12. Verify that for each valid combination of destination MAC and VID, an ARP Reply from each internal port is received from the DUT port the ARP Request packet has been sent to. 13. Verify that for each invalid VLAN, no ARP Request or ARP Reply has been received on any port.
	Changed Pass Criteria	The DUT traffic counters match the expected test device traffic counters for	The DUT traffic counters match the expected test device traffic counters for

		each combination of VLAN, Source MAC, Source IP and Destination IP.	each combination of VLAN, Source MAC, Source IP and Destination IP. The traffic received by the test device is tagged or untagged in accordance to the VLAN configuration provided by the manufacturer.
SWITCH_VLAN_03	Changed Test Procedure	1. Create the test topology as shown in the Setup. 2. Configure the test ports acting as Hosts to send join for the destination multicast groups. 3. Verify that the test ports acting as Hosts are receiving queries and the test port acting as Querier is receiving join/leave messages. 4. On test port acting as IGMP Querier, iterate through each VLAN, source MAC, source IP and destination multicast IP (destination MAC is derived from destination IP), and transmit multicast IP traffic for each combination. 5. Verify that for each valid combination of source IP, destination IP and VLAN, the IP multicast traffic is correctly replicated only on those test ports that have receivers for the IGMP Group (identified by destination multicast IP). Note: The ECU switch can be a pure/hybrid L2 switch and in that case multicast forwarding	1. Create the test topology as shown in the Setup. 2. Configure the test ports acting as Hosts to send join for the destination multicast groups. 3. Verify that the test ports acting as Hosts are receiving queries and the test port acting as Querier is receiving join/leave messages. 4. On each test port acting as IGMP Querier, iterate through each source MAC, source IP and destination multicast IP (destination MAC is derived from destination IP), and transmit untagged multicast IP traffic for each combination. 5. Verify that for each valid combination of source IP and destination IP, the IP multicast traffic is correctly replicated only on those test ports that have receivers for the IGMP Group (identified by destination multicast IP). Note: The ECU switch can be a pure/hybrid L2 switch and in that case multicast forwarding will be enabled either by IGMP snooping or static multicast forwarding entries. 6. On each test port acting

		will be enabled either by IGMP snooping or static multicast forwarding entries. 6. Verify that for each invalid VLAN, the IP multicast packet is dropped on the receiving ECU Switch Port. 7. Verify that for each combination of invalid destination multicast IP and valid VLAN, the IP traffic is not forwarded.	as IGMP Querier, iterate through each VID, source MAC, source IP and destination multicast IP (destination MAC is derived from destination IP), and transmit single-tagged multicast IP traffic for each combination. 7. Verify that for each valid combination of source IP and destination IP, the IP multicast traffic is correctly replicated only on those test ports that have receivers for the IGMP Group (identified by destination multicast IP). 8. On each test port acting as IGMP Querier, iterate through each VID (as outer VLAN tag), source MAC, source IP and destination multicast IP (destination MAC is derived from destination IP), and transmit double-tagged multicast IP traffic for each combination. 9. Verify that for each valid combination of source IP and destination IP, the IP multicast traffic is correctly replicated only on those test ports that have receivers for the IGMP Group (identified by destination multicast IP).
	Changed Pass Criteria	DUT traffic counters match the expected test device traffic counters for each combination of VLAN, Source MAC, Source IP and Destination IP.	DUT traffic counters match the expected test device traffic counters for each combination of VLAN, Source MAC, Source IP and Destination IP. The traffic received by the test device is tagged or untagged in accordance to

			the VLAN configuration provided by the manufacturer.
SWITCH_QOS_01	Deprecated		Removed as the test case is actually not testing QoS behavior, but only verifying the VLAN tag, which is also done in test cases SWITCH_VLAN_01, SWITCH_VLAN_02 and SWITCH_VLAN_03.
SWITCH_QOS_02	Deprecated		Removed due to referenced TC11 test case
SWITCH_QOS_03	Deprecated		Removed due to referenced TC11 test case
SWITCH_QOS_04	Deprecated		Removed due to referenced TC11 test case
SWITCH_QOS_05	Deprecated		Removed due to referenced TC11 test case
8021X__EAPoL_Authentication_positive	New test		New test is necessary as the according test cases in the TC11 test specification only cover switch dependent features.
8021X__EAPoL_Authentication_negative	New test		New test is necessary as the according test cases in the TC11 test specification only cover switch dependent features.
8021AE_MACsec_frames_forwarded	New test		New test is necessary as the according test cases in the TC11 test specification only cover switch dependent features.
SWITCH_GENERAL_01	Deprecated		Removed due to referenced TC11 test case
SWITCH_GENERAL_02	Deprecated		Removed due to referenced TC11 test case
SWITCH_GENERAL_03	Deprecated		Removed due to referenced TC11 test case
SWITCH_GENERAL_04	Deprecated		Removed due to referenced TC11 test case
Port_Disabling	Renamed	SWITCH_GENERAL_05	Port_Disabling
	Changed Test Procedure	The test system continuously transmits unicast and broadcast	1. Receive and monitor all traffic from all ports 2. From the test station

		frames with minimum (42 bytes) and maximum regular (1500 bytes) payload length on all connected ports of the DUT. The test shall run until at least 100 frames were dropped at each disabled port.	send traffic to a port A addressed to a port B 3. Observe if test station receives any traffic from port B 4. Through any vendor specified means, disable port B 5. Observe if link at port B is still up when the port is disabled 6. From the test station send traffic to port A addressed to port B 7. Observe if test station receives any traffic from port B 8. From the test station send traffic to port B addressed to port A and with a source MAC address X that has not been learnt on any DUT port yet. 9. Observe if test station receives any traffic from port A that has been sent to port B in step 8. 10. If flooding of frames with unknown destination MAC addresses is enabled, then from the test station send traffic to port A addressed to the MAC address X. 11. If flooding of frames with unknown destination MAC addresses is enabled, observe if test station receives on any port any traffic that has been sent to port A in step 10. 12. Repeat the test steps 1-11 for all ports that can be disabled as port B.
	Changed Pass Criteria	Verify that all frames received on disabled ports ($P_{disabled}$) are not forwarded	3. Test Station receives traffic from port B 5. Link at port B is still up

		to any other port. Verify that no frames are transmitted on a disabled port by the DUT.	7. Test Station receives no traffic from port B 9. Test Station receives no traffic from port A 11. Test Station receives traffic from all ports
Address_Learning_read_ARL_table	New test		New test is necessary as the according test cases in the TC11 test specification is not applicable for TC8 ECU testing
Address_Learning_write_ARL_table	New test		New test is necessary as the according test cases in the TC11 test specification is not applicable for TC8 ECU testing
Address_Learning_with_untagged_frames_at_external_ports	New test		New test is necessary as the according test cases in the TC11 test specification is not applicable for TC8 ECU testing
Address_Learning_with_tagged_frames_at_external_ports	New test		New test is necessary as the according test cases in the TC11 test specification is not applicable for TC8 ECU testing
Address_Learning_at_internal_ports	New test		New test is necessary as the according test cases in the TC11 test specification is not applicable for TC8 ECU testing
SWITCH_INGRESS_01	Deprecated		Removed due to referenced TC11 test case
SWITCH_INGRESS_02	Deprecated		Removed due to referenced TC11 test case
Policing_Information_external_ports	New test		New test is necessary as the according test cases in the TC11 test specification is not applicable for TC8 ECU testing
Policing_Information_internal_ports	New test		New test is necessary as the according test cases in the TC11 test specification is not applicable for TC8 ECU testing
SWITCH_DIAGNOSTICS_01	Deprecated		Removed due to referenced TC11 test case
SWITCH_DIAGNOSTICS_02	Deprecated		Duplicate to

		OABR_SIGNAL_01 and OABR_SIGNAL _02
SWITCH_DIAGNOSTICS_03	Deprecated	Duplicate to OABR_CABLE_01 and OABR_CABLE_02

1.5 Feedback

Any feedback for correcting, improving or adding new content is welcome. We encourage to bring forward this feedback in the regular meetings of the OPEN Alliance TC8. In case you are not an OPEN Alliance member you can relay questions or feedback through the following contacts:

Company	Name	Email address
C&S Group		eth-testing@cs-group.de
IXIA		tc8feedback@ixiacom.com
NXP	Rajeev Roy	rajeev.roy@nxp.com
Ruetz System Solutions GmbH		support@ruetz-system-solutions.com
Spirent Communications	Dirk Tepelmann Fares Mokrani Stephan Pietsch	dirk.tepelmann@spirent.com fares.mokrani@spirent.com stephan.pietsch@spirent.com
TÜV NORD	Heiko Ehrich	hehrich@tuev-nord.de

1.6 References

[1] OA_100BASE-T1 Interoperability Test Suite 1v0

[2] IEEE Std 802.3bw™ – 2015 Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1)..

[3] IEEE 100BASE-T1 Physical Media Attachment Test Suite Version 1.0

[4] IEEE 100BASE-T1 Definitions for Communication Channel, Version 1.0 .

[5] IEEE 100BASE-T1 EMC Test Specification for Transceivers Version 1v0

1.7

2 Test Scope Layer 1 of Automotive Ethernet

2.1 Interoperability Tests

2.1.1 General

The following test specifications are adapted from [1] to fit the general requirements of an DUT.

2.1.2 Link-up time

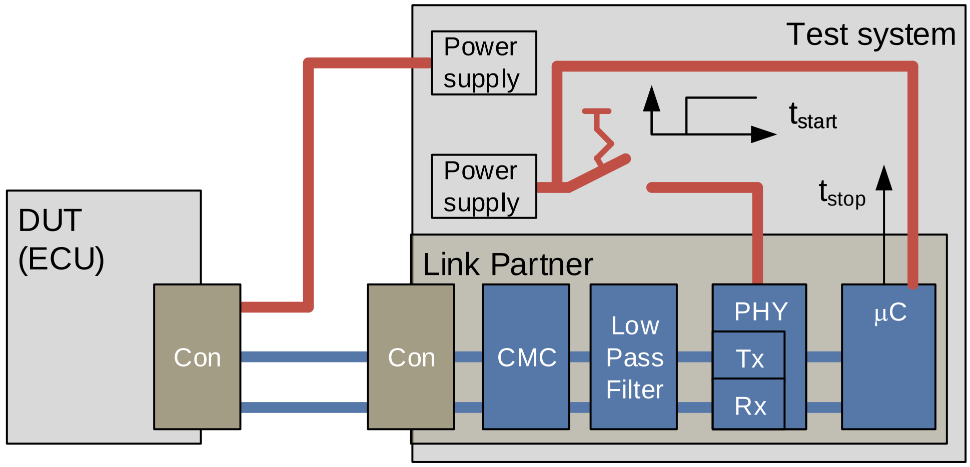
(based on 100BASET1_ IOP_21 and 100BASET1_ IOP_22 of [1])

3 test cycles:

- Power on Link Partner
- Power on DUT
- Wake up DUT

OABR_LINKUP_01: Link-up time - Trigger: Power on Link Partner

Synopsis	Shall ensure that the link is established within a given time without a high time variation.
Prerequisites	1. The DUT is connected to a stable power supply. 2. The DUT must be operated in normal mode. 3. The Test System provides special awake conditions for the DUT such as a wakeup line or network management CAN messages if necessary. 4. If the DUT contains a switch all links have to be tested separately. 5. The mean start up time of the Link Partner is available: $\bar{t}_{ready}$
Test setup	The DUT must be connected to the Link Partner with opposite master/slave configuration. The polarity of the communication channel must be correct. The power supplies are controlled by the test system.

<table>
<tr><td></td><td>
</td></tr>
<tr><td>Test procedure</td><td>1. DUT shall be active and ready to build up link.
Repeat Step 2 to Step 5 n=100times:

2. Power on Link Partner. $t_{start}=t_{PowerOnLinkPartner}$
3. Polling of Link Partner status register. If link_control= active link: $t_{stop}=t_{ActiveLink}$
4. Calculate the time t_{up} between power on and link up: $t_{up}= t_{stop} - t_{start}$
5. Power off Link Partner.
End of Repeat

6. Calculate as follows:

$$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_{up}(i)$$

$$\sigma t = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(t_{up}(i) - \bar{t})^2}$$

$$\mathrm{t_{min}} = \min(t_{up}(i))$$

$$\mathrm{t_{max}} = \max(t_{up}(i))$$
</td></tr>
<tr><td>Pass criteria</td><td>

$$\sigma t \leq 50\ \mathrm{ms}$$

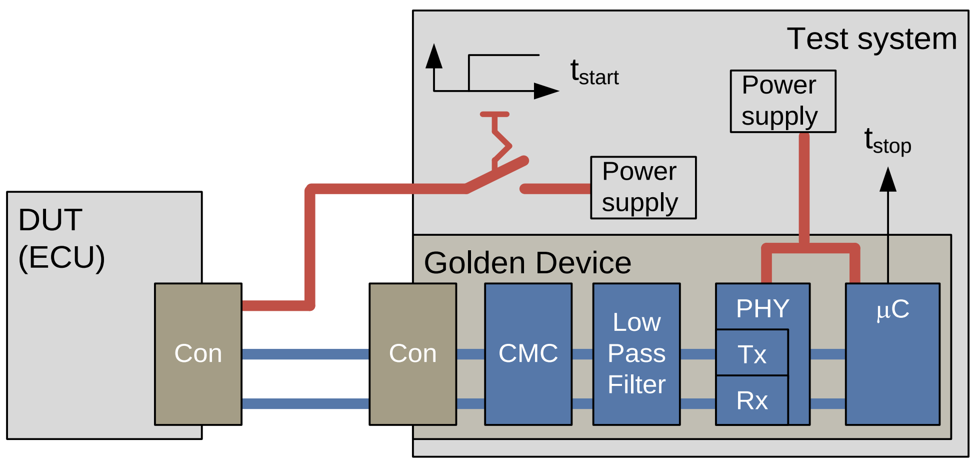
$$\mathrm{t_{min}} > 10\ \mathrm{ms} + \bar{t}_{ready}$$

$$\mathrm{t_{max}} < 100\ \mathrm{ms} + \bar{t}_{ready}$$
</td></tr>
<tr><td>Notes</td><td>This test has to be performed for each port of the DUT, if it has a switch inside.

In dependency of the design of the link partner, the Test system may switch also the</td></tr>
</table>

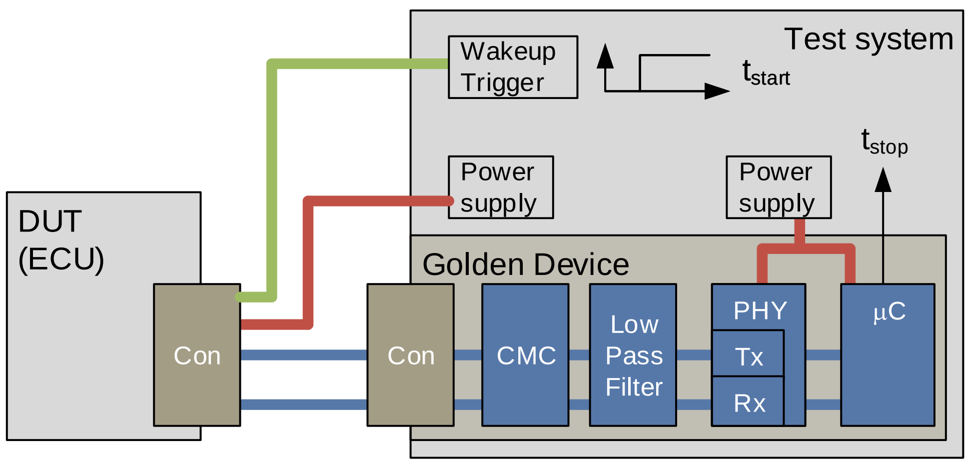
	power supply of the µC together with the power supply of the PHY.

OABR_LINKUP_02: Link-up time - Trigger: Power on DUT

Synopsis	Shall ensure that the link is established within a given time without a high time variation.
Prerequisites	1. The Link Partner is connected to a stable power supply. 2. The Test System provides special awake conditions for the DUT such as a wakeup line or network management CAN messages if necessary. 3. The manufacturer has to provide the mean start up time of the DUT: $\bar{t}_{ready1}$
Test setup	The DUT must be connected to the Link Partner with opposite master/slave configuration. The polarity of the communication channel must be correct. The power supplies are controlled by the test system. 
Test procedure	1. Link Partner shall be active and ready to build up link. Repeat Step 2 to Step 5 n=100times: 2. Power on DUT. $t_{start}=t_{PowerOnDUT}$ 3. Polling of Link Partner status register. If link_control= active link: $t_{stop}=t_{ActiveLink}$ 4. Calculate the time t_{up} between power on and link up: $t_{up}= t_{stop} - t_{start}$ 5. Power off DUT. End of Repeat 6. Calculate as follows: $$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_{up}(i)$$

	$$\sigma t = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(t_{up}(i) - \bar{t})^2}$$ $$\mathrm{t_{min}} = \min(t_{up}(i))$$ $$\mathrm{t_{max}} = \max(t_{up}(i))$$
Pass criteria	$$\sigma t \leq 50\ \mathrm{ms}$$ $$\mathrm{t_{min}} > 10\ \mathrm{ms} + \bar{t}_{ready1}$$ $$\mathrm{t_{max}} < 100\ ms + \bar{t}_{ready1}$$
Notes	This test has to be performed for each port of the DUT, if it has a switch inside.

OABR_LINKUP_03: Link-up time - Trigger: Wake up DUT

<table>
<tr><td>Synopsis</td><td>Shall ensure that the link is established within a given time without a high time variation.</td></tr>
<tr><td>Prerequisites</td><td>1. The DUT and the Link Partner are connected to a stable power supply.
2. The DUT must be operated in normal mode.
3. Wake up message is necessary. The Test System provides special awake conditions for the DUT such as a wakeup line or network management CAN messages.
4. The manufacturer has to provide the value $I_{sleep.}$
5. The manufacturer has to provide the mean wake up time of the DUT: $\bar{t}_{ready2}$</td></tr>
<tr><td>Test setup</td><td>The DUT must be connected to the Link Partner with opposite master/slave configuration. The polarity of the communication channel must be correct. The power supplies are controlled by the test system.

</td></tr>
<tr><td>Test procedure</td><td>1. DUT shall be in sleep mode and Link Partner shall be active and ready to build up link.
Repeat Step 2 to Step 6 n=100times:

2. Turn on Wake up signal for DUT.
3. $t_{WakeUpDUT}$ if $I_{DUT} > I_{sleep}$, $t_{start}=t_{WakeUpDUT}$
4. Polling of Link Partner status register. If link_control= active link: $t_{stop}=t_{ActiveLink}$
5. Calculate the time t_{up} between wake up and link up: $t_{up}= t_{stop} - t_{start}$
6. Switch DUT to sleep mode.
End of Repeat

7. Calculate as follows:
$$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_{up}(i)$$</td></tr>
</table>

	$$\sigma t = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(t_{up}(i) - \bar{t})^2}$$ $$\mathrm{t_{min}} = \min(t_{up}(i))$$ $$\mathrm{t_{max}} = \max(t_{up}(i))$$
Pass criteria	$$\sigma t \leq 50\ \mathrm{ms}$$ $$\mathrm{t_{min}} > 10\ \mathrm{ms} + \bar{t}_{ready2}$$ $$\mathrm{t_{max}} < 100\ ms + \bar{t}_{ready2}$$
Notes	This test has to be performed for each port of the DUT, if it has a switch inside.

2.1.3 Signal Quality

OABR_SIGNAL_01: Indicated signal quality for channel with decreasing quality

(based on 100BASET1_ IOP_24a of [1])

Synopsis	Shall ensure that the DUT's indicated signal quality decreases for a channel with decreasing channel quality and that there is coherence between the SQI indicated values on the DUT and the respective artificial noise injection.
Prerequisites	1. The DUT and the Link Partner are connected to a stable power supply. 2. The DUT must be operated in normal mode. 3. The Test system allows varying and determining the quality of the communication channel that connects the DUT and Link Partner. 4. DUT must be able to monitor the signal quality indicated by the PHY. The information of the signal quality can be provided by an applicative message. To be able to obtain the DUT information of the signal quality with the respective applied channel degradation step, an additional communication channel like CAN should be available.
Test setup	See chapter 7.3 Artificial degradation of channel quality of [1].
Test procedure	See Test procedure of 100BASET1_IOP_24a of [1]
Pass criteria	See Pass criteria of 100BASET1_ IOP_24a of [1]
Notes	This test has to be performed for each port of the DUT, if it has a switch inside.

OABR_SIGNAL_02: Indicated signal quality for channel with increasing quality

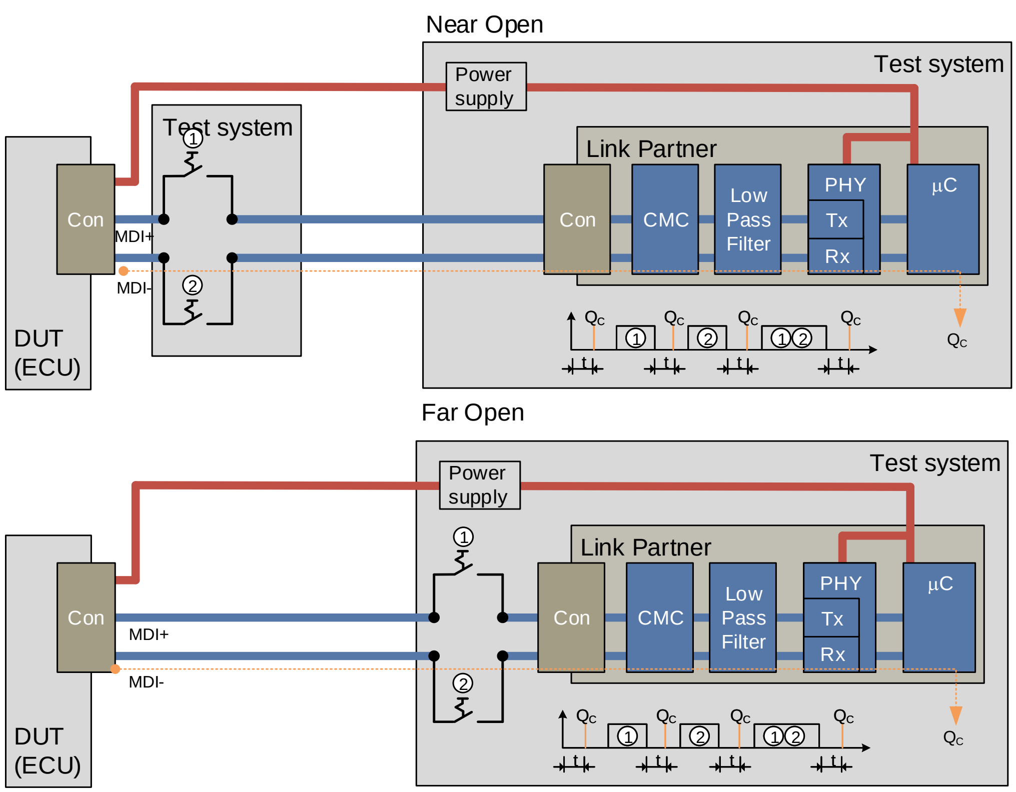
(based on 100BASET1_ IOP_24b of [1])

Synopsis	Shall ensure that the DUT's indicated signal quality increases for a channel with increasing channel quality and that there is coherence between the SQI indicated values on the DUT and the respective artificial noise injection.
Prerequisites	1. The DUT and the Link Partner are connected to a stable power supply. 2. The DUT must be operated in normal mode. 3. The Test system allows varying and determining the quality of the communication channel that connects the DUT and Link Partner. 4. DUT must be able to monitor the signal quality indicated by the PHY. The information of the signal quality can be provided by an applicative message. To be able to obtain the DUT information of the signal quality with the respective applied channel degradation step, an additional communication channel like CAN should be available.
Test setup	See chapter 7.3 Artificial degradation of channel quality of [1].
Test procedure	See Test procedure of 100BASET1_ IOP_24b of [1]
Pass criteria	See Pass criteria of 100BASET1_ IOP_24b of [1]
Notes	This test has to be performed for each port of the DUT, if it has a switch inside.

2.1.4 Cable diagnostics

OABR_CABLE_01: Cable diagnostics for near and far end open

(based on 100BASET1_ IOP_32 of [1])

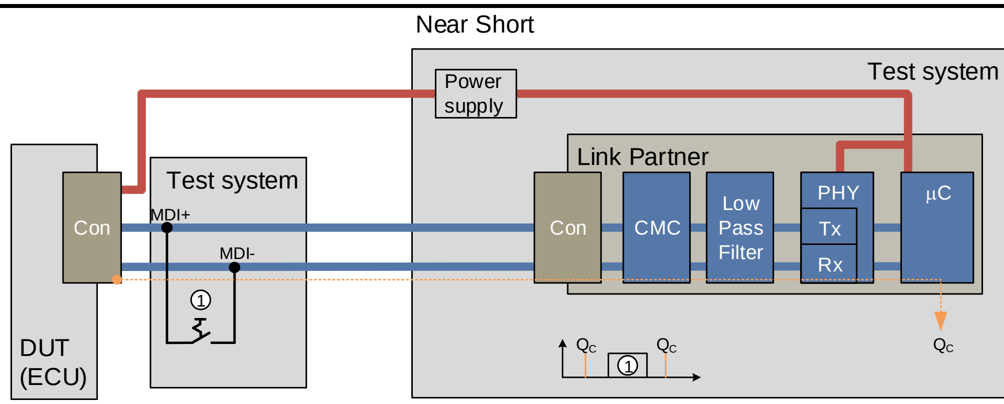
Synopsis	Shall ensure that the DUT’s cable diagnostic reliably detects an open of one or both of the bus lines. The test shall be performed for both a near end open at the connector of the DUT, and for a far end open at the connector of the Link Partner.
Prerequisites	1. The channel should be terminated properly. 2. The DUT must be capable to start cable diagnostic of its PHY 3. The DUT must be able to detect any cable errors. This means the DUT has to provide the possibility to trigger the cable diagnostic feature. The result of the DUT’s cable diagnostic can be provided by an applicative Ethernet message, an UDS communication or another communication channel like CAN.
Test setup	
Test procedure	The following steps shall be applied to test near and far end open cable diagnostics 1. The DUT cable diagnostic feature is triggered. The DUT cable diagnostics has to be executed within t_{error}. 2. The test system creates acable error for a defined time t_{error}. 3. After the wait time t the test system reads out all identified cable errors Q_C from the

Restriction Level:

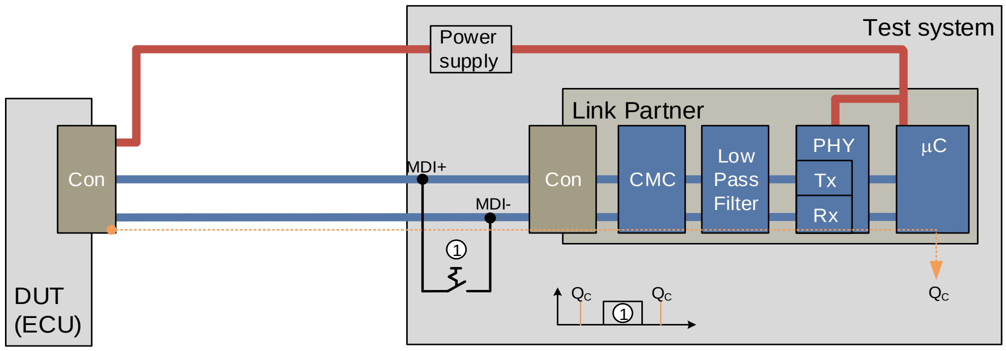
	DUT. 4. Repeat step 1 to 3 for all error combinations (alternately MDI+ and/or MDI- are open). For additional information regarding the test instances, please refer to test instances Table of 100BASET1_IOP_32 of [1].
Pass criteria	Each test iteration shall be classified as passed, if the DUT reports all expected cable errors.
Test iterations	5 times.
Notes	For additional information regarding the near and far end open, please refer to Notes of 100BASET1_ IOP_32 of [1]. The results shall be reported for each 100BASE-T1 port available in the DUT.

OABR_CABLE_02: Cable diagnostics for near and far end short

(based on 100BASET1_ IOP_33 of [1])

Synopsis	Shall ensure that the DUT's cable diagnostic reliably detects a short of the bus lines. The test shall be performed for both a near end short at the connector of the DUT, and for a far end short at the connector of the Link Partner.
Prerequisites	1. The channel should be terminated properly. 2. The DUT must be capable to start cable diagnostic of its PHY. 3. The DUT must be able to detect any cable errors. This means the DUT has to provide the possibility to trigger the cable diagnostic feature. The result of the DUT's cable diagnostic can be provided by an applicative Ethernet message, an UDS communication or another communication channel like CAN.
Test setup	

<table>
<tr><td></td><td>Far Short



DUT is connected to a properly terminated link partner. The bus wires are connected via a <= 1 Ohm resistor during following error situations:

- SHORT between both bus wires, far and near end.
- SHORT of both conductors to ground (GND) , far and near end.
- SHORT of both conductors to supply line (VBAT) , far and near end.

Please note that in the figures above there is presented only the 1st error situation (SHORT between both bus wires, far and near end). The 2nd and 3rd error situation, where the two wires MDI+ and MDI- are additionally connected to GND resp. VBAT, are not presented here.</td></tr>
<tr><td>Test procedure</td><td>The following steps shall be applied to test near and far end short cable diagnostics

1. The DUT cable diagnostic feature is triggered. The DUT cable diagnostics has to be executed within t_{error}.
2. The test system creates a cable error for a defined time t_{error}.
3. After the wait time t the test system requests all identified cable errors Q_C from the DUT.
4. Repeat step 1 to 3 for all error combinations. For additional information regarding the test instances, please refer to test instances Table of 100BASET1_IOP_33 of [1].</td></tr>
<tr><td>Pass criteria</td><td>Each test iteration shall be classified as passed, if the DUT reports all expected cable errors.</td></tr>
<tr><td>Test iterations</td><td>5 times.</td></tr>
<tr><td>Notes</td><td>For additional information regarding the near and far end short, please refer to Notes of 100BASET1_ IOP_33 of [1] .

The results shall be reported for each 100BASE-T1 port available in the DUT.</td></tr>
</table>

2.2 PMA

2.2.1 General

This chapter shall be used for evaluation of the Physical Layer of a 100BASE-T1 interface on DUT level. Except otherwise stated the measurements shall be conducted by room temperature (RT=23°C±5°C).

The tests shall be carried out based on the definitions of the related test specifications [2], [3], [5].

In the Synopsis the test Classification indicates if the test must be done or not in term of qualification.

Mandatory: the test is required and must be evaluated according to the specified pass/fail criterium.

Optional : the test could be executed but not required for an official qualification pass/fail criterium. Underground of the optional test is that PHY that already passed the whole test of the TC1 or parameters doubled checked do not need to be retested.

2.2.2 Transmitter Electrical Specifications

The following test cases specify the Requirements of the Transmitter Side (measurement point: MDI).

OABR_PMA_TX_01: Check the Transmitter outpout droop

Synopsis	Verification of the transmitter output droop. The test case shall be executed according to the definitions in [3], Test 5.1.1. **Test Classification: Optional**
Prerequisites	1. The DUT is connected to a stable power supply. 2. Use Link Partner or an interface to set the DUT's PHY into Test Mode operation (via 100BASE-T1, Standard Ethernet, CAN, FlexRay e.g.) 3. DUT must be able to set its PHY into Test Mode 1.
Test setup	according to [3], Test 5.1.1
Test procedure	according to [3], Test 5.1.1
Pass criteria	according to [3], Test 5.1.1
Test iterations	Accumulate min. 10 Samples to increase the Accuracy
Notes	The test shall be executed for each port of the DUT if it has a switch inside.

OABR_PMA_TX_02: Check the Transmitter Timing Jitter in MASTER Mode

Synopsis	Verification of the transmitter timing jitter in MASTER mode The test case shall be executed according to the definitions in [3], Test 5.1.3, Case 1 **Test Classification: Mandatory**
Prerequisites	1. The DUT is connected to a stable power supply. 2. Use Link Partner or an interface to set the DUT's PHY into Test Mode operation (via 100BASE-T1, Standard Ethernet, CAN, FlexRay e.g.) 3. DUT must be able to set its PHY into Test Mode 2.
Test setup	according to [3], Test 5.1.3, Case 1
Test procedure	according to [3], Test 5.1.3, Case 1
Pass criteria	according to [3], Test 5.1.3, Case 1
Test iterations	Accumulate min. 10 Samples to increase the Accuracy
Notes	The test shall be executed for each port of the DUT if it has a switch inside.

OABR_PMA_TX_03: Check the Transmit Clock Frequency

Synopsis	Verification of the transmit clock frequency. The test case shall be executed according to the definitions in [3], Test 5.1.5 **Test Classification: Mandatory**
Prerequisites	1. The DUT is connected to a stable power supply. 2. Use Link Partner or an interface to set the DUT's PHY into Test Mode operation (via 100BASE-T1, Standard Ethernet, CAN, FlexRay e.g.) 3. DUT must be able to set its PHY into Test Mode 2.
Test setup	according to [3], Test 5.1.5
Test procedure	according to [3], Test 5.1.5
Pass criteria	according to [3], Test 5.1.5
Test iterations	This test shall be conducted at all corner temperatures of the DUT (e.g. -40°C/RT/105°C). Accumulate min. 10 Samples to increase the Accuracy.
Notes	As the test is realized on three corner temperatures use a test cable that does not influence the test result . The test shall be executed for each port of the DUT if it has a switch inside. The corner temperatures of the DUT for the test have to be provided by the DUT manufacturer.

OABR_PMA_TX_04: Check the Transmitter Power Spectral Density (PSD)

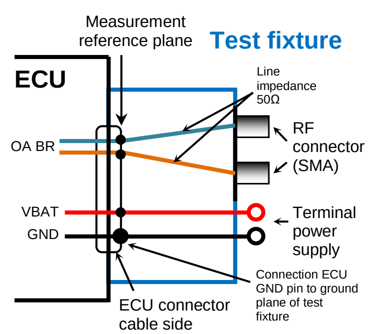
Synopsis	Verification of the transmitter power spectral density. The test case shall be executed according to the definitions in [3], Test 5.1.4 **Test Classification: Optional**
Prerequisites	1. The DUT is connected to a stable power supply. 2. Use Link Partner or an interface to set the DUT's PHY into Test Mode operation (via 100BASE-T1, Standard Ethernet, CAN, FlexRay e.g.) 3. DUT must be able to set its PHY into Test Mode 5.
Test setup	according to [3], Test 5.1.4
Test procedure	according to [3], Test 5.1.4
Pass criteria	according to [3], Test 5.1.4
Test iterations	If performing the test with a DSO, the averaging function of the scope shall be set at least to 50 times
Notes	The test shall be executed for each port of the DUT if it has a switch inside.

OABR_PMA_TX_05: Check MDI return Loss

The test case definition is based on chapter 96.8.2.1 MDI Return Loss of [2], Test 5.1.6 of [3] and the definitions in [4].

<table>
<tr><td>Synopsis</td><td>Shall ensure that the DUT respects the limits for the Return Loss.

Test Classification: Mandatory</td></tr>
<tr><td>Prerequisites</td><td>1. The DUT is connected to a stable power supply.
2. Use Link Partner or an interface to set the DUT’s PHY into SLAVE Mode operation (via 100BASE-T1, Standard Ethernet, CAN, FlexRay e.g.)
3. DUT must be able to set its PHY into Slave Mode operation.</td></tr>
<tr><td>Test setup</td><td>The measurement of the Return Loss shall be carried out with a Network Analyzer.



To achieve a high degree of reliability of measurement results the use of a specific test fixture for the connection to the DUT connector MDI pins is required. A test fixture according to the diagram above and in line with definitions of [4] shall be used. The ground pin(s) of the DUT shall be directly connected to the ground plane of the test fixture. If possible the original harness connector shall be used. It shall be a fixed part of the test fixture. The calibration reference plane is defined at the beginning of the harness connector on the test fixture. The following VNA settings shall be used for the measurement:

<table>
<tr><th>Parameter</th><th>Value</th></tr>
<tr><td>Sweep f_{Start}</td><td>300 kHz</td></tr>
<tr><td>Sweep f_{Stop}</td><td>1 GHz</td></tr>
<tr><td>Sweep type</td><td>Logarithmic</td></tr>
<tr><td>Sweep points</td><td>1600</td></tr>
<tr><td>Output power</td><td>minimum -10 dBm</td></tr>
<tr><td>Measurement bandwidth</td><td>100 Hz</td></tr>
<tr><td>Logic Port Impedance Differential Mode</td><td>100 Ω</td></tr>
<tr><td>Logic Port Impedance Common Mode</td><td>25 Ω</td></tr>
<tr><td>Data calibration kit (VNA)</td><td>used kit for calibration</td></tr>
<tr><td>Averaging function</td><td>16 times</td></tr>
</table>
</td></tr>
</table>

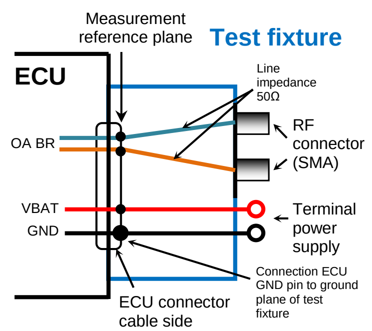
	Smoothing function deactivated
Test procedure	1. The DUT PHY is in SLAVE mode operation. 2. Use a test fixture as described in the test setup. 3. Connect the MDI via the test fixture to the Network Analyzer. 4. Measure the Value Return Loss (S_{dd11}) 5. Analyze the waveform. 6. Report the result with a resolution that shows: no limit Violation was detected.
Pass criteria	The test shall be classified as passed, if the value of the MDI Return Loss (S_{dd11}) fulfills the limit defined in chapter 96.8.2.1 MDI Return Loss of [2].
Test iterations	Single VNA measurement with enabled averaging function set to at least 16 times.
Notes	The test shall be executed for each port of the DUT if it has a switch inside.

OABR_PMA_TX_06: Check MDI Mode conversion

The test case definition is based on chapter 96.8.2.2 MDI mode conversion loss of [2], Test 5.1.7 of [3] and the definitions in [4].

<table>
<tr><td>Synopsis</td><td>Shall ensure that the DUT respects the limits for the Mode conversion. Shall ensure that the DUT front end respects the appropriate symmetry requirements.

Test Classification: Mandatory</td></tr>
<tr><td>Prerequisites</td><td>1. The DUT is connected to a stable power supply.
2. Use Link Partner or an interface to set the DUT’s PHY into SLAVE Mode operation (via 100BASE-T1, Standard Ethernet, CAN, FlexRay e.g.)
3. DUT must be able to set its PHY into Slave Mode operation.</td></tr>
<tr><td>Test setup</td><td>The measurement of the Mode Conversion shall be carried out with a Network Analyzer.

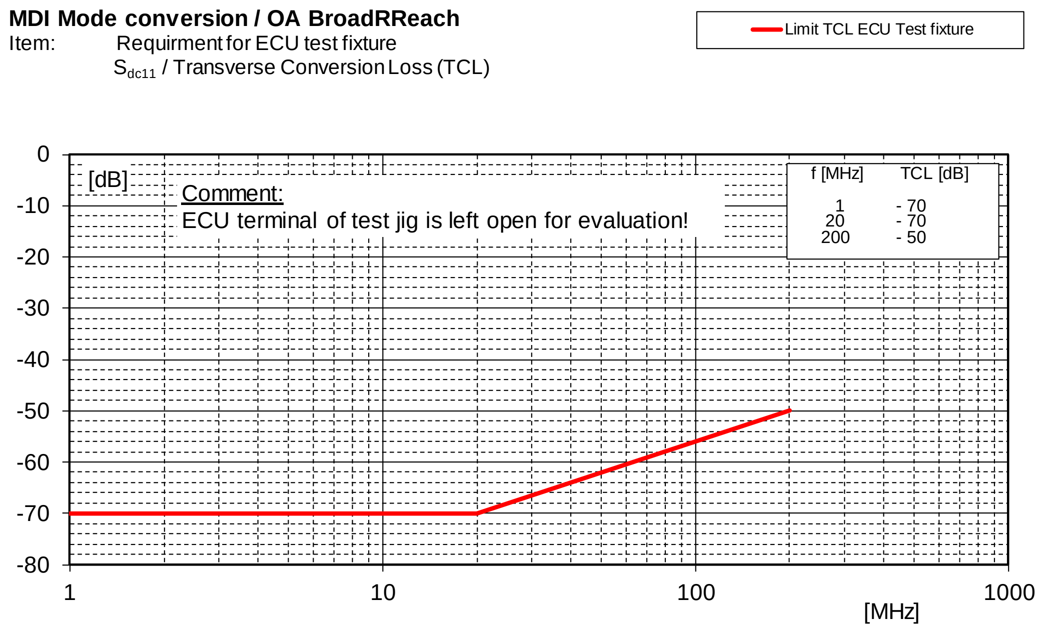


To achieve a high degree of reliability of measurement results the use of a specific test fixture for the connection to the DUT connector MDI pins is required. A test fixture according to the diagram above and in line with definitions of [4] shall be used. The ground pin(s) of the DUT shall be directly connected to the ground plane of the test fixture. If possible the original harness connector shall be used. It shall be a fixed part of the test fixture. The calibration reference plane is defined at the beginning of the harness connector on the test fixture. Additionally the used test fixture shall fulfill the limit for fixture self-conversion given below while the test fixture is not connected to the DUT (terminal left open).

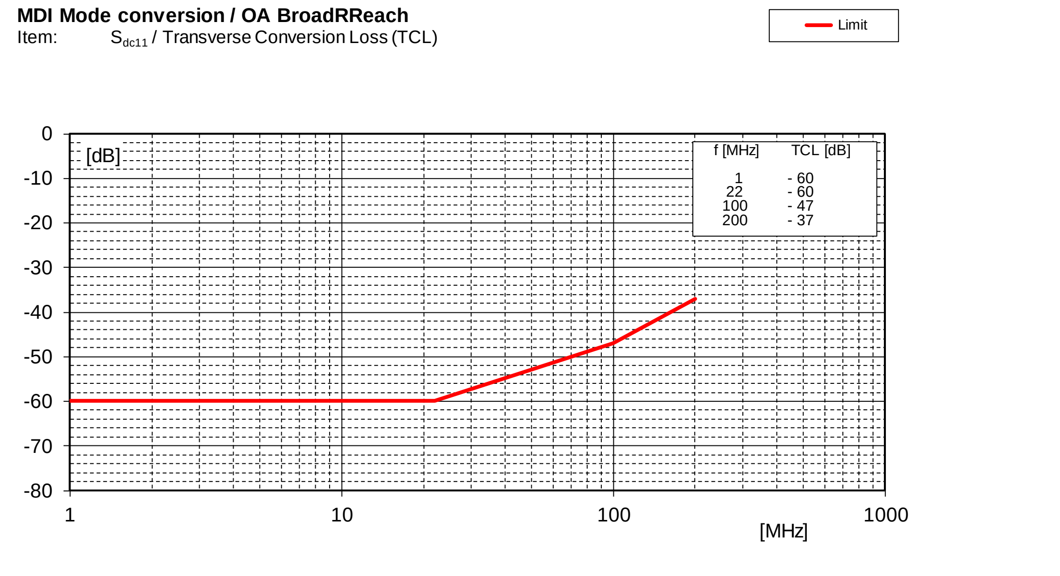
The following VNA settings shall should be used for the measurement:

<table>
<tr><th>Parameter</th><th>Value</th></tr>
<tr><td>Sweep f_{Start}</td><td>300 kHz</td></tr>
<tr><td>Sweep f_{Stop}</td><td>1 GHz</td></tr>
<tr><td>Sweep type</td><td>Logarithmic</td></tr>
</table></td></tr>
</table>

<table>
<tr>
<td></td>
<td>
Sweep points 1600
Output power minimum -10 dBm
Measurement bandwidth 100 Hz
Logic Port Impedance Differential Mode 100 Ω
Logic Port Impedance Common Mode 25 Ω
Data calibration kit (VNA) used kit for calibration
Averaging function 16 times
Smoothing function deactivated

Limit for test fixture self-conversion


</td>
</tr>
<tr>
<td>Test procedure</td>
<td>
1. The DUT PHY is in SLAVE mode operation.
2. Use a test fixture as described in the test setup.
3. Connect the MDI via the test fixture to the Network Analyzer.
4. Measure the Value Mode Conversion (S_{dc11})
5. Analyze the waveform.
6. Report the result with a resolution that shows: no limit Violation was detected.
</td>
</tr>
<tr>
<td>Pass criteria</td>
<td>For evaluation of MDI mode conversion measurements in the frequency range of 1 MHz to 1 GHz are required. The following limit shall be fulfilled.</td>
</tr>
</table>

	
Test iterations	Single VNA measurement with enabled averaging function set to at least 16 times.
Notes	The test shall be executed for each port of the DUT if it has a switch inside.

OABR_PMA_TX_07: Check MDI Common Mode emission

The test case shall be executed according to the definitions in [5], Appendix D – (informative) Test method for measuring of MDI RF common mode emission of ECUs

Synopsis	Measurement of the RF common mode emission at the DUT MDI. **Test Classification: Optional**
Prerequisites	1. The DUT is connected to a stable power supply. 2. Use Link Partner or an interface to set the DUT's PHY into Test Mode operation (via 100BASE-T1, Standard Ethernet, CAN, FlexRay e.g.) 3. DUT must be able to set its PHY into Test Mode 5.
Test setup	according to [5], Appendix D
Test procedure	according to [5], Appendix D
Pass criteria	The test shall be classified as passed, if the value of the MDI common mode emission fulfills the limit defined in [5], Figure D-2
Test iterations	according to [5], Appendix D (see Table D1 Numbers of pass)
Notes	The test shall be executedexecuted for each port of the DUT if it has a switch inside. The common mode emission is measured in a frequency range according to the definitions in [5], Appendix D – (informative) **Table D-1: Settings for measurement device for RF common mode emission measurement at MDI** . The absolute pass criteria is specified in the **Figure D-2: Recommended limit for MDI RF common mode emission**. In case of violations of the limit for frequencies greater than 70Mhz a warning issue or a comment shall be added in the test report.

OABR_PMA_TX_08: Check the Transmitter Distortion

Synopsis	Verification of the transmitter distortion. The test case shall be executed according to the definitions in [3], Test 5.1.2

	Test Classification: Optional
Prerequisites	1. The DUT is connected to a stable power supply. 2. Use Link Partner or an interface to set the DUT's PHY into Test Mode operation (via 100BASE-T1, Standard Ethernet, CAN, FlexRay e.g.) 3. DUT must be able to set its PHY into Test Mode 4.
Test setup	according to [3], Test 5.1.2
Test procedure	according to [3], Test 5.1.2
Pass criteria	according to [3], Test 5.1.2
Test iterations	10 times
Notes	The execution of this test is optional and not mandatory for the Compliance test. In Case of DUT test the TX_TCLK (66.7Mhz) is not accessible. Therefore, it is necessary to recover the TX_TCLK form the signal Itself. See Appendix 1AInstead of a 100 Ω differential voltage generator, the test case may be executed also with a single-ended voltage generator and a balun.

2.2.3 Appendix 1A Transmitter Distortion Test.

Transmitter Distortion test can be executed without TX_TCLK.

This part will give an overview of the test setup without TX_TCLK access, with disturber.

- The DUT is set in test mode 4 and the disturber signal of 5.4Vpp 11.111Mhz sine wave is injected on the DUT Transmitter.
- The main idea is to recover the TX_TCLK clock from the test mode 4 signal and apply the recovered timing to determine the right samples needed for [3],Test 5.1.2.

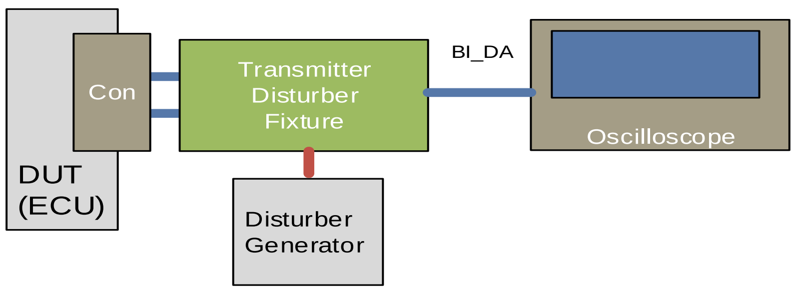


Figure 1 Transmitter Distortion Test

3 Test Scope Layer 2 of Automotive Ethernet

3.1 Overview & Requirements for ECU Automotive Ethernet Switch Testing Test Scope

The tests in this scope validate the behavior of the “Automotive Ethernet Switch“ within the ECU. The “Automotive Ethernet Switch“ is an entity that includes the switch silicon hardware and any additional hardware, firmware, and software needed to meets the requirements for a “MAC bridge“ set forth in the IEEE 802.1 standards.

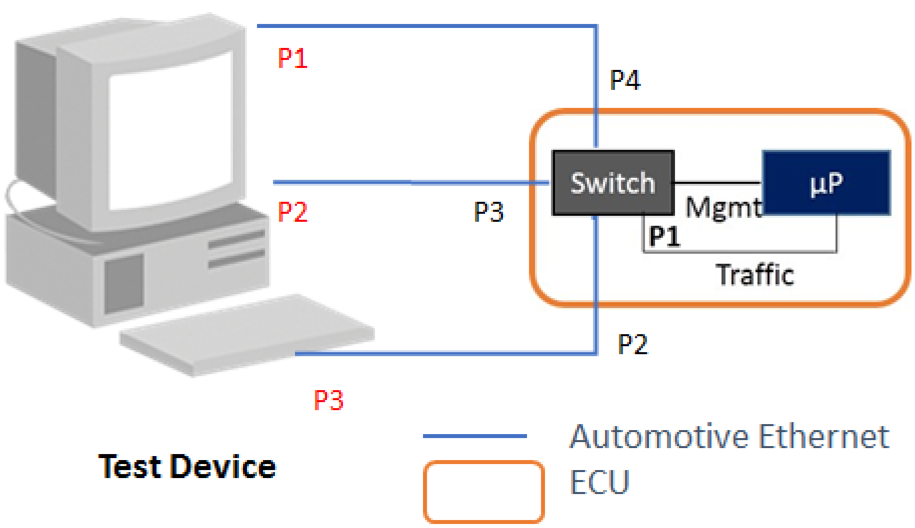
Any reference to “DUT“ in this test scope refers to the logical “Automotive Ethernet Switch“ including any software or configuration done in an MCU or CPU. The “test device“ shall include both externally connected hardware/software as well as software running on any MCU / CPU on the ECU that is connected to the “Automotive Ethernet Switch“ via an Automotive Ethernet Port.

The tests in this test scope are designed to test that the “Automotive Ethernet Switch“ entity is configured & operating correctly as per the ECU configuration, but assume that the functionality of the switch silicon, PHYs, or other components has been verified elsewhere.

The test cases are grouped by functional areas. Only those functional areas and test cases that are applicable to a given ECU need to be tested. The configuration of each ECU (including the switch configuration) should be used to determine which test cases are applicable.

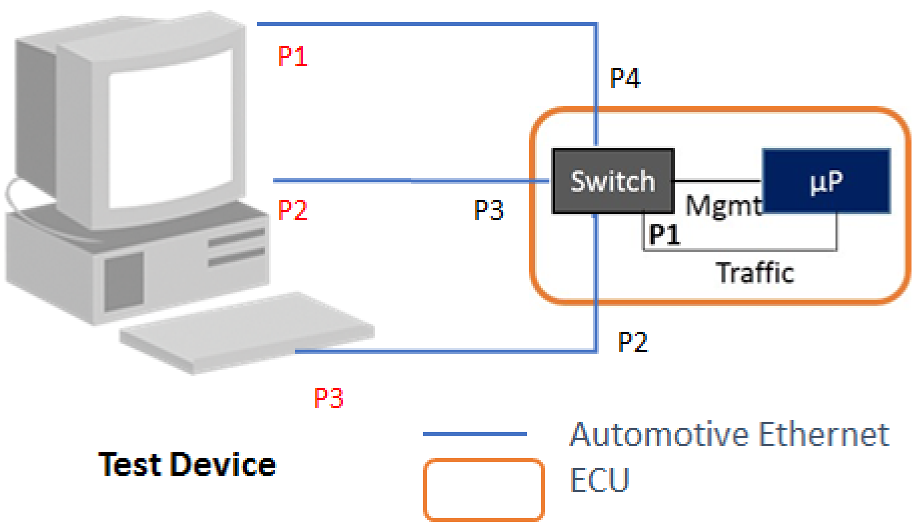
3.2 VLAN Testing

SWITCH_VLAN_01: VLAN Unicast Traffic Forwarding

Synopsis	The test shall ensure that the switch is able to forward the traffic for unidentified MACs in presence of line rate traffic, and drop the traffic received from un-configured VLANs.
Prerequisites	• The DUT shall be operated in normal mode, i.e. the switch is configured and ready to receive/forward Ethernet frames. • The DUT shall be capable of responding to Echo-Request message with Echo-Reply message on some or all of its ports.
Test setup	All available (external) ports of the DUT are connected to the test system. 
Test Input Parameters	1. Switch port Valid Source/Destination MAC Addresses are configured in either of these ways: a. read from the ECU Configuration file, if configured/available b. MAC Addresses identified for testing purpose 2. Valid VLANs are read from the ECU Configuration File, if available.
Test Procedure	1. Create the test topology as shown in the Setup. 2. On each test port, iterate through each source MAC and destination MAC address and transmit untagged Echo Request packets for each combination. 3. Verify that for each valid destination MAC, the Echo Request is received from each external DUT port except from the port the Echo Request packet has been sent to. 4. Verify that for each valid combination of destination MAC and VID, an Echo Reply from each internal port is received from the DUT port the Echo Request packet has been sent to. 5. Verify that for each invalid VLAN or each invalid MAC/VLAN combination, no Echo Request or Echo Reply has been received on any port. 6. On each test port, iterate through each VLAN ID, source MAC and destination MAC

	address and transmit single-tagged Echo Request packets for each combination. 7. Verify that for each valid destination MAC, the Echo Request is received from each external DUT port except from the port the Echo Request packet has been sent to. 8. Verify that for each valid combination of destination MAC and VID, an Echo Reply from each internal port is received from the DUT port the Echo Request packet has been sent to. 9. Verify that for each invalid VLAN or each invalid MAC/VLAN combination, no Echo Request or Echo Reply has been received on any port. 10. On each test port, iterate through each VLAN ID (as outer VLAN tag), source MAC and destination MAC address and transmit double-tagged Echo Request packets for each combination. 11. Verify that for each valid destination MAC, the Echo Request is received from each external DUT port except from the port the Echo Request packet has been sent to. 12. Verify that for each valid combination of destination MAC and VID, an Echo Reply from each internal port is received from the DUT port the Echo Request packet has been sent to. 13. Verify that for each invalid VLAN or each invalid MAC/VLAN combination, no Echo Request or Echo Reply has been received on any port.
Pass criteria	The DUT traffic counters match the expected test device traffic counters for each combination of VLAN, Source MAC and Destination MAC. The traffic received by the test device is tagged or untagged in accordance to the VLAN configuration provided by the manufacturer.
Test iterations	The test shall be repeated for each combination of ingress and egress ports for all the combinations of: • Source MAC Addresses • Destination MAC Addresses • 4094 VLANs (exclude reserved VIDs 0 and 4095) per port and VLAN untagged
Notes	Test derived from switch requirement VLAN01 and VLAN02: The switch shall support VLAN handling according to IEEE 802.1Q. The switch shall support at least x different VLAN-IDs which can be chosen freely from the entire range of all available 4095 VLAN-IDs.

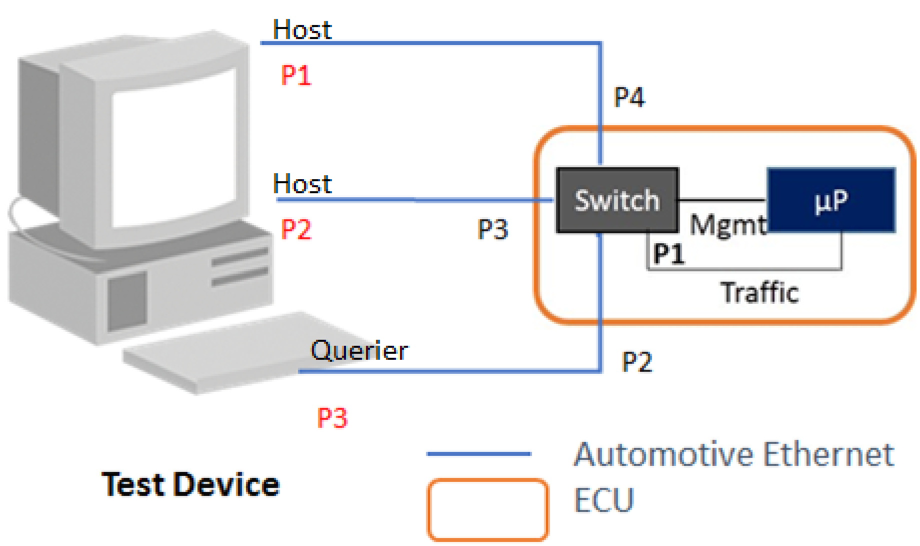
SWITCH_VLAN_02: VLAN Broadcast Traffic Forwarding

Synopsis	The test shall ensure that the switch is able to successfully forward the broadcast traffic from all configured VLANs in presence of line rate traffic, and drop the traffic received from un-configured VLANs.
Prerequisites	• The DUT shall be operated in normal mode, i.e. the switch is configured and ready to receive/forward Ethernet frames. • The DUT shall be capable of handling ARP request on some or all of its ports.
Test setup	All available (external) ports of the DUT are connected to the test system. 
Test Input Parameters	• Source MAC Addresses are configured in either of these ways: read from the ECU Configuration file, if configured/available MAC Addresses identified for testing purpose • Source IP Addresses are configured in either of these ways: read from the ECU Configuration file, if configured/available IP Addresses identified for testing purpose • Destination IP Addresses are configured in either of these ways: read from the ECU Configuration file, if configured/available IP Addresses identified for testing purpose • VLANs are read from the ECU Configuration File, if available.
Test	1. Create the test topology as shown in the Setup.

Procedure	2. On each test port, iterate through each source MAC, source IP and destination IP and transmit untagged broadcast ARP Request packets for each combination. 3. Verify that for each valid destination IP, the ARP Request is received from each external DUT port except from the port the ARP Request packet has been sent to. 4. Verify that for each valid combination of destination IP and VID, an ARP Reply from each internal port is received from the DUT port the ARP Request packet has been sent to. 5. Verify that for each invalid VLAN, no ARP Request or ARP Reply has been received on any port. 6. On each test port, iterate through each VLAN, source MAC, source IP and destination IP address and transmit single-tagged broadcast ARP Request packets for each combination. 7. Verify that for each valid destination MAC, the ARP Request is received from each external DUT port except from the port the ARP Request packet has been sent to. 8. Verify that for each valid combination of destination MAC and VID, an ARP Reply from each internal port is received from the DUT port the ARP Request packet has been sent to. 9. Verify that for each invalid VLAN, no ARP Request or ARP Reply has been received on any port. 10. On each test port, iterate through each VLAN (as outer VLAN tag), source MAC, source IP and destination IP address and transmit double-tagged broadcast ARP Request packets for each combination. 11. Verify that for each valid destination MAC, the ARP Request is received from each external DUT port except from the port the ARP Request packet has been sent to. 12. Verify that for each valid combination of destination MAC and VID, an ARP Reply from each internal port is received from the DUT port the ARP Request packet has been sent to. 13. Verify that for each invalid VLAN, no ARP Request or ARP Reply has been received on any port.
Pass criteria	The DUT traffic counters match the expected test device traffic counters for each combination of VLAN, Source MAC, Source IP and Destination IP. The traffic received by the test device is tagged or untagged in accordance to the VLAN configuration provided by the manufacturer.
Test iterations	The test shall be repeated for each combination of ingress and egress ports for all the combinations of: • Source MAC Addresses • Source IP Addresses • Destination IP Addresses • 4094 VLANs (exclude reserved VIDs 0 and 4095) per port and VLAN untagged

Notes	Test derived from switch requirement VLAN01 and VLAN02: The switch shall support VLAN handling according to IEEE 802.1Q. The switch shall support at least x different VLAN-IDs which can be chosen freely from the entire range of all available 4095 VLAN-IDs.

SWITCH_VLAN_03: VLAN Multicast Traffic Forwarding

Synopsis	The test shall ensure that the switch is able to forward multicast packets based on the DUT configuration.
Prerequisites	• The DUT shall be operated in normal mode, i.e. the switch is configured and ready to receive/forward Ethernet frames. • The DUT shall be capable of forwarding multicast message on all the ports belonging to a specific VLAN.
Test setup	All available (external) ports of the DUT are connected to the test system. 
Test Input Parameters	• Source MAC Addresses are configured in either of these ways: ○ read from the ECU Configuration file, if configured/available ○ MAC Addresses identified for testing purpose • Source IP Addresses are configured in either of these ways: ○ read from the ECU Configuration file, if configured/available ○ IP Addresses identified for testing purpose • Destination Multicast IP Addresses ○ read from the ECU Configuration file, if configured/available ○ Multicast IP Addresses identified for testing purpose • Destination MAC Address is the multicast MAC derived from the multicast Destination IP • 4094 VLANs (excluding reserved VLAN IDs 0 and 4095) per port and VLAN untagged • Frame size – varies with each test cycle

	• Bandwidth – varies with each test cycle
Test Procedure	1. Create the test topology as shown in the Setup. 2. Configure the test ports acting as Hosts to send join for the destination multicast groups. 3. Verify that the test ports acting as Hosts are receiving queries and the test port acting as Querier is receiving join/leave messages. 4. On each test port acting as IGMP Querier, iterate through each source MAC, source IP and destination multicast IP (destination MAC is derived from destination IP), and transmit untagged multicast IP traffic for each combination. 5. Verify that for each valid combination of source IP and destination IP, the IP multicast traffic is correctly replicated only on those test ports that have receivers for the IGMP Group (identified by destination multicast IP). Note: The ECU switch can be a pure/hybrid L2 switch and in that case multicast forwarding will be enabled either by IGMP snooping or static multicast forwarding entries. 6. On each test port acting as IGMP Querier, iterate through each VID, source MAC, source IP and destination multicast IP (destination MAC is derived from destination IP), and transmit single-tagged multicast IP traffic for each combination. 7. Verify that for each valid combination of source IP and destination IP, the IP multicast traffic is correctly replicated only on those test ports that have receivers for the IGMP Group (identified by destination multicast IP). 8. On each test port acting as IGMP Querier, iterate through each VID (as outer VLAN tag), source MAC, source IP and destination multicast IP (destination MAC is derived from destination IP), and transmit double-tagged multicast IP traffic for each combination. 9. Verify that for each valid combination of source IP and destination IP, the IP multicast traffic is correctly replicated only on those test ports that have receivers for the IGMP Group (identified by destination multicast IP).
Pass criteria	DUT traffic counters match the expected test device traffic counters for each combination of VLAN, Source MAC, Source IP and Destination IP. The traffic received by the test device is tagged or untagged in accordance to the VLAN configuration provided by the manufacturer.
Test iterations	• Repeat steps 4 through 7 for each frame size and bandwidth that needs to be tested.
Notes	Test derived from switch requirement VLAN01 and VLAN02: The switch shall support VLAN handling according to IEEE 802.1Q. The switch shall support at least x different VLAN-IDs which can be chosen freely from the entire range of all available 4095 VLAN-IDs.

3.3 General

8021X_EAPoL_Authentication_positive

Synopsys	Check if the switch authenticates a port
Prerequisites	DUT port A is not yet authenticated at the DUT
Test setup	
Test Input Parameters	EAPoL authentication information
Test Procedure	1. From the test station, send traffic to DUT port A. This test traffic shall contain frames such that every external and internal DUT port is addressed. The frames addressed to internal ports shall be frames that will be processed and answered by the application behind the according internal port. 2. At the test station, check if frames are being forwarded or replied to any DUT port 3. From the test station, send traffic to every external DUT port except port A, addressed to port A 4. At the test station, check if frames are being forwarded to DUT port A 5. Send EAPoL frames with valid authentication information to DUT port A in order to authenticate port A at the DUT 6. From the test station, send traffic to DUT port A. This test traffic shall contain frames such that every external and internal DUT port is addressed. The frames addressed to internal ports shall be frames that will be processed and answered by the application behind the according internal port. 7. Check, if frames are being forwarded to DUT egress ports 8. From the test station, send traffic to every external DUT port except port A, addressed to port A 9. At the test station, check if frames are being forwarded to DUT port A
Pass Criteria	2. The test station does not receive from any port any frame sent in step 1 or any reply to them 4. The test station does not receive any frame sent in step 3 from port A 7. The test station receives every frame that has been addressed to an external DUT port in step 4 7. For every frame addressed to an internal DUT port in step 4, the test station receives a reply from port A 9. The test station receives all frames sent in step 8 from port A
Reference	

Notes	

8021X_EAPoL_Authentication_negative

Synopsys	Check if the switch refuses authentication of a port for invalid EAPoL frames
Prerequisites	DUT port A is not yet authenticated at the DUT
Test setup	
Test Input Parameters	EAPoL authentication information
Test Procedure	1. From the test station, send traffic to DUT port A. This test traffic shall contain frames such that every external and internal DUT port is addressed. The frames addressed to internal ports shall be frames that will be processed and answered by the application behind the according internal port. 2. At the test station, check if frames are being forwarded or replied to any DUT port 3. From the test station, send traffic to every external DUT port except port A, addressed to port A 4. At the test station, check if frames are being forwarded to DUT port A 5. Send EAPoL frames with invalid authentication information to DUT port A in order to authenticate port A at the DUT 6. From the test station, send traffic to DUT port A. This test traffic shall contain frames such that every external and internal DUT port is addressed. The frames addressed to internal ports shall be frames that will be processed and answered by the application behind the according internal port. 7. Check, if frames are being forwarded to DUT egress ports 8. From the test station, send traffic to every external DUT port except port A, addressed to port A 9. At the test station, check if frames are being forwarded to DUT port A
Pass Criteria	2. The test station does not receive from any port any frame sent in step 1 or any reply to them 4. The test station does not receive any frame sent in step 3 from port A 7. The test station does not receive from any port any frame sent in step 1 or any reply to them 9. The test station does not receive any frame sent in step 3 from port A
Reference	

Notes	

8021AE_MACsec_frames_forwarded

Synopsys	Check if the switch performs MACsec encryption correctly
Prerequisites	
Test setup	
Test Input Parameters	MACsec configuration settings of the DUT
Test Procedure	1. Capture traffic on all DUT ports, including the host controller port 2. From the test station, send frames to port A addressed to port B 3. Check, if frames are being forwarded to DUT port B 4. Check, if frames are being forwarded to any other DUT port 5. Repeat test steps 1-4 for every combination of ports A and B for which MACsec is supported.
Pass Criteria	3. For every frame sent in step 2, the test station received a corresponding, correctly MACsec encrypted frame from DUT port B. 4. No frame sent in step 2 is received by the test station on any other port than B, neither encrypted nor unencrypted.
Reference	

Port_Disabling

Synopsys	Check if switch has Port Disabling implemented. Sending frames on every ingress mirror port like specified/configured and check if frames are getting dropped and ignored for address learning.
Prerequisites	1. Link up and stable between test station and DUT ports
Test setup	
Test Input Parameters	
Test Procedure	1. Receive and monitor all traffic from all ports 2. From the test station send traffic to a port A addressed to a port B

	3. Observe if test station receives any traffic from port B 4. Through any vendor specified means, disable port B 5. Observe if link at port B is still up when the port is disabled 6. From the test station send traffic to port A addressed to port B 7. Observe if test station receives any traffic from port B 8. From the test station send traffic to port B addressed to port A and with a source MAC address X that has not been learnt on any DUT port yet. 9. Observe if test station receives any traffic from port A that has been sent to port B in step 8. 10. If flooding of frames with unknown destination MAC addresses is enabled, then from the test station send traffic to port A addressed to the MAC address X. 11. If flooding of frames with unknown destination MAC addresses is enabled, observe if test station receives on any port any traffic that has been sent to port A in step 10. 12. Repeat the test steps 1-11 for all ports that can be disabled as port B.
Pass Criteria	3. Test Station receives traffic from port B 5. Link at port B is still up 7. Test Station receives no traffic from port B 9. Test Station receives no traffic from port A 11. Test Station receives traffic from all ports
Reference	
Notes	

3.4 Address Learning

Address_Learning_read_ARL_table

Synopsys	Check if switch supports reading the learned ARL table.
Prerequisites	
Test setup	
Test Input Parameters	
Test Procedure	1. Receive and monitor all traffic from all ports 2. Through any vendor specified means have the DUT delete its Address Table entries 3. Wait 10 seconds 4. From the test station, send at least 2 untagged and at least 2 tagged frames

	to each of the DUT ports, each frame with a different, valid source MAC address and each of the tagged frames with a different valid VLAN tag according to the VLAN configuration of the corresponding port 5. Through any vendor specified means, read the learned MAC address table
Pass Criteria	5. The read table correctly lists the MAC addresses and the corresponding ports as learned in step 2.
Reference	
Notes	The ARL table MAY contain additional addresses that have been learnt from frames that have been generated from applications behind the internal ports.

Address_Learning_write_ARL_table

Synopsys	Check if switch supports writing to the learned ARL table.
Prerequisites	
Test setup	
Test Input Parameters	
Test Procedure	1. Receive and monitor all traffic from all ports 2. Through any vendor specified means have the DUT delete its Address Table entries 3. Through any vendor specified means, write to the learned MAC address table. The written address table should not contain more than the maximum number of supported, dynamically learnt entries. Furthermore, the written MAC address table should contain both unicast and multicast entries. For every DUT port, there should be at least one unicast and one multicast entry addressing to it. 4. Through any vendor specified means, read the learned MAC address table 5. From the test station, send untagged frames to every DUT port, each frame using one of the MAC addresses written to the MAC address table in step 2 as the destination MAC address. For every written entry the corresponding MAC address should be used at least once. 6. At every DUT port, check if the frames have been forwarded correctly 7. From the test station, send tagged frames to each of the remaining DUT ports, each frame using one of the MAC addresses written to the MAC address table in step 2 as the destination MAC address and a valid VLAN tag according to the VLAN configuration for the port corresponding to the destination address. For every written entry the corresponding MAC address should be used at least once. 8. At every DUT port, check if the frames have been forwarded correctly

Pass Criteria	4. The table correctly lists the MAC addresses and the corresponding ports as configured in step 2. 6. The test station receives all frames exclusively at the correct destination ports according to the configuration in step 2. 8. The test station receives all frames exclusively at the correct destination ports according to the configuration in step 2.
Reference	
Notes	

Address_Learning_with_untagged_frames_at_external_ports

Synopsys	Check if switch is capable of MAC address learning using untagged frames with unique source MAC addresses.
Prerequisites	
Test setup	
Test Input Parameters	
Test Procedure	1. Through any vendor specified means, have the DUT delete its Address Table entries 2. Capture and monitor traffic on every port 3. From the test station, send n untagged frames to one external DUT port A, each frame with a different, valid source MAC address; n shall be the number of ARL entries that are expected to be learnt (i.e. 256 minus the number of statically configured ARL entries) 4. From the test station, send n untagged frames to each of the remaining external DUT ports, each frame using one of the source MAC addresses from step 1 as the destination MAC address. 5. At every DUT port, check if the frames have been forwarded 6. From the test station, send n tagged frames to each of the remaining external DUT ports, each frame using one of the source MAC addresses from step 1 as the destination MAC address and a valid VLAN tag according to the VLAN configuration for port A. 7. At every DUT port, check if the frames have been forwarded 8. Repeat test steps 1 to 7 for every external DUT port as port A
Pass Criteria	5. At port A, all n frames per each other external port are seen 5. At the remaining external ports, no frame is seen 7. At port A, all n frames per each other external port are seen 7. At the remaining external ports, no frame is seen

Reference	
Notes	

Address_Learning_with_tagged_frames_at_external_ports

Synopsys	Check if switch is capable of MAC address learning using tagged frames with unique source MAC addresses.
Prerequisites	
Test setup	
Test Input Parameters	
Test Procedure	1. Through any vendor specified means, have the DUT delete its Address Table entries 2. Capture and monitor traffic on every port 3. From the test station, send n tagged frames to one external DUT port A, each frame with a different, valid source MAC address and a valid VLAN tag according to the VLAN configuration for port A; n shall be the number of ARL entries that are expected to be learnt (i.e. 256 minus the number of statically configured ARL entries) 4. From the test station, send n untagged frames to each of the remaining external DUT ports, each frame using one of the source MAC addresses from step 1 as the destination MAC address. 5. At every DUT port, check if the frames have been forwarded 6. From the test station, send n tagged frames to each of the remaining external DUT ports, each frame using one of the source MAC addresses from step 1 as the destination MAC address and the same VLAN tag used as in step 3. 7. At every DUT port, check if the frames have been forwarded 8. From the test station, send n tagged frames to each of the remaining external DUT ports, each frame using one of the source MAC addresses from step 1 as the destination MAC address and a valid VLAN tag according to the VLAN configuration of port A, but different to that VLAN tag used in step 3. 9. At every DUT port, check if the frames have been forwarded 10. Repeat test steps 1 to 7 for every external DUT port as port A
Pass Criteria	5. At port A, all n frames per each other external port are seen 5. At the remaining external ports, no frame is seen 7. At port A, all n frames per each other external port are seen 7. At the remaining external ports, no frame is seen

	9. At port A, all n frames per each other external port are seen 9. At the remaining external ports, no frame is seen
Reference	
Notes	

Address_Learning_at_internal_ports

Synopsys	Check if switch is capable of MAC address learning using untagged frames with unique source MAC addresses.
Prerequisites	
Test setup	
Test Input Parameters	
Test Procedure	1. Through any vendor specified means, have the DUT delete its Address Table entries 2. Capture and monitor traffic on every port 3. From the test station, send frames that will be processed and answered by the DUT to an external DUT port A, addressed to an internal DUT port B and with a valid VLAN tag 4. Wait for the test station to receive an answer and note the source MAC address of the answer 5. From the test station, send frames to an external DUT port A, addressed to the internal port's MAC address obtained in step 4 6. At the test station, check if the frames are forwarded to any external DUT port 7. At the test station, check if an answer to the frame sent in step 5 is received from port A 8. Repeat test steps 1 to 7 for every internal DUT port as port B
Pass Criteria	6. The test station did not receive the frames from any external DUT port 7. The test station received an answer from port A
Reference	
Notes	

3.5 Filtering of incoming frames

SWITCH_Policing_Information_external_ports

Synopsys	Check if the switch is able to provide status information regarding the configured ingress filters.
Prerequisites	
Test setup	
Test Input Parameters	Filtering rules of the DUT
Test Procedure	1. Enable traffic monitoring on all switch ports 2. Send 100 frames with MAC Destination Address <L2_DestinationAddress_Drop> from Test Station to a DUT port A 3. Wait to check if the frames are being forwarded 4. Read DUT counter indicating number of frames affected by the filtering rules of port A
Pass Criteria	3. All frames are being blocked (none of the frames sent in step 2 were captured on any egress port) 4. The counter of ingress filtered frames at port A are equal to 100
Reference	
Notes	

SWITCH_Policing_Information_internal_ports

Synopsys	Check if the switch is able to provide status information regarding the configured ingress filters.
Prerequisites	
Test setup	
Test Input Parameters	Filtering rules of the DUT
Test Procedure	1. Enable traffic monitoring on all switch ports 2. Wait 60 seconds 3. Read DUT counters indicating a number of frames affected by the filtering rules of all internal ports

Pass Criteria	3. The counters of ingress-filtered frames at internal ports are equal to 0 If a counter of ingress-filtered frames at any internal port is not 0, the test result shall be reported as “Warning” instead of “Fail”.
Reference	
Notes	In general, traffic generated by an application behind internal ports might influence several counters, including the counter of frames affected by the filtering rules at an internal port’s ingress. An application is, however, not expected to send frames that are filtered already at the ingress, but it is not impossible, either. It is out of the scope of this test to do further investigations in case that a counter of filtered frames for any internal port has a non-zero value as – most likely – this would be an issue caused by the application itself which unnecessarily or unintentionally sends frames that will be filtered at ingress. This test is settled in the section “Filtering of incoming frames”, but not passing the Pass Criteria does not clearly indicate a failure of the filtering mechanism. Thus, in such a case, the test result shall be reported as “Warning” instead of “Fail”.

3.6 Referenced TC 11 Tests

Since the TC11 Test specification does define a set of test cases for switch silicon it is useful to reference these testcases when useful to do a retest for switches which are implemented in ECUs.

One of the main differences of testing switch chips and already implemented switch chips (as done in ECUs) is that the ECU has a defined configuration. In the most use cases, the configuration is fixed and a modification of the configuration shoud be not possible at all, due to secutirty issues.

Therefore there is a subset of TC11 testcases which are should be considered again when ECUs are tested. It is obvious that the verification of a correct implementaion of a configuration adds value to the project.

General Requirements (GEN)

Test ID	Test Name
	Operate_as_Store_and_Forward_Switch
	Non-blocking_architecture
	Boot_time_with_PHYs
	8021X_Drop_Frames_With_Unknown_Source_Address
	8021X_Forward_Frames_With_Unknown_Source_Address_To_Host
	Port_Mirroring
	Port_Disabling
	Support_Jumbo_Frames
	General_Disable_Jumbo_Frames
	Read_Out_Device_ID

Address Resolution (ADDR)

Test ID	Test Name
	Address_Learning_ageing
	Address_Learning_ageing_time
	Address_Learning_disable_learning_on_specific_port
	Address_Learning_behaviour_upon_unknown_destination_address
	Address_Learning_behaviour_upon_MAAP_destination_address
	Address_Learning_behaviour_upon_01_80_C2_destination_address
	Address_Learning_ARL_table_overflow_status_info
	Address_Learning_forward_to_specific_port
	Address_Learning_one_shot_mode
	Address_Learning_limited_number_of_learnt_addresses_per_port
	Address_Learning_1024_addresses_without_hash_conflict_with_untagged_frames
	Address_Learning_1024_addresses_without_hash_conflict_with_tagged_frames

Time Synchronization (TIME)

Test ID	Test Name
	PTP_1_Step_Clock
	PTP_2_Step_Clock
	PTP_1_Step_Clock_Specific_MAC_Header
	PTP_1_and_2_Step_Clock_simultaneously

Quality of Service and Audio/Video Bridging (QOS)

Test ID	Test Name
	Check_Support_Priority_Based_Quality_Of_Service
	Support_of_WRR
	Mapping_Priority_Information_Based_On_802.1Q
	Overwrite_Priority_Of_Frame_AT_Ingress_Port
	Freely_Mappable_Priorities_To_Internal_Queues_1
	Freely_Mappable_Priorities_To_Internal_Queues_2
	Freely_Mappable_To_Egress_per_Port
	Support_Of_8_Shapers_Per_Egress_Port
	Support_Of_Leaky_Bucket_Algorithm
	Credit_Based_Shaper_1
	Credit_Based_Shaper_2
	Individual_Deactivation_Of_Shaper

Filtering of Incoming Frames (FILT)

Test ID	Test Name
	Rate_limitation_ingress_VID_based
	Rate_limitation_ingress_priority_based
	Broadcast_Storm_Protection
	Multicast_Storm_Protection
	8_Policies_Per_Port
	Filtering_for_L2_Fields_DA
	Filtering_for_L2_Fields_SA
	Filtering_for_L2_Fields_EtherType
	Filtering_for_L3_Address_UntaggedIPv4
	Filtering_for_L3_Address_EtherTypeIPv4
	Filtering_for_L3_Address_TaggedIPv4
	Filtering_for_L4_Address_Ipv4HeaderVariableLength
	Filtering_for_L4_Address_UDPPorts
	Filtering_for_L4_Address_TCPPorts

Diagnostics (DIAG)

Test ID	Test Name
	Port_Based_Counters
	Phy_Counters_and_Status
	MIB_support_Perfomance_Counters

4 Test Scope TCP/IP Protocol Family

4.1 Prerequisites

To enable test depth for testing TCP/IP stack features inECU implementations it is necessary to introduce an Upper Tester interface between Tester and TCP/IP stack. This interface has to be implemented in the ECU and defines routines for configuration, triggering or result evaluation.

An example for a Upper Tester Implementation is following specification by AUTOSAR:

- Specification of Testability Protocol and Service Primitives AUTOSAR TC Release 1.2.0

4.2 Address Resolution Protocol (ARP)

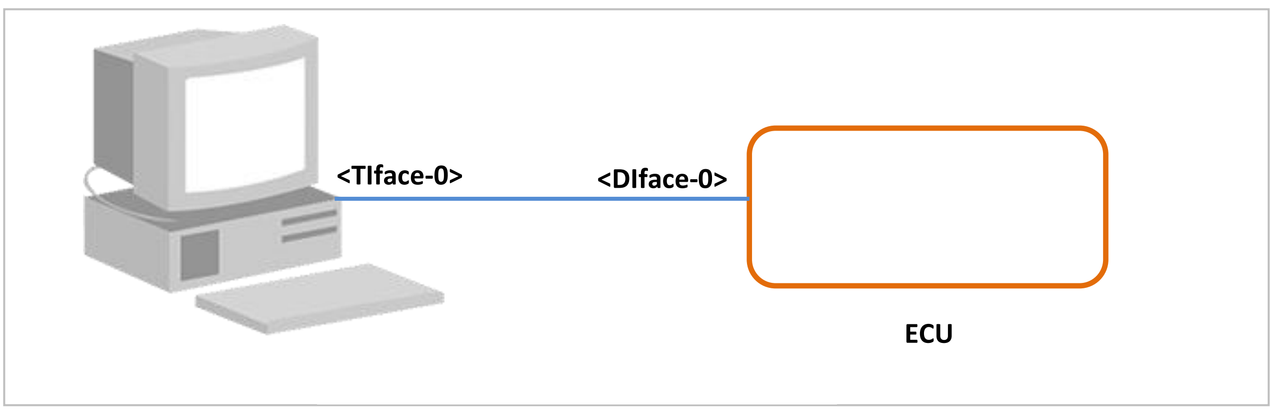
4.2.1 General

Referenced specification

The scope of this chapter is to specify test cases for Address Resolution Protocol from the following standards:

- RFC826

Simulated topologies



Required topology related configuration

Tester configuration required for the tests in the following sections pertaining to ARP tests:

- Correct DUT MAC Address for DUT interface connected to TESTER interface
- All test cases that use IP interface need to do ARP Packet exchange. This ARP packet exchange is performed after the DUT interface is configured with IP Address. Using this packet exchange

TESTER automatically learns the MAC address of DUT. The learned MAC address is then used in the test cases to send packets to DUT.

- All the test cases in this suite require DUT to be configured with only one IP interface

Coverage

Specification Document	Section Number	Test Category	Test Number(s)
1. RFC 826:An Ethernet Address Resolution Protocol, Packet Generation.	4.2.4.1	Packet Generation	ARP_01 to ARP_15
1. RFC 826:An Ethernet Address Resolution Protocol, Packet Reception.	4.2.4.2	Packet Reception	ARP_16 to ARP_49

4.2.2 Parameters used in the tests

User defined configuration parameters for IUT

Parameter used in test	Description
<DYNAMIC-ARP-CACHE-TIMEOUT>	This is the time for which a dynamic entry will be present in the ARP Cache. This timeout is only effective if it has been configured using the script named ARP DUT Configure ARP Dynamic Cache Entry Timeout Command
<DYNAMIC-ARP-CACHE-TIMEOUT>+<ARP-TOLERANCE-TIME>	This is the time calculated by adding <DYNAMIC-ARP-CACHE-TIMEOUT> and <ARP-TOLERANCE-TIME>

User defined configuration parameters for TESTER

Parameter used in test	Description
<HOST-1>	This denotes ARP Host simulated in the tester
<HOST-1-IP>	This denotes IP address of Host-1
<DIface-0>	This denotes the DUT interface to which TESTER host1 is connected.
<DIface-0-IP>	This denotes IP address of <DIface-0
<ParamListenTime>	This is the maximum time interval for which TESTER waits for a packet for cases when a certain event has been triggered on the DUT either by some protocol timer or using some external mechanism
<MAC-ADDR1>	The first unused MAC address that the TESTER can use for emulating specific topologies needed in test.

<MAC-ADDR2>	The second unused MAC address that the TESTER can use for emulating specific topologies needed in test. This is auto generated as consecutive to <MAC-ADDR1>
<MAC-ADDR3>	The third unused MAC address that the TESTER can use for emulating specific topologies needed in test. This is auto generated as consecutive to <MAC-ADDR2>
<DIFACE_O_MAC_ADDR>	This is the MAC address of <DIface-0> of DUT
<ARBIT_MAC_ADDR>	This indicates an arbitrary MAC address. This value is equal to 12:34:56:78:90:00
<ARP_HARDWARE_TYPE_UNKNOWN>	This indicates an unknown/wrong value of hardware type Resolution Packet
<UNKNOWN_HW_ADDR_LEN>	This indicates an unknown/wrong length of hardware address
<ARP_PROTOCOL_UNKNOWN>	This indicates an unknown/wrong value of protocol type
<UNKNOWN_PROTCOL_ADDR_LEN>	This indicates an unknown/wrong length of protocol address
<ARP-TOLERANCE-TIME>	This value depicts the time variance associated to any wait-event
<ETHERNET_ADDR_LEN>	The length in bytes of the Ethernet MAC Address, has the value 6

4.2.3 Terminology used in Test Procedure

Name	Description
DUT_CONFIGURE	This entry causes DUT to configure/execute various commands for clearing cache, adding static address, send Echo Request etc.
TESTER	Entity which is responsible for validating the Device under Test (DUT)
DUT	Device under Test
CLEANUP	This is a command which causes DUT to remove the static entry from its ARP cache

4.2.4 Test Cases ARP

4.2.4.1 Packet Generation

ARP_01: Static ARP entry (no ARP request)

Synopsis	The Address Resolution module tries to find the <protocol type, target protocol address> pair in a table. If it finds the pair, it gives the corresponding 48.bit Ethernet address back to the caller (hardware driver) which then transmits the packet. (Note: Here DUT is configured to add a static entry <HOST-1-IP, MAC-ADDR1> in its ARP cache. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will NOT send any ARP Request.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to add a static entry in its ARP Cache with - IP Address of <HOST-1-IP> - Ethernet Source Address of <MAC-ADDR1> 3. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 4. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 5. DUT: Does not send ARP Request 6. CLEANUP: Configure DUT to delete static entry in its ARP Cache with - IP Address of <HOST-1-IP> - Ethernet Source Address of <MAC-ADDR1>
Pass Criteria	5. DUT: Does not send ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_02: Static ARP entry (ARP entry used)

Synopsis	The Address Resolution module tries to find the <protocol type, target protocol address> pair in a table. If it finds the pair, it gives the corresponding 48.bit Ethernet address back to the caller (hardware driver) which then transmits the packet. (Note: Here DUT is configured to add a static entry <HOST-1-IP, MAC-ADDR1> in its ARP cache. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ICMP Echo Request.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to add a static entry in its ARP Cache with - IP Address of <HOST-1-IP> - Ethernet Source Address of <MAC-ADDR1> 3. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 4. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 5. DUT: Sends ICMP Echo Request - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1> 6. CLEANUP: Configure DUT to delete static entry in its ARP Cache with - IP Address of <HOST-1-IP> - Ethernet Source Address of <MAC-ADDR1>
Pass Criteria	5. DUT: Sends ICMP Echo Request - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_03: ARP entry learned on ARP request (no ARP request)

Synopsis	The Address Resolution module tries to find the <protocol type, target protocol address> pair in a table. If it finds the pair, it gives the corresponding 48.bit Ethernet address back to the caller (hardware driver) which then transmits the packet. (Note: The objective of the test case is to validate the ARP Learning mechanism on ARP requests. .Here TESTER sends an ARP Request to DUT so that an entry <HOST-1-IP, MAC-ADDR1> gets added in DUT's ARP cache. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will NOT send any ARP Request.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Source IP Address set to <HOST-1-IP> - Destination IP Address set to <DIface-0-IP> - Ethernet Source Address set to <MAC-ADDR1> 3. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 6. DUT: Does not send ARP Request
Pass Criteria	6. DUT: Does not send ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_04: ARP entry learned on ARP request (ARP entry used)

Synopsis	The Address Resolution module tries to find the <protocol type, target protocol address> pair in a table. If it finds the pair, it gives the corresponding 48.bit Ethernet address back to the caller (hardware driver) which then transmits the packet. (Note: Here TESTER sends an ARP Request to DUT so that an entry <HOST-1-IP, MAC-ADDR1> gets added in DUT's ARP cache. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ICMP Echo Request.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Source IP Address set to <HOST-1-IP> - Destination IP Address set to <DIface-0-IP> - Ethernet Source Address set to <MAC-ADDR1> 3. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 6. DUT: Sends ICMP Echo Request - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1>
Pass Criteria	6. DUT: Sends ICMP Echo Request - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_05: ARP entry learned on gratuitous ARP response (no ARP request)

Synopsis	The Address Resolution module tries to find the <protocol type, target protocol address> pair in a table. If it finds the pair, it gives the corresponding 48.bit Ethernet address back to the caller (hardware driver) which then transmits the packet. (Note: Here TESTER sends an ARP Response to DUT so that an entry <HOST-1-IP, MAC-ADDR1> gets added in DUT's ARP cache. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will NOT send any ARP Request.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Source IP Address set to <HOST-1-IP> - Destination IP Address set to <DIface-0-IP> - Ethernet Source Address set to <MAC-ADDR1> - Ethernet Destination Address set to ETHERNET_BROADCAST_ADDR 3. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the ARP cache of DUT to get refreshed 4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 6. DUT: Does not send ARP Request
Pass Criteria	6. DUT: Does not send ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_06: ARP entry learned on gratuitous ARP response (ARP entry used)

Synopsis	The Address Resolution module tries to find the <protocol type, target protocol address> pair in a table. If it finds the pair, it gives the corresponding 48.bit Ethernet address back to the caller (hardware driver) which then transmits the packet. (Note: Here TESTER sends an ARP Response to DUT so that an entry <HOST-1-IP, MAC-ADDR1> gets added in DUT's arp cache. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ICMP Echo Request.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Source IP Address set to <HOST-1-IP> - Destination IP Address set to <DIface-0-IP> - Ethernet Source Address set to <MAC-ADDR1> 3. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 6. DUT: Sends ICMP Echo Request - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1>
Pass Criteria	6. DUT: Sends ICMP Echo Request - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_07: ARP request sending (ARP request send on missing entry)

Synopsis	The Address Resolution module tries to find the <protocol type, target protocol address> pair in a table. If it does not, it probably informs the caller that it is throwing the packet away (on the assumption the packet will be retransmitted by a higher network layer), and generates an Ethernet packet with a type field of ether_type$ADDRESS_RESOLUTION.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - for Packet with Ethernet Type set to <Address-Resolution> 4. DUT: Sends ARP Request
Pass Criteria	4. DUT: Sends ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_08: ARP request sending (Hardware Type check)

Synopsis	The Address Resolution module sets the ar$hrd field to ares_hrd$Ethernet.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Hardware Type is set to ARP_HARDWARE_ETHERNET
Pass Criteria	4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Hardware Type is set to ARP_HARDWARE_ETHERNET
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_09: ARP request sending (Protocol Type check)

Synopsis	The Address Resolution module sets the ar$pro to the protocol type that is being resolved.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Protocol Type is set to ARP_PROTOCOL_IP
Pass Criteria	4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Protocol Type is set to ARP_PROTOCOL_IP
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_10: ARP request sending (Hardware Address Length check)

Synopsis	The Address Resolution module sets the ar$hln to 6 (the number of bytes in a 48.bit Ethernet address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Hardware Address Length is set to <ETHERNET_ADDR_LEN>
Pass Criteria	4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Hardware Address Length is set to <ETHERNET_ADDR_LEN>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_11: ARP request sending (Protocol Address Length check)

Synopsis	The Address Resolution module sets the ar$pln to the length of an address in that protocol
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Protocol Address Length is set to IP_ADDR_LEN
Pass Criteria	4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Protocol Address Length is set to IP_ADDR_LEN
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_12: ARP request sending (Operation Code check)

Synopsis	The Address Resolution module sets the ar$op to ares_op$REQUEST
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Operation code is set to OPERATION_REQUEST
Pass Criteria	4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Operation code is set to OPERATION_REQUEST
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_13: ARP request sending (ARP Sender Hardware Address check)

Synopsis	The Address Resolution module sets the ar$sha with the 48.bit ethernet address of itself.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - ARP Sender Hardware Address is set to <DIFACE-0-MAC-ADDR>
Pass Criteria	4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - ARP Sender Hardware Address is set to <DIFACE-0-MAC-ADDR>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_14: ARP request sending (Source IP Address check)

Synopsis	The Address Resolution module sets the ar$spa with the protocol address of itself
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Source IP Address is set to <DIface-0-IP>
Pass Criteria	4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Source IP Address is set to <DIface-0-IP>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_15: ARP request sending (Destination IP Address correct)

Synopsis	The Address Resolution module sets the ar$tpa with the protocol address of the machine that is trying to be accessed
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Destination IP Address is set to <HOST-1-IP>
Pass Criteria	4. DUT: Sends ARP Request 5. TESTER: Verify that received ARP Request contains: - Destination IP Address is set to <HOST-1-IP>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

4.2.4.2 Packet Reception

ARP_16: ARP request reception (ARP Target Hardware Address = 00:00:00:00:00:00)

Synopsis	The Address Resolution module does not set ar$tha to anything in particular, because it is this value that it is trying to determine. (Note: In this test TESTER sends an ARP Request with ARP Target Hardware Address set to all zeroes, and, expects that DUT will send an ARP Response after receiving the ARP Request)

Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Target Hardware Address set to <all-zeroes> 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Sends ARP Response
Pass Criteria	3. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_17: ARP request reception (ARP Target Hardware Address = ff:ff:ff:ff:ff:ff)

Synopsis	The Address Resolution module does not set ar$tha to anything in particular, because it is this value that it is trying to determine. (Note: In this test TESTER sends an ARP Request with ARP Target Hardware Address set to Ethernet Broadcast Address, and, expects that DUT will send an ARP Response after receiving the ARP Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Sends ARP Response

Pass Criteria	3. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_18: ARP request reception (ARP Target Hardware Address = random)

Synopsis	The Address Resolution module does not set ar$tha to anything in particular, because it is this value that it is trying to determine. (Note: In this test TESTER sends an ARP Request with ARP Target Hardware Address set to an arbitrary value, and, expects that DUT will send an ARP Response after receiving the ARP Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Target Hardware Address set to ARBIT_MAC_ADDR 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Sends ARP Response
Pass Criteria	3. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_19: ARP request reception (ARP Target Hardware Address = Address of DUT)

Synopsis	The Address Resolution module does not set ar$tha to anything in particular, because it is this value that it is trying to determine. (Note: In this test TESTER sends an ARP Request with ARP Target Hardware Address set to DUT MAC Address, Ethernet Destination Address set to Ethernet Broadcast Address, and, expects that DUT will send an ARP Response after

	receiving the ARP Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Target Hardware Address set to <DIFACE-0-MAC-ADDR> - Ethernet Destination Address set to ETHERNET_BROADCAST_ADDR 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Sends ARP Response
Pass Criteria	3. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Generation" (MUST)
Notes	

ARP_20: ARP request *reception (Hardware Type correct)*

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Do I have the hardware type in ar$hrd? (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with hardware type field set to Ethernet. All the other fields in the ARP Request Packet are set to their correct values. It then expects that DUT should send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> Check section general Input Parameters

Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Hardware Type set to ARP_HARDWARE_ETHERNET - Ethernet Source Address set to <MAC-ADDR1> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Response
Pass Criteria	4. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_21: ARP request reception (*Hardware Type wrong)*

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Do I have the hardware type in ar$hrd? (Note:Here TESTER is sending correct values for all the fields in the ARP Request packet except hardware type field and also TESTER is configuring DUT to clear its ARP Cache entries.The hardware type field is set to an unknown hardware type value, and TESTER expects that DUT will not send any ARP Response)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Hardware Type set to ARP_HARDWARE_TYPE_UNKNOWN

	3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Does not send ARP Response
Pass Criteria	4. DUT: Does not send ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_22: ARP response reception (Hardware Type wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Do I have the hardware type in ar$hrd? (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries.TESTER then sends an ARP Response with hardware type field set to an unknown hardware type value. All the other fields in the ARP Response Packet are set to their correct values. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ARP Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Hardware Type set to ARP_HARDWARE_TYPE_UNKNOWN - ARP Sender Hardware Address set to <MAC-ADDR1> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 3. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP>

	- Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> 6. DUT: Sends ARP Request
Pass Criteria	6. DUT: Sends ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_23: ARP request reception (Hardware Address Length correct)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet [optionally check the hardware length ar$hln] (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with hardware address length field set to Ethernet Address Length. All the other fields in the ARP Request Packet are set to their correct values. It then expects that DUT should send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Hardware Address Length set to <ETHERNET_ADDR_LEN> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Response
Pass Criteria	4. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MAY)

Notes	

ARP_24: ARP request reception (Hardware Address Length wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet [optionally check the hardware length ar$hln] (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with hardware address length field set to unknown hardware address length. All the other fields in the ARP Request Packet are set to their correct values. It then expects that DUT should NOT send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Hardware Address Length set to UNKNOWN_HW_ADDR_LEN 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Does not send ARP Response
Pass Criteria	4. DUT: Does not send ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MAY)
Notes	

ARP_25: ARP request response (Hardware Address Length wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the

	following:Negative conditionals indicate an end of processing and a discarding of the packet [optionally check the hardware length ar$hln] (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries.TESTER then sends an ARP Response with hardware address length field set to an unknown hardware address length. All the other fields in the ARP Response Packet are set to their correct values. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ARP Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Hardware Address Length set to UNKNOWN_HW_ADDR_LEN - ARP Sender Hardware Address set to <MAC-ADDR1> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 3. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> 6. DUT: Sends ARP Request
Pass Criteria	6. DUT: Sends ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MAY)
Notes	

ARP_26: ARP request reception (Protocol Type correct)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Do I speak the protocol in ar$pro? (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with protocol type field set to type IP. All the other fields in the ARP Request Packet are set to their correct values. It then expects that DUT should send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Protocol Type set to ARP_PROTOCOL_IP 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Response
Pass Criteria	4. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_27: ARP request reception (Protocol Type wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Do I speak the protocol in ar$pro? (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with protocol type field set to an unknown protocol type value. All the other fields in the ARP Request Packet are set to

	their correct values. It then expects that DUT should NOT send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Protocol Type set to ARP_PROTOCOL_UNKNOWN 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Does not send ARP Response
Pass Criteria	4. DUT: Does not send ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_28: ARP response reception (Protocol Type wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Do I speak the protocol in ar$pro? (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries.TESTER then sends an ARP Response with protocol type field set to Unknown Protocol value. All the other fields in the ARP Response Packet are set to their correct values. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ARP Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input	<MAC-ADDR1>

Parameters	<ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Protocol Type set to ARP_PROTOCOL_UNKNOWN - ARP Sender Hardware Address set to <MAC-ADDR1> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 3. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> 6. DUT: Sends ARP Request
Pass Criteria	6. DUT: Sends ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_29: ARP request reception (Protocol Address Length correct)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet [optionally check the protocol length ar$pln] (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with protocol length field set to IP Address Length. All the other fields in the ARP Request Packet are set to their correct values. It then expects that DUT should send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1

Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Protocol Address Length set to IP_ADDR_LEN 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Response
Pass Criteria	4. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MAY)
Notes	

ARP_30: ARP request reception (Protocol Address Length wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet [optionally check the protocol length ar$pln] (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with protocol address length field set to an unknown protocol address length. All the other fields in the ARP Request Packet are set to their correct values. It then expects that DUT should NOT send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing:

	- Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Protocol Address Length set to UNKNOWN_PROTOCOL_ADDR_LEN 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Does not send ARP Response
Pass Criteria	4. DUT: Does not send ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MAY)
Notes	

ARP_31: ARP response reception (Protocol Address Length wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet [optionally check the protocol length ar$pln] (Note:In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Response with protocol length field set to an unknown protocol address length. All the other fields in the ARP Response Packet are set to their correct values. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ARP Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Protocol Address Length set to UNKNOWN_PROTOCOL_ADDR_LEN - ARP Sender Hardware Address set to <MAC-ADDR1> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 3. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed

	4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> 6. DUT: Sends ARP Request
Pass Criteria	6. DUT: Sends ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MAY)
Notes	

ARP_32: ARP entry update (Request/Request)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following. Negative conditionals indicate an end of processing and a discarding of the packet. Merge_flag := false If the pair <protocol type, sender protocol address is already in my translation table, update the ARP sender hardware address field of the entry with the new information in the packet and set Merge_flag to true. (Note: Here TESTER sends an ARP Request to DUT so that an entry gets added to the dut arp cache. TESTER then sends another ARP Request with a different Ethernet source address to DUT to check if the existing entry gets updated.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <MAC-ADDR2> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1>

	3. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR2> - Ethernet Source Address set to <MAC-ADDR2> 4. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 5. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 6. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR2> 7. DUT: Sends ICMP Echo Request
Pass Criteria	7. DUT: Sends ICMP Echo Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_33: ARP entry update (Response/Response)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following. Negative conditionals indicate an end of processing and a discarding of the packet. Merge_flag := false If the pair <protocol type, sender protocol address is already in my translation table, update the ARP sender hardware address field of the entry with the new information in the packet and set Merge_flag to true. (Note: Here TESTER sends an ARP Response to DUT so that an entry gets added to the dut arp cache. TESTER then sends another ARP Response with a different Ethernet source address to DUT to check if the existing entry gets updated.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <MAC-ADDR2> <ARP-TOLERANCE-TIME>

	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 3. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR2> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 4. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 5. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 6. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR2> 7. DUT: Sends ICMP Echo Request
Pass Criteria	7. DUT: Sends ICMP Echo Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_34: ARP entry update (Request/Response)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following. Negative conditionals indicate an end of processing and a discarding of the packet. Merge_flag := false If the pair <protocol type, sender protocol address is already in my translation table, update the ARP sender hardware address field of the entry with the new information in the packet and set Merge_flag to true. (Note: Here TESTER sends an ARP Request to DUT so that an entry gets added to the dut arp cache. TESTER then sends an ARP Response with a different Ethernet source address to DUT to check if the existing entry

	gets updated.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <MAC-ADDR2> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> 3. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR2> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 4. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 5. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 6. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR2> 7. DUT: Sends ICMP Echo Request
Pass Criteria	7. DUT: Sends ICMP Echo Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_35: ARP entry update (Response/Request)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following. Negative conditionals indicate an end of processing and a discarding of the packet. Merge_flag := false If the pair <protocol type, sender protocol address is already in my translation table, update the ARP sender hardware address field of the entry with the new information in the packet and set Merge_flag to true. (Note: Here TESTER sends an ARP Response to DUT so that an entry gets added to the dut arp cache. TESTER then sends an ARP Request with a different Ethernet source address to DUT to check if the existing entry gets updated.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <MAC-ADDR2> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 3. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR2> 4. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 5. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 6. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR2> 7. DUT: Sends ICMP Echo Request

Pass Criteria	7. DUT: Sends ICMP Echo Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_36: ARP request reception (Target Protocol Address correct)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Am I the target protocol address? (Note: In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with target protocol address field set to <DIface-0-IP>. All the other fields in the ARP Request Packet are set to their correct values. It then expects that DUT should send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Response
Pass Criteria	4. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_37: ARP request reception (Target Protocol Address wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Am I the target protocol address? (Note: In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Request with target protocol address field set to an IP First unused address value. All the other fields in the ARP Request Packet are set to their correct values. It then expects that DUT should NOT send an ARP Response.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <IP-FIRST-UNUSED-ADDR-INTERFACE-1> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Does not send ARP Response
Pass Criteria	4. DUT: Does not send ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_38: ARP response reception (Target Protocol Address wrong)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following:Negative conditionals indicate an end of processing and a discarding of the packet ?Am I the target protocol address? (Note: In this test TESTER is configuring DUT to clear its ARP Cache entries. TESTER then sends an ARP Response with target protocol address field set to IP First unused address value. All the other fields in the ARP Response

	Packet are set to their correct values. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ARP Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Source IP Address set to <HOST-1-IP> - Destination IP Address set to <IP-FIRST-UNUSED-ADDR-INTERFACE-1> - Ethernet Source Address set to <MAC-ADDR1> - Ethernet Destination Address set to ETHERNET_BROADCAST_ADDR 3. TESTER: <HOST-1> Waits up to (1) second(s) for the ARP cache of DUT to get refreshed 4. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 5. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> 6. DUT: Sends ARP Request
Pass Criteria	6. DUT: Sends ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_39: ARP learning (request answers request)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following. Negative conditionals indicate an end of processing and a discarding of the packet. If Merge_flag is false, add the triplet <protocol type, sender protocol address, sender hardware address> to the translation table. (Note: In this test TESTER sends an ARP

	Request with <sender protocol address, sender hardware address> fields set to <HOST-1-IP, MAC-ADDR2>. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ICMP Echo Request with Ethernet Destination Address set to <MAC-ADDR2>)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR2> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR2> 6. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 7. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 8. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR2> 9. DUT: Sends ICMP Echo Request
Pass Criteria	4. DUT: Sends ARP Request 9. DUT: Sends ICMP Echo Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_40: ARP learning (response answers request)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following. Negative conditionals indicate an end of processing and a discarding of the packet. If Merge_flag is false, add the triplet <protocol type, sender protocol address, sender hardware address> to the translation table. (Note:In this test TESTER sends an ARP Response with <sender protocol address, sender hardware address> fields set to <HOST-1-IP, MAC-ADDR3>. TESTER then causes DUT to send an ICMP Echo Request and expects that DUT will send an ICMP Echo Request with Ethernet Destination Address set to <MAC-ADDR3>)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR3> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Request 5. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR3> - ARP Target Hardware Address set to ETHERNET_BROADCAST_ADDR 6. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 7. DUT CONFIGURE: Configure DUT to send a UDP Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 8. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR3> 9. DUT: Sends ICMP Echo Request

Pass Criteria	4. DUT: Sends ARP Request 9. DUT: Sends ICMP Echo Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_41: ARP responding (response answers request)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following: ?Is the opcode ares_op$REQUEST? Yes:Set the ar$op field to ares_op$REPLY (Note: In this test TESTER sends an ARP Request. All other fields in the ARP Request message are set correctly. TESTER then expects that DUT will send an ARP Packet with Operation code field set to Response)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - for ARP Packet with Operation Code set to <Response> 3. DUT: Sends ARP Response
Pass Criteria	3. DUT: Sends ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_42: ARP responding (no response to response)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following: Negative conditionals indicate an end of processing and a discarding of the packet. ?Is the opcode ares_op$REQUEST? (Note:In this test TESTER sends an ARP Packet with opcode field set to response value, and expects that DUT will not send any ARP Response)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Response to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - Operation code set to OPERATION_RESPONSE 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Does not send ARP Response
Pass Criteria	3. DUT: Does not send ARP Response
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_43: ARP response (Ethernet Source Hardware Address check)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following: ?Is the opcode ares_op$REQUEST? Swap hardware field, putting the local hardware address in the sender field.
Prerequisites	Check section prerequisites
Test setup	Topology 1

Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Source IP Address set to <HOST-1-IP> - Destination IP Address set to <DIface-0-IP> 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Sends ARP Response 4. TESTER: Verify that received ARP Response contains: - Ethernet Source Hardware Address is set to <DIFACE-0-MAC-ADDR>
Pass Criteria	3. DUT: Sends ARP Response 4. TESTER: Verify that received ARP Response contains: - Ethernet Source Hardware Address is set to <DIFACE-0-MAC-ADDR>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_44: ARP response (Sender IP Address check)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following: ?Is the opcode ares_op$REQUEST? Swap protocol field, putting the local protocol address in the sender field.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Sends ARP Response 4. TESTER: Verify that received ARP Response contains: - Sender IP Address is set to <DIface-0-IP>

Pass Criteria	3. DUT: Sends ARP Response 4. TESTER: Verify that received ARP Response contains: - Sender IP Address is set to <DIface-0-IP>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_45: ARP response (ARP Target Hardware Address check)

Synopsis	When an address resolution packet is received, the receiving Ethernet module gives the packet to the Address Resolution module which goes through an algorithm similar to the following: ?Is the opcode ares_op$REQUEST? Yes: Send the packet to the (new) ARP target hardware address on the same hardware on which the request was received.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <MAC-ADDR2> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> 3. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends ARP Response 5. TESTER: Verify that received ARP Response contains: - ARP Target Hardware Address is set to <MAC-ADDR1> 6. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR2> 7. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 8. DUT: Sends ARP Response 9. TESTER: Verify that received ARP Response contains:

	- ARP Target Hardware Address is set to <MAC-ADDR2>
Pass Criteria	4. DUT: Sends ARP Response 5. TESTER: Verify that received ARP Response contains: - ARP Target Hardware Address is set to <MAC-ADDR1> 8. DUT: Sends ARP Response 9. TESTER: Verify that received ARP Response contains: - ARP Target Hardware Address is set to <MAC-ADDR2>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_46: ARP response (Hardware Type check)

Synopsis	For the 10Mbit Ethernet <ar$hrd> takes on the value <1>
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Sends ARP Response 4. TESTER: Verify that received ARP Response contains: - Hardware Type is set to ARP_HARDWARE_ETHERNET
Pass Criteria	3. DUT: Sends ARP Response 4. TESTER: Verify that received ARP Response contains: - Hardware Type is set to ARP_HARDWARE_ETHERNET
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_47: ARP response (Hardware Address Length check)

Synopsis	For the 10Mbit Ethernet <ar$hln> takes on the value <6>
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> 2. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 3. DUT: Sends ARP Response 4. TESTER: Verify that received ARP Response contains: - Hardware Address Length is set to <ETHERNET_ADDR_LEN>
Pass Criteria	3. DUT: Sends ARP Response 4. TESTER: Verify that received ARP Response contains: - Hardware Address Length is set to <ETHERNET_ADDR_LEN>
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Packet Reception" (MUST)
Notes	

ARP_48: ARP timeout (idle)

Synopsis	If no packets are received from a host for a suitable length of time, the address resolution entry is forgotten.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <DYNAMIC-ARP-CACHE-TIMEOUT> <ARP-TOLERANCE-TIME>

	<DYNAMIC-ARP-CACHE-TIMEOUT+ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to set a timeout of <DYNAMIC-ARP-CACHE-TIMEOUT> seconds for the dynamic entries in the ARP Cache of <DIface-0> 3. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> 4. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the ARP cache of DUT to get refreshed 5. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 6. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Target IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1> 7. DUT: Sends ICMP Echo Request 8. TESTER: <HOST-1> Waits up to (<DYNAMIC-ARP-CACHE-TIMEOUT+ARP-TOLERANCE-TIME>) second(s) for the arp cache of DUT to get refreshed 9. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 10. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 11. DUT: Sends ARP Request 12. CLEANUP: Configure DUT to clear a timeout of <DYNAMIC-ARP-CACHE-TIMEOUT> seconds for the dynamic entries in the ARP Cache of <DIface-0>
Pass Criteria	7. DUT: Sends ICMP Echo Request 11. DUT: Sends ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Related issue" (SHOULD)
Notes	

ARP_49: ARP timeout (busy)

Synopsis	It may be desirable to have table aging and/or timeouts. (Note: In this test case TESTER expects that DUT should delete a given Dynamic ARP Cache Entry even if it is being used.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<MAC-ADDR1> <DYNAMIC-ARP-CACHE-TIMEOUT> <ARP-TOLERANCE-TIME> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Configure DUT to clear the dynamic entries in the ARP Cache of <DIface-0> - containing IP Address <HOST-1-IP> 2. DUT CONFIGURE: Configure DUT to set a timeout of <DYNAMIC-ARP-CACHE-TIMEOUT> seconds for the dynamic entries in the ARP Cache of <DIface-0> 3. TESTER: <HOST-1> Sends ARP Request to DUT through <DIface-0> containing: - Sender IP Address set to <HOST-1-IP> - Target IP Address set to <DIface-0-IP> - ARP Sender Hardware Address set to <MAC-ADDR1> 4. TESTER: <HOST-1> Waits up to (<ARP-TOLERANCE-TIME>) second(s) for the ARP cache of DUT to get refreshed 5. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 6. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1> 7. DUT: Sends ICMP Echo Request 8. TESTER: <HOST-1> Waits up to (<DYNAMIC-ARP-CACHE-TIMEOUT>/2>) second(s) for the ARP cache of DUT to get refreshed 9. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 10. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> - Destination IP Address set to <HOST-1-IP> - Ethernet Destination Address set to <MAC-ADDR1> 11. DUT: Sends ICMP Echo Request 12. TESTER: <HOST-1> Waits up to

	((<DYNAMIC-ARP-CACHE-TIMEOUT>/2)+<ARP-TOLERANCE-TIME>) second(s) for the ARP cache of DUT to get refreshed 13. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> 14. TESTER: <HOST-1> Listens (up to <ParamListenTime>) on <DIface-0> 15. DUT: Sends ARP Request 16. CLEANUP: Configure DUT to clear a timeout of <DYNAMIC-ARP-CACHE-TIMEOUT> seconds for the dynamic entries in the ARP Cache of <DIface-0>
Pass Criteria	7. DUT: Sends ICMP Echo Request 11. DUT: Sends ICMP Echo Request 15. DUT: Sends ARP Request
Reference	RFC 826 "An Ethernet Address Resolution Protocol", section "Related issue" (MAY)
Notes	

4.3 Internet Control Message Protocol Version 4 (ICMPv4)

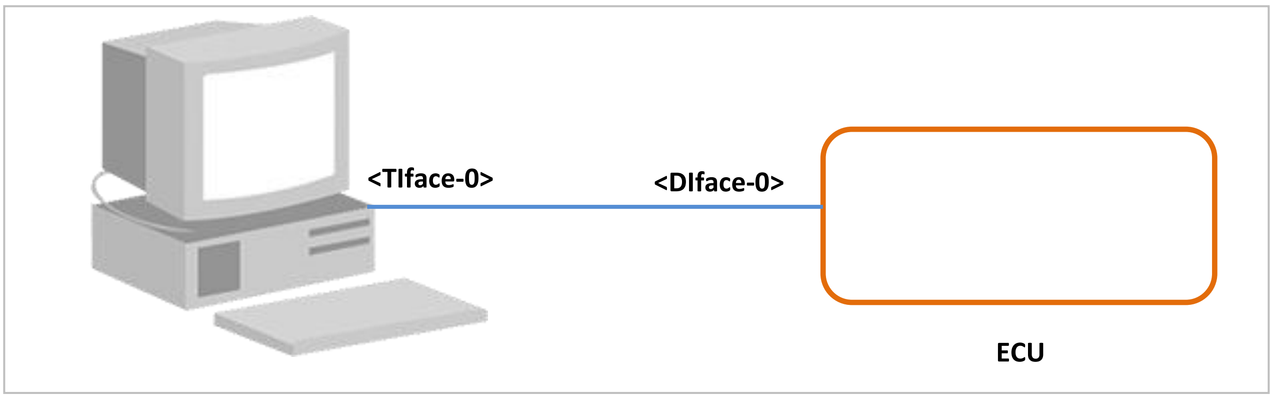
4.3.1 General

4.3.1.1 Referenced specification

The scope of this chapter is to specify test cases for the Internet Control Message Protocol Version 4 (ICMPv4) from the following standards:

- RFC 792 - Internet Control Message Protocol
- RFC 1122 - Requirements for Internet Hosts -- Communication Layers
- RFC 1812 - Requirements for IP Version 4 Routers

4.3.1.2 Simulated topologies



4.3.1.3 Required topology related configuration (prerequisites)

- This test suite expects to be running against an IP stack
- All tests run with one interface

4.3.1.4 Coverage

Specification Document	Section Number	Test Category	Test Number(s)
RFC 792: Internet Control Message Protocol		Error Handling	ICMPv4_ERROR_1 to ICMPv4_ERROR_3
RFC 1122: Requirements for Internet Hosts	3.2.2	Error Handling	ICMPv4_ERROR_4, ICMPv4_ERROR_5
RFC 792: Internet Control Message Protocol	4.3.3.2	ICMP Types	ICMPv4_TYPE_1 to ICMPv4_TYPE_12, ICMPv4_TYPE_16 to ICMPv4_TYPE_18, ICMPv4_TYPE_21,

			ICMPv4_TYPE_22

4.3.2 Parameters used in the tests

Parameter used in test	Description
<idfr>	Identifier used in the ICMP Messages to identify an ICMP Message
<seqno>	Sequence Numbers used in ICMP Messages to identify an ICMP Message
<broadcast-address>	IP broadcast address
<origTimestampValue>	Time the sender last touched a message before sending it. It is 32 bits of milliseconds since midnight UT.
<invalidChecksum>	This is the checksum which is different from the calculated checksum, i.e different from 16 bit one's complement of the one's complement sum of the ICMP message starting with the ICMP Type.
<ListenTime>	This is the maximum time interval for which TESTER waits for an ICMP Reply packet. This defaults to 3 seconds unless DUT configuration specifies otherwise.
<UnusedUDPPort>	An unused UDP port available on the DUT.
<FragReassemlyTimeout>	The fragment reassembly timeout. This defaults to 15 seconds.
<DUTSupportsIPOptions>	Automotive ECUs may not support IP options, either for performance or security reasons. TRUE indicates DUT supports IP options FALSE indicates DUT does not support IP options Default: TRUE
<TriggerTime>	This is the maximum time interval for which TESTER waits for a packet for cases when a certain event has to be manually triggered on the DUT either by some protocol timer or using some external mechanism. This defaults to 30 seconds unless DUT configuration specifies otherwise.
<unsupportedProtocol>	This is an IP protocol number that is not supported by the DUT.
<unknownType>	This is an ICMP type that is not assigned by any RFC.
<unreachablePort>	This defines a process port which is not active.

NOTE: As a general test pattern it can be assumed that all events will be guarded by a timer, e.g. receiving or non-receiving of message. Due to the difference nature of guarding, different default values can be assumed:

- Guarding user interactions: 30 seconds
- Guarding awaiting of response: 3 seconds
- Guarding awaiting of no response: 10 seconds

Protocol should be mentioned in the respective test cases.

4.3.3 Test cases ICMPv4

4.3.3.1 Error Handling

ICMPv4_ERROR_01: Avoid the infinite loop for ICMP message error

Synopsis	The ICMP messages typically report errors in the processing of datagrams. To avoid the infinite regress of messages about messages etc., no ICMP messages are sent about ICMP messages. (Note: This tests that no ICMP message is sent when an erroneous ICMP messages is received, thus avoiding an infinite loop. This test is ran when <DUTSupportsIPOptions> is TRUE)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Cause DUT to send an ICMP Echo Request through <DIface-0>, containing: - IP Destination Address field set to address of host-1 2. TESTER: Listen (for up to <TriggerTime> seconds) on <DIface-0> 3. DUT: Send an ICMP Echo Request message 4. TESTER: Send an ICMP Parameter Problem Message to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 12 - pointer field set to 11 - flg field set to zero - one timestamp value - ICMP Pointer field set to 20 (pointing option type) - ICMP Data field set to "Internet header + 64 bits of original data datagram" 5. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 6. DUT: Do not send ICMP Parameter Problem Message
Pass Criteria	3. DUT: Send an ICMP Echo Request message 6. DUT: Do not send ICMP Parameter Problem Message
Reference	RFC 792 p1 Introduction (MUST)

ICMPv4_ERROR_02: ICMP messages are only sent for fragment 0

Synopsis	Also ICMP messages are only sent about errors in handling fragment zero of fragmented datagrams. (Note: This tests that ICMP error message is sent on receiving the fragment

	having Fragment Offset field set to zero. This test is ran when <DUTSupportsIPOptions> is TRUE)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 10 - pointer field set to 9 - one timestamp value - first half of the constructed ICMP packet 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send one ICMP Parameter Problem message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 22 (Basic IP Header length (20) + third octet (pointer field of timestamp option)
Pass Criteria	3. DUT: Send one ICMP Parameter Problem message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 22 (Basic IP Header length (20) + third octet (pointer field of timestamp option)
Reference	RFC 792 p1 Introduction (MUST)

ICMPv4_ERROR_03: ICMP messages are not sent when fragment not 0

Synopsis	Also ICMP messages are only sent about errors in handling fragment zero of fragmented datagrams. (Note: This tests that ICMP error message is not sent on receiving non zero fragment i.e Fragment Offset field set to non-zero value. This test is ran when <DUTSupportsIPOptions> is TRUE)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input	Check section “Parameters used in the tests”

Parameters	
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 12 - pointer field set to 9 - one timestamp value - first half of the constructed ICMP packet 2. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags first, containing: - MF bit set to zero - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 10 - pointer field set to 9 - one timestamp value - last half of the constructed ICMP packet 3. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 4. DUT: Do not send ICMP Parameter Problem messages
Pass Criteria	4. DUT: Do not send ICMP Parameter Problem messages
Reference	RFC 792 p1 Introduction (MUST)
Notes	

ICMPv4_ERROR_04: ICMP messages are not sent for broadcast address

Synopsis	An ICMP error message MUST NOT be sent as the result of receiving a datagram destined to an IP broadcast. (Note: This test is ran when <DUTSupportsIPOptions> is TRUE)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<broadcast-address> Check section "Parameters used in the tests"
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to <broadcast-address> - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 10 - pointer field set to 9 - one timestamp value 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Do not send ICMP Parameter Problem Message
Pass Criteria	3. DUT: Do not send ICMP Parameter Problem Message
Reference	RFC 1122 s3.2.2 p39 Internet Control Message Protocol -- ICMP (MUST)
Notes	

ICMPv4_ERROR_05: Uknown ICMP message types are ignored

Synopsis	If an ICMP message of unknown type is received, it MUST be silently discarded.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an ICMP Message to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Type field set to <InvalidICMPType> 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Do not send any ICMP Message
Pass Criteria	3. DUT: Do not send any ICMP Message
Reference	RFC 1122 s3.2.2 p38 Internet Control Message Protocol -- ICMP (MUST)
Notes	

4.3.3.2 ICMP Types

ICMPv4_TYPE_01: Send ICMP Destination Unreachable for unknown port

Synopsis	If, in the destination host, the IP module cannot deliver the datagram because the indicated protocol module or process port is not active, the destination host may send a destination unreachable message to the source host.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send a UDP Packet to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Protocol field set to <ipTypeUDP> - UDP Destination Port set to <UnusedUDPPort> 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Destination Unreachable message 4. TESTER: Verify that the received Destination Unreachable message contains: - Code value set to 3 (port unreachable)
Pass Criteria	3. DUT: Send ICMP Destination Unreachable message 4. TESTER: Verify that the received Destination Unreachable message contains: - Code value set to 3 (port unreachable)
Reference	RFC 792 p5 Destination Unreachable Message (MAY)
Notes	

ICMPv4_TYPE_02: Discard packet with missing fragments

Synopsis	If a host reassembling a fragmented datagram cannot complete the reassembly due to missing fragments within its time limit it discards the datagram.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - first half of the constructed ICMP packet 2. TESTER: Wait for <FragReassemlyTimeout> seconds 3. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - last half of the constructed ICMP packet 4. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 5. DUT: Discard the fragments and do not send ICMP Echo Reply
Pass Criteria	5. DUT: Discard the fragments and do not send ICMP Echo Reply
Reference	RFC 792 p7 Time Exceeded Message (MUST)
Notes	

ICMPv4_TYPE_03: Send ICMP Time Exceeded message on fragmentation error

Synopsis	If a host reassembling a fragmented datagram cannot complete the reassembly due to missing fragments within its time limit it may send a time exceeded message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”

Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - first half of the constructed ICMP packet 2. TESTER: Wait for <FragReassemlyTimeout> seconds 3. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 4. DUT: Send an ICMP Time Exceeded message 5. TESTER: Verify that the received ICMP Time Exceeded message contains: - Code field set to 1 (fragment reassembly time exceeded)
Pass Criteria	4. DUT: Send an ICMP Time Exceeded message 5. TESTER: Verify that the received ICMP Time Exceeded message contains: - Code field set to 1 (fragment reassembly time exceeded)
Reference	RFC 792 p7 Time Exceeded Message (MAY)
Notes	

ICMPv4_TYPE_04: Do no send ICMP Time Exceeded message if missing fragment 0

Synopsis	If fragment zero is not available then no time exceeded need be sent at all.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to half of the constructed ICMP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - last half of the constructed ICMP packet 2. TESTER: Wait for <FragReassemlyTimeout> seconds 3. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 4. DUT: Do not send ICMP Time Exceeded message
Pass Criteria	4. DUT: Do not send ICMP Time Exceeded message
Reference	RFC 792 p7 Time Exceeded Message (MUST)
Notes	

ICMPv4_TYPE_05: Discard messages with header parameter problem

Synopsis	If the gateway or host processing a datagram finds a problem with the header parameters such that it cannot complete processing the datagram it must discard the datagram. (Note: This test is ran when <DUTSupportsIPOptions> is TRUE)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 10 - pointer field set to 9 - one timestamp value 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Discard the ICMP Echo Request and do not send ICMP Echo Reply
Pass Criteria	3. DUT: Discard the ICMP Echo Request and do not send ICMP Echo Reply
Reference	RFC 792 p9 Parameter Problem Message (MUST)
Notes	

ICMPv4_TYPE_06: Sending of ICMP Prameter Problem message

Synopsis	If the gateway or host processing a datagram finds a problem with the header parameters such that it cannot complete processing the datagram it may notify the source host via the parameter problem message. (Note: This test is ran when <DUTSupportsIPOptions> is TRUE)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing:

	- length field set to 10 - pointer field set to 9 - one timestamp value 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 22 (Basic IP Header length (20) + Length of Type field (1) + Length of Length field (1)
Pass Criteria	3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 22 (Basic IP Header length (20) + Length of Type field (1) + Length of Length field (1)
Reference	RFC 792 p9 Parameter Problem Message (MAY)
Notes	

ICMPv4_TYPE_07: ICMP Parameter Problem Message contents

Synopsis	Unused filed in Parameter Problem Message is set to zero and it contains the exactly same Internet Header and 64 bits of Original Datagram. (Note: This test is ran when <DUTSupportsIPOptions> is TRUE)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - Two Internet Timestamp options, each containing: - length field set to 12 - pointer field set to 9 - flg field set to zero - one timestamp value 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Unused field set to zero - Data field contains exactly same IP header and 64 bits of Original Data sent to the DUT

Pass Criteria	3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Unused field set to zero - Data field contains exactly same IP header and 64 bits of Original Data sent to the DUT
Reference	RFC 792 p9 Parameter Problem Message (MAY)
Notes	

ICMPv4_TYPE_08: ICMP Echo Reply message data field

Synopsis	The data received in the echo message must be returned in the echo reply message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Data field set to "ECU NETWORK VALIDATION TEST" 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that the received ICMP Echo Reply contains: - Data field set to "ECU NETWORK VALIDATION TEST"
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that the received ICMP Echo Reply contains: - Data field set to "ECU NETWORK VALIDATION TEST"
Reference	RFC 792 p15 Echo or Echo Reply Message (MUST)
Notes	

ICMPv4_TYPE_09: ICMP Echo Reply message id and sequence field

Synopsis	The identifier and sequence number may be used by the echo sender to aid in matching the replies with the echo requests.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<idfr> <seqno> Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1

	- IP Destination Address field set to address of DUT - Identifier field set to <idfr> - Sequence Number field set to <seqno> 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that the received ICMP Echo Reply contains: - Identifier field set to <idfr> - Sequence Number field set to <seqno>
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that the received ICMP Echo Reply contains: - Identifier field set to <idfr> - Sequence Number field set to <seqno>
Reference	RFC 792 p15 Echo or Echo Reply Message (MAY)
Notes	

ICMPv4_TYPE_10: ICMP checksum is checked

Synopsis	The checksum is the 16-bit one's complement of the one's complement sum of the ICMP message starting with the ICMP Type. For computing the checksum, the checksum field should be zero. If the total length is odd, the received data is padded with one octet of zeros for computing the checksum. (Note: This tests that a node does not send ICMP Echo Reply if ICMP Checksum is incorrect).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<invalidChecksum> Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Checksum field set to <invalidChecksum> 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Do not send ICMP Echo Reply
Pass Criteria	3. DUT: Do not send ICMP Echo Reply
Reference	RFC 792 p15 Echo or Echo Reply Message (MUST)
Notes	

ICMPv4_TYPE_11: ICMP Timestamp Reply message content

Synopsis	The data received (a timestamp) in the message is returned in the reply together with an additional timestamp.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<origTimestampValue> Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an ICMP Timestamp Message to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Originate Timestamp field set to <origTimestampValue> - Receive Timestamp field set to zero 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Timestamp Reply 4. TESTER: Verify that the received ICMP Timestamp Reply contains: - Originate Timestamp field set to <origTimestampValue> - Receive Timestamp field set to non-zero - Transmit Timestamp field set to non-zero
Pass Criteria	3. DUT: Send ICMP Timestamp Reply 4. TESTER: Verify that the received ICMP Timestamp Reply contains: - Originate Timestamp field set to <origTimestampValue> - Receive Timestamp field set to non-zero - Transmit Timestamp field set to non-zero
Reference	RFC 792 p17 Timestamp or Timestamp Reply Message (MAY)
Notes	

ICMPv4_TYPE_12: ICMP Timestamp Reply message id and sequence field

Synopsis	The identifier and sequence number may be used by the echo sender to aid in matching the replies with the requests.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<idfr> <seqno> Check section “Parameters used in the tests”
Test Procedure	1. TESTER: Send an ICMP Timestamp Message to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Identifier field set to <idfr> - Sequence Number field set to <seqno> 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Timestamp Reply 4. TESTER: Verify that the received ICMP Timestamp Reply contains: - Identifier field set to <idfr> - Sequence Number field set to <seqno>
Pass Criteria	3. DUT: Send ICMP Timestamp Reply 4. TESTER: Verify that the received ICMP Timestamp Reply contains: - Identifier field set to <idfr> - Sequence Number field set to <seqno>
Reference	RFC 792 p17 Timestamp or Timestamp Reply Message (MAY)
Notes	

ICMPv4_TYPE_16: Ensure that the DUT does not accept an ICMPv4 Information Request and does not generate a ICMPv4 Information Reply

Synopsis	A host SHOULD NOT implement these messages. [Note: referring to Information Request or Information Reply Message]
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1) TESTER: Send an ICMP Information Request to <DIface-0>, containing: - IP Source Address field set to network portion of IP address of host-1 - IP Destination Address field set to zero - Identifier field set to <idfr> - Sequence Number field set to <seqno> 2) TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3) DUT: Do not send ICMP Information Reply
Pass Criteria	Pass criteria: 3) DUT: Do not send ICMP Information Reply
Test Iterations	
Reference	Derived from RFC792, RFC1122 3.2.2.7
Notes	

DEPRECATED - ICMPv4_TYPE_17: Ensure that the DUT accepts an ICMPv4 Timestamp and generates a valid ICMPv4 Timestamp Reply

Synopsis	Ensure that when a DUT receives an ICMPv4 Packet containing a Type indicating a value of 13 (Timestamp Request) and containing a Code indicating a value of 0 and containing a valid Checksum and containing an Identifier indicating a value of ID1 and containing a Sequence Number indicating a value of SEQ1 then the DUT sends an ICMPv4 Packet containing a Type indicating a value of 14 (Timestamp Reply)

	and containing a Code indicating a value of 0 and containing a valid Checksum and containing an Identifier indicating a value of ID1 and containing a Sequence Number indicating a value of SEQ1
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Timestamp Message 2. DUT: Send ICMPv4 Timestamp Reply
Pass Criteria	The DUT sends an ICMPv4 Packet containing a Type indicating a value of 14 (Timestamp Reply) and containing a Code indicating a value of 0 and containing a valid Checksum and containing an Identifier indicating a value of ID1 and containing a Sequence Number indicating a value of SEQ1
Test Iterations	
Reference	Derived from RFC792, RFC1122 3.2.2.8
Notes	OPTIONAL

ICMPv4_TYPE_18: Send ICMP Destination Unreachable for unknown protocol

Synopsis	Ensure that when a DUT receives an IPv4 Packet containing an IPv4 Header containing a Protocol indicating a value of <unsupportedProtocol> then the DUT sends an ICMPv4 Packet containing a Type indicating a value of 3 (Destination Unreachable) and containing a Code indicating a value of 2 (Protocol unreachable)
Prerequisites	None
Test setup	Topology 1

Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 Packet with protocol value of <unsupportedProtocol> 2. DUT: Sends ICMPv4 Destination Unreachable message indicating Protocol Unreachable
Pass Criteria	The DUT sends an ICMPv4 Packet containing a Type indicating a value of 3 (Destination Unreachable) and containing a Code indicating a value of 2 (Protocol unreachable)
Test Iterations	
Reference	Derived from RFC792, RFC1122 3.2.2.1
Notes	

DEPRECATED - ICMPv4_TYPE_21: Ensure that the DUT does not generate an ICMPv4 Time Exceeded message when reassembly fails due to the absence of the first fragment

Synopsis	Ensure that when a DUT receives an IPv4 Packets containing an IPv4 Header containing Flags containing DF flag indicating a value of 0 containing MF flag indicating a value of 1 and containing Offset indicating a value not equal to 0 then the DUT does not send an ICMPv4 Packet containing a Type indicating a value of 11
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 Packet with DF=1 (Don't Fragment) and MF=1 (More Fragments) and an Offset not equal to 0 2. DUT: Does not send an ICMPv4 Time Exceeded message
Pass Criteria	The DUT does not send an ICMPv4 Packet containing a Type indicating a value of 11
Test Iterations	
Reference	Derived from RFC792, RFC1122 3.2.2.4
Notes	

ICMPv4_TYPE_22: Send ICMP Echo Reply on receiving ICMP Echo Request

Synopsis	A host must respond to all ICMP Echo Requests sent to it, by sending an ICMP Echo Reply back to the sender of ICMP Echo Request.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section "Parameters used in the tests"
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing:

	- IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply
Pass Criteria	3. DUT: Send ICMP Echo Reply
Reference	RFC 792 p15 Echo or Echo Reply Message (MUST)
Notes	

4.4 Internet Protocol Version 4 (IPv4)

4.4.1 General

4.4.1.1 Referenced specification

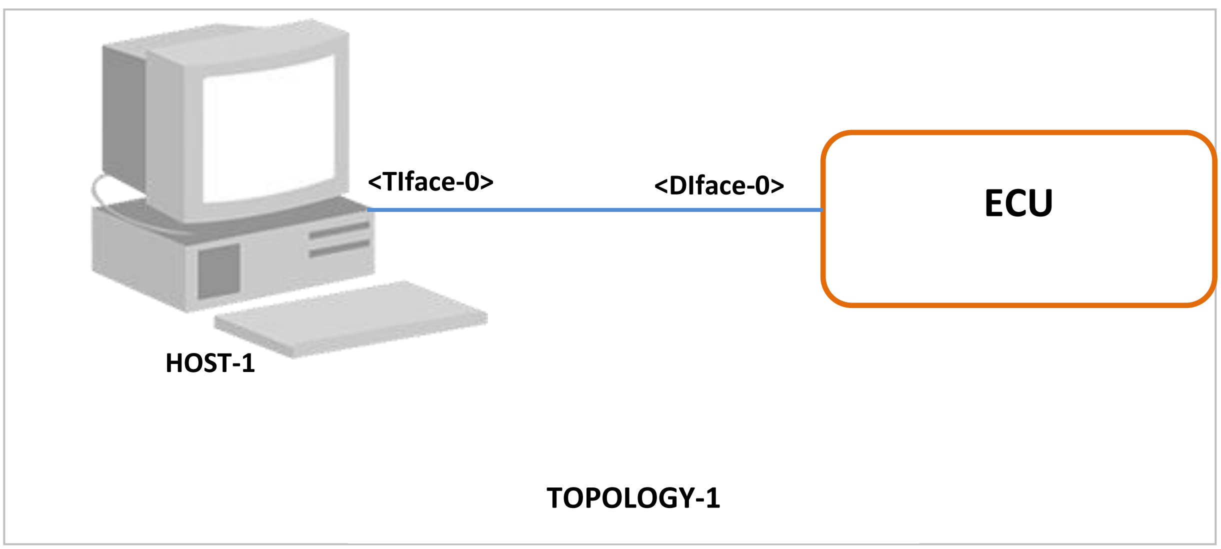
The scope of this chapter is to specify test cases for the Internet Protocol Version 4 (IPv4) from the following standards:

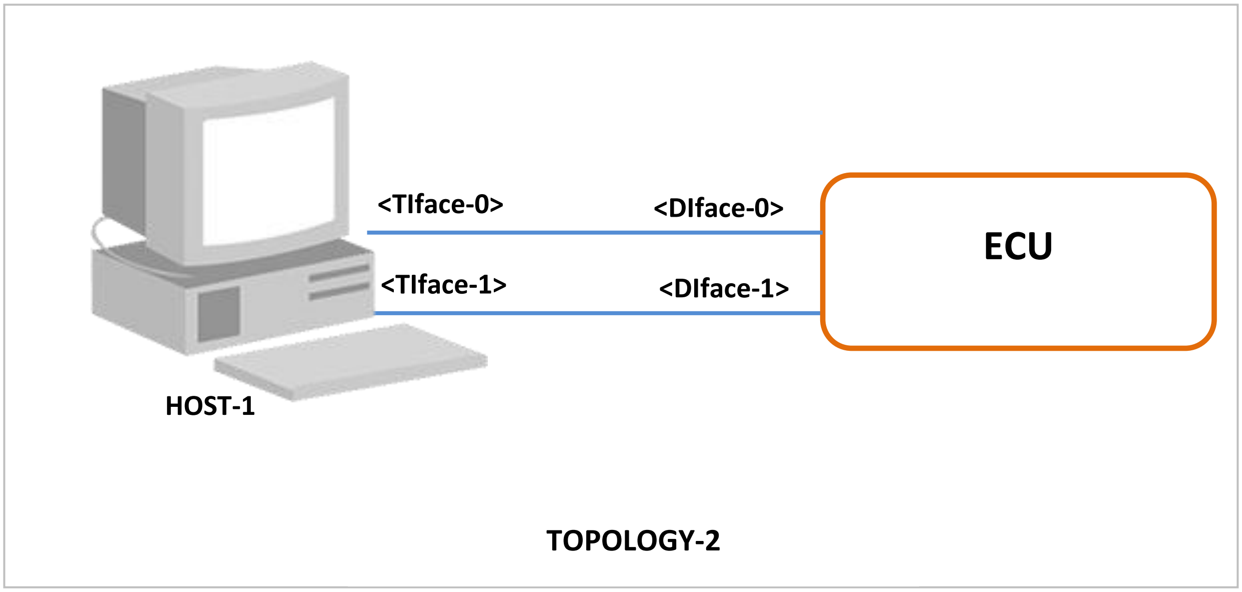
- RFC 791 - INTERNET PROTOCOL, DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION
- RFC 1122 - Requirements for Internet Hosts - Communication Layers

Though the focus of conformance testing has been limited to the above mentioned specification documents we have relied heavily upon the following document as specification for transmission of IP Datagrams over Ethernet Options:

- RFC 894 - A Standard for the Transmission of IP Datagrams over Ethernet Networks

4.4.1.2 Simulated topologies





4.4.1.3 Required topology related configuration

- This test suite expects to be running against an IP stack

4.4.1.4 Coverage

Specification Document	Section Number	Test Category	Test Number(s)
RFC 791	3.1	IPv4 Header	IPv4_HEADER_01 to IPv4_HEADER_05, IPv4_HEADER_08, IPv4_HEADER_09
RFC 791	3.1	IPv4 Checksum	IPv4_CHECKSUM_01, IPv4_CHECKSUM_02, IPv4_CHECKSUM_04, IPv4_CHECKSUM_05
RFC 791	3.1	IPv4 Time to Live	IPv4_TTL_01 , IPv4_TTL_03 to IPv4_TTL_05
RFC 791	3.1	IPv4 Version Number	IPv4_VERSION_01, IPv4_VERSION_03, IPv4_VERSION_04
RFC 791	3.2	IPv4 Addressing	IPv4_ADDRESSING_01 to IPv4_ADDRESSING_03
RFC 791	2.3	IPv4 Fragments	IPv4_FRAGMENTS_1 to IPv4_FRAGMENTS_5
RFC 791	3.2	IPv4 Options	IPv4_OPTIONS_1 to IPv4_OPTIONS_14
RFC 791	3.2	IPv4 Reassembly	IPv4_REASSEMBLY_01 to IPv4_REASSEMBLY_11

4.4.2 Parameters used in the tests

Parameter used in test	Description
<validTTL>	Time To Live field value used in IP packet to be sent. If the packet is to be forwarded then this value must be greater than 2
<invalidChecksum>	This is the checksum which is different from the calculated checksum, i.e different from 16 bit one's complement of the one's complement sum of all 16 bit words in the header..
<id> - <id1> <id2>	Identification of the IP packet
<LargeTTLValue>	Time To Live value in a Fragmented packet. This value is greater than the initial timer setting which is 15 seconds
<LowTTLValue>	Time To Live value in a Fragmented packet. This value is less than the initial timer setting which is 15 seconds
<non-reserve-value>	A value which is not zero.
<limitedBroadcastAddress>	The limited broadcast address addresses every host on the connected physical network. {-1, -1} -> 255.255.255.255
<directedBroadcastAddress>	The directed broadcast address addresses a specific group in the network. {<Network-number>, -1}, e.g. 192.168.255.255
<loopBackAddress>	The internal host loopback address, e.g. 127.0.0.1
<MTU>	The Maximum Transmission Unit size.

NOTE: As a general test pattern it can be assumed that all events will be guarded by a timer, e.g. receiving or non-receiving of message. Due to the difference nature of guarding, different default values can be assumed:

- Guarding user interactions: 30 seconds
- Guarding awaiting of response: 3 seconds
- Guarding awaiting of no response: 10 seconds

Protocol should be mentioned in the respective test cases.

4.4.3 IPv4 Test cases

4.4.3.1 IPv4 Header

IPv4_HEADER_01: Ensure that the DUT generates an IPv4 Packet with a Total Length greater than or equal to 20.

Synopsis	Ensure that when the DUT is requested to generate an IPv4 packet, then the DUT generates an IPv4 Packet containing an IPv4 Header containing a Total Length indicating a value greater than or equal to 20.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request. 2. DUT: Generates an ICMPv4 Echo Reply.
Pass Criteria	The DUT generates an IPv4 Packet containing an IPv4 Header containing a Total Length indicating a value greater than or equal to 20
Test Iterations	
Reference	Derived from RFC791, section 3.1
Notes	

IPv4_HEADER_02: Ensure that the DUT discards an IPv4 Packet with an invalid Header Length

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a Header Length indicating a value less than 20, then the DUT discards the IPv4 Packet silently.
Prerequisites	None
Test setup	Topology 1

Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with a header length indicating a value less than 20 2. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT discards the IPv4 Packet silently.
Test Iterations	
Reference	Derived from RFC791, section 3.1
Notes	

IPv4_HEADER_03: Ensure that the DUT generates an IPv4 Packet with the Source Address being one of its IPv4 Addresses

Synopsis	Ensure that when the DUT is requested to generate an IPv4 packet, then the DUT sends an IPv4 Packet containing an IPv4 Header containing a Source Address indicating one of its defined IPv4 addresses.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request 2. DUT: Generates an ICMPv4 Echo Reply with Source Address being one of its defined IPv4 addresses.
Pass Criteria	The DUT sends an IPv4 Packet containing an IPv4 Header containing a Source Address indicating one of its defined IPv4 addresses.
Test Iterations	
Reference	Derived from RFC791 section 3.1, 3.2, RFC1122 section 3.2.1.3
Notes	

IPv4_HEADER_04: Ensure that the DUT discards an IPv4 Packet with an incorrect Destination Address

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a Destination Address indicating a value different from the DUT’s IPv4 address and is not a Broadcast or Multicast address, then the DUT discards the IPv4 Packet silently.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with destination address different than the DUT’s IPv4 address 2. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT discards the IPv4 Packet silently.
Test Iterations	
Reference	Derived from RFC791 section 3.1, 3.2, RFC1122 section 3.2.1.3
Notes	

IPv4_HEADER_05: IP Maximum datagram length check

Synopsis	All hosts must be prepared to accept datagrams of up to 576 octets.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Total Length field set to 576 - IP Payload field, containing: - 556 bytes data 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent
Test Iterations	
Notes	Derived from RFC 791 s3.1 p13 Internet Header Format (MUST)

IPv4_HEADER_08: IP Header length validation

Synopsis	Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5. (Note: Tests that DUT discards a packet with total length smaller than implied by IHL value)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Data field set to "ECU NETWORK VALIDATION TEST" - IP IHL field set to 13 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Does not Send ICMP Echo Reply
Pass Criteria	3. DUT: Does not Send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p11 Internet Header Format (MUST)

IPv4_HEADER_09: IP Total Length validation

Synopsis	Total Length is the length of the datagram, measured in octets, including internet header and data. (Note: Tests that DUT discards a packet with total length bigger than the actual transmitted data)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Data field set to "ECU NETWORK VALIDATION TEST" - IP Total Length field set to 48 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Does not Send ICMP Echo Reply
Pass Criteria	3. DUT: Does not Send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p13 Internet Header Format (MUST)

4.4.3.2 IPv4 Checksum

DEPRECATED – IPv4_CHECKSUM_01: Ensure that the DUT generates an IPv4 Packet with a valid Header Checksum

Synopsis	Ensure that when the DUT generates an IPv4 packet then the DUT sends an IPv4 Packet containing an IPv4 Header containing a Header Checksum indicating a valid checksum.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request 2. DUT: Sends an ICMPv4 Echo Reply containing a valid checksum
Pass Criteria	The DUT sends an IPv4 Packet containing an IPv4 Header containing a Header Checksum indicating a valid checksum.
Test Iterations	
Reference	Derived from RFC791 section 3.1, RFC1122 section 3.2.1.2
Notes	Deprecated in version 2. - Test case is duplicated to IPv4_CHECKSUM_05.

IPv4_CHECKSUM_02: IP Checksum method validation on receiving

Synopsis	If the header checksum fails, the internet datagram is discarded at once by the entity which detects the error.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	<invalidChecksum> Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with Header Checksum indicating <invalidChecksum> 2. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT discards the IPv4 Packet silently.
Test Iterations	
Reference	Derived from RFC791 section 3.1, RFC1122 section 3.2.1.2
Notes	

IPv4_CHECKSUM_04: IP Checksum method validation on sending

Synopsis	The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero. (Note: Here we are verifying that DUT uses same checksum calculation method as we are expecting)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to send a IP packet <DIface-0>, containing: - IP Destination Address set to address of host-1 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send a IP packet 4. TESTER: Verify that the received IP packet contains: - IP Checksum field set to "16 bit one's complement of the one's complement sum of all 16 bit words in the header"
Pass Criteria	3. DUT: Send an IP Packet 4. TESTER: Verify that the received IP packet contains: - IP Checksum field set to "16 bit one's complement of the one's complement sum of all 16 bit words in the header"
Test Iterations	
Notes	Derived from RFC 791 s3.1 p14 Internet Header Format (MUST) Request / Reply Mechnism has to be supported by DUT and a packet may be an ICMP Request or UDP datagram on a port DUT is listening.

IPv4_CHECKSUM_05: IP Checksum method validation

Synopsis	The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero. (Note: Here we send an Echo Request with checksum calculated according to rfc. DUT receives this Echo Request, verifies the Echo Request and then sends Echo Reply. We then verify that DUT uses the checksum calculation method according to rfc)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Checksum field set to "16 bit one's complement of the one's complement sum of all 16 bit words in the header" 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that the received ICMP Echo Reply contains: - IP Checksum field to "16 bit one's complement of the one's complement sum of all 16 bit words in the header"
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that the received ICMP Echo Reply contains: - IP Checksum field to "16 bit one's complement of the one's complement sum of all 16 bit words in the header"
Test Iterations	
Notes	Derived from RFC 791 s3.1 p14 Internet Header Format (MUST)

4.4.3.3 IPv4 Time to Live

IPv4_TTL_01: A host MUST NOT send a datagram with a Time-to-Live (TTL) value of zero

Synopsis	A host MUST NOT send a datagram with a Time-to-Live (TTL) value of zero.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request packet 2. DUT: Sends an ICMPv4 Echo Reply packet with a TTL value greater than 0
Pass Criteria	The DUT sends an IPv4 packet containing an IPv4 Header containing a TTL indicating a value greater than 0.
Test Iterations	
Reference	Derived from RFC791 section 3.1 and 3.2, RFC1122 section 3.2.1.7
Notes	

DEPRECATED - IPv4_TTL_03: Ensure that the DUT decrements the TTL of a forwarded IPv4 Packet by at least 1

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a TTL indicating a value TTL1 then the DUT forwards and sends an IPv4 packet containing an IPv4 Header containing a TTL indicating a value less than TTL1.
Prerequisites	None
Test setup	Topology 2
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 packet with a TTL value TTL1 which is greater than 2 2. DUT: Sends an IPv4 packet with a TTL value less than TTL1
Pass Criteria	The DUT forwards and sends an IPv4 packet containing an IPv4 Header containing a TTL indicating a value less than TTL1.
Test Iterations	
Reference	Derived from RFC791 section 3.1 and 3.2, RFC1122 section 3.2.1.7
Notes	Deprecated in version 2 - test case only applicable for router!

DEPRECATED - IPv4_TTL_04: Ensure that the DUT discards an IPv4 packet with a TTL of 0 if the IPv4 packet has to be forwarded

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a TTL indicating a value 0 then the DUT does not forwards and discards the IPv4.
Prerequisites	None
Test setup	Topology 2
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with a TTL value of 0 which has to be forwarded 2. DUT: Does not forward the ICMPv4 Echo Request
Pass Criteria	The DUT does not forwards and discards the IPv4.
Test Iterations	
Reference	Derived from RFC791 section 3.1 and 3.2, RFC1122 section 3.2.1.7
Notes	Deprecated in version 2- test case only applicable for router!

IPv4_TTL_05: Packets with 0 or 1 TTL are not discarded by hosts

Synopsis	A host MUST NOT discard a datagram just because it was received with TTL less than 2.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP TTL field set to <ttl> 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply
Pass Criteria	3. DUT: Send ICMP Echo Reply
Test Iterations	1. CASE: <ttl> = 0 2. CASE: <ttl> = 1
Notes	Derived from RFC 1122 s3.2.1.7 p34 Time-to-Live: RFC-791 Section 3.2 (MUST)

4.4.3.4 IPv4 Version Number

IPv4_VERSION_01: Ensure that the DUT accepts an IPv4 Packet with a valid Version 4

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a Version indicating a value of 4, then the DUT accepts the IPv4 Packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with version 4 2. DUT: Sends an ICMPv4 Echo Reply
Pass Criteria	The DUT accepts the IPv4 Packet and replies correctly with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791, section 3.1, RFC1122, section 3.2.1.1
Notes	

IPv4_VERSION_03: Ensure that the DUT generates a IPv4 Packet with a valid Version 4

Synopsis	Ensure that when the DUT generates an IPv4 packet then the DUT sends an IPv4 Packet containing an IPv4 Header containing a Version indicating a value of 4.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request 2. DUT: Does not send an ICMPv4 Echo Reply containing a valid version 4
Pass Criteria	The DUT sends an IPv4 Packet containing an IPv4 Header containing a Version indicating a value of 4.
Test Iterations	
Reference	Derived from RFC791, section 3.1, RFC1122, section 3.2.1.1
Notes	

IPv4_VERSION_04: IP Version validation

Synopsis	A datagram whose version number is not 4 MUST be silently discarded.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Version field set to other than <IP_VERSION_4> 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Discard ICMP Echo Request and do not send ICMP Echo Reply
Pass Criteria	3. DUT: Discard ICMP Echo Request and do not send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p11 Internet Header Format (Version), RFC 1122 s3.2.1.1 p29 Version Number: RFC-791 Section 3.1 (MUST)

4.4.3.5 IPv4 Addressing

IPv4_ADDRESSING_01: Ensure that the DUT receives an IPv4 Packet with a Destination Address being a Limited Broadcast Address

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a Destination Address indicating a value of Limited Broadcast, then the DUT accepts the IPv4 Packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	<limitedBroadcastAddress> Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with destination address being <limitedBroadcastAddress> 2. DUT: Sends an ICMPv4 Echo Reply
Pass Criteria	The DUT accepts the IPv4 Packet and answers correctly with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791 section 3.2, RFC1122 section 3.2.1.3
Notes	

IPv4_ADDRESSING_02: Ensure that the DUT discards an IPv4 Packet with a Destination Address being a Directed Broadcast Address

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a Destination Address indicating a value of Directed Broadcast, then the DUT discards the IPv4 Packet silently.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	<directedBroadcastAddress> Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with destination address being <directedBroadcastAddress> 2. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT discards the IPv4 Packet silently.
Test Iterations	
Reference	Derived from RFC791 section 3.2, RFC1122 section 3.2.1.3
Notes	

IPv4_ADDRESSING_03: Ensure that the DUT discards an IPv4 Packet with a Destination Address being a Loop Back Address

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing a Destination Address indicating a value of Loop Back, then the DUT discards the IPv4 packet silently.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	<loopBackAddress> Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with destination address being <loopBackAddress> 2. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT discards the IPv4 packet silently.
Test Iterations	
Reference	Derived from RFC791 section 3.2, RFC1122 section 3.2.1.3
Notes	

4.4.3.6 IPv4 Fragments

IPv4_FRAGMENTS_01: IP Reconstruct fragments validation

Synopsis	To assemble the fragments of an internet datagram, an internet protocol module (for example at a destination host) combines internet datagrams that all have the same value for the four fields: identification, source, destination, and protocol.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<id> Check section general Input Parameters
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to zero - Flags, containing: - MF bit set to 1 - first half of the constructed ICMP packet 2. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags, containing: - MF bit set to zero - last half of the constructed ICMP packet 3. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 4. DUT: Send ICMP Echo Reply 5. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Pass Criteria	4. DUT: Send ICMP Echo Reply 5. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Test Iterations	
Notes	Derived from RFC 791 s2.3 p9 Function Description (Fragmentation), RFC 791 s3.2 p29 Discussion (Identification), RFC 1122 s3.3.2 p56 Reassembly (MUST)

IPv4_FRAGMENTS_02: IP Reconstruct fragments, negative test on id

Synopsis	To assemble the fragments of an internet datagram, an internet protocol module (for example at a destination host) combines internet datagrams that all have the same value for the four fields: identification, source, destination, and protocol. (Note: This test verifies that IP module does not assemble the fragments if identification is different).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id1> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - first half of the constructed ICMP packet 2. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id2> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - last half of the constructed ICMP packet 3. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 4. DUT: Do not send ICMP Echo Reply 5. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id1> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - last half of the constructed ICMP packet 6. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 7. DUT: Send ICMP Echo Reply 8. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two

	fragments. 9. TESTER: Wait for <ipIniReassembleTimeout> seconds 10. NOTE: TESTER has to wait for <ipIniReassembleTimeout> seconds because when TESTER is sending the fragment with different identification, the fragment gets stored in DUT's reassembly buffer. If this fragment is not removed from the buffer then it may lead to a scenario where this fragment may be mistaken for a fragment-1 of another test case. So a timeout needs to occur for this fragment.
Pass Criteria	4. DUT: Do not send ICMP Echo Reply 7. DUT: Send ICMP Echo Reply 8. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Test Iterations	
Notes	Derived from RFC 791 s2.3 p9 Function Description (Fragmentation) (MUST)

IPv4_FRAGMENTS_03: IP Reconstruct fragments, negative test on source

Synopsis	To assemble the fragments of an internet datagram, an internet protocol module (for example at a destination host) combines internet datagrams that all have the same value for the four fields: identification, source, destination, and protocol. (Note: This test verifies that IP module does not assemble the fragments if source is different).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<id> Check section general Input Parameters
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - first half of the constructed ICMP packet 2. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to different address from host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - last half of the constructed ICMP packet 3. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 4. DUT: Do not send ICMP Echo Reply 5. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - last half of the constructed ICMP packet 6. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0>

	7. DUT: Send ICMP Echo Reply 8. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments. 9. TESTER: Wait for <ipIniReassembleTimeout> seconds 10. NOTE: TESTER has to wait for <ipIniReassembleTimeout> seconds because when TESTER is sending the fragment with different source address, the fragment gets stored in DUT's reassembly buffer. If this fragment is not removed from the buffer then it may lead to a scenario where this fragment may be mistaken for a fragment-1 of another test case. So a timeout needs to occur for this fragment.
Pass Criteria	4. DUT: Do not send ICMP Echo Reply 7. DUT: Send ICMP Echo Reply 8. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Test Iterations	
Notes	Derived from RFC 791 s2.3 p9 Function Description (Fragmentation) (MUST)

IPv4_FRAGMENTS_04: IP Reconstruct fragments, negative test on protocol

Synopsis	To assemble the fragments of an internet datagram, an internet protocol module (for example at a destination host) combines internet datagrams that all have the same value for the four fields: identification, source, destination, and protocol. (Note: This test verifies that IP module does not assemble the fragments if protocol is different).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<id> Check section general Input Parameters
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - first half of the constructed ICMP packet 2. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Protocol field set to <ipTypeTCP> - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - last half of the constructed ICMP packet 3. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 4. DUT: Do not send ICMP Echo Reply 5. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Identification field set to <id> - Protocol field set to <ipTypeICMP> - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - last half of the constructed ICMP packet 6. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 7. DUT: Send ICMP Echo Reply

	8. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments. 9. TESTER: Wait for <ipIniReassembleTimeout> seconds 10. NOTE: TESTER has to wait for <ipIniReassembleTimeout> seconds because when TESTER is sending the fragment with different protocol, the fragment gets stored in DUT's reassembly buffer. If this fragment is not removed from the buffer then it may lead to a scenario where this fragment may be mistaken for a fragment-1 of another test case. So a timeout needs to occur for this fragment.
Pass Criteria	4. DUT: Do not send ICMP Echo Reply 7. DUT: Send ICMP Echo Reply 8. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Test Iterations	
Notes	Derived from RFC 791 s2.3 p9 Function Description (Fragmentation) (MUST)

IPv4_FRAGMENTS_05: IP send unfragmented data validation

Synopsis	The fragmentation strategy is designed so than an unfragmented datagram has all zero fragmentation information (MF = 0, fragment offset = 0).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	Test step 1. TESTER: Cause DUT to send a Message with <UDPDefaultData> from <DIface-0> with - Source IP Address set to <DIface-0-IP> - Destination IP Address set to <HOST-1-IP> - Source UDP Port field set to <unusedUDPDstPort1> (20001) - Destination UDP Port field set to <unusedUDPSrcPort> (20000) 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Request 4. TESTER: Verify that the received packet contains: - IP Flags field, containing: - MF bit set to zero - IP Fragment Offset field set to zero
Pass Criteria	3. DUT: Send ICMP Echo Request 4. TESTER: Verify that the received packet contains: - IP Flags field, containing: - MF bit set to zero - IP Fragment Offset field set to zero
Test Iterations	
Notes	Derived from RFC 791 s3.2 p25 Discussion (MUST) The message can be an ICMP Echo Request or a simple UDP message.

4.4.3.7 IPv4 Reassembly

DEPRECATED – IPv4_REASSEMBLY_01: Ensure that the DUT reassembles fragments of an IPv4 Packet

Synopsis	Ensure that when the DUT receives a series of IPv4 Fragments of a large IPv4 packet then the DUT reassembles and accepts the IPv4 packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset = 0, containing an ICMPv4 Echo Request where the Total length is less than the <MTU> 2. TESTER: Send several IPv4 packets as continuation of the ICMPv4 Echo Request with MF = 1 (More Fragments) and Offset incremented each time by the payload size of the previous packet, and the ID being the same as the first fragment 3. TESTER: Send an IPv4 packet with MF = 0 (Last Fragment) and Offset incremented by the payload size of the previous packet, and the ID being the same as the first fragment 4. DUT: Sends an ICMPv4 Echo Reply
Pass Criteria	The DUT reassembles and accepts the IPv4 packet and answers correctly with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	Deprected in version 2 - test case is duplicated to IPv4_FRAGMENTS_01.

DEPRECATED – IPv4_REASSEMBLY_02: Ensure that the DUT does not reassemble fragments of an IPv4 Packet with different IDs

Synopsis	Ensure that when the DUT receives a series of IPv4 Fragments of a large IPv4 packet containing an IPv4 Header containing an ID indicating different values then the DUT does not reassemble and accept the IPv4 packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset = 0, containing an ICMPv4 Echo Request where the Total length is less than the <MTU> 2. TESTER: Send an IPv4 packet with), MF = 0 (Last Fragment) and Offset incremented by the payload size of the previous packet, and the ID being different from the first fragment 3. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT does not reassemble and accept the IPv4 packet and does not answer with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	Deprected in version 2 - test case is duplicated to IPv4_FRAGMENTS_02.

DEPRECATED - IPv4_REASSEMBLY_03: Ensure that the DUT does not reassemble fragments of an IPv4 Packet with different Protocols

Synopsis	Ensure that when the DUT receives a series of IPv4 Fragments of a large IPv4 packet containing an IPv4 Header containing a Protocol indicating different values then the DUT does not reassemble and accept the IPv4 packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset = 0, containing an ICMPv4 Echo Request where the Total length is less than the <MTU> 2. TESTER: Send an IPv4 packet with MF = 0 (Last Fragment) and Offset incremented by the payload size of the previous packet, and the Protocol being different from the first fragment 3. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT does not reassemble and accept the IPv4 packet and does not answer with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	Deprected in version 2 - test case is duplicated to IPv4_FRAGMENTS_04.

IPv4_REASSEMBLY_04: Ensure that the DUT reassembles fragments of an IPv4 Packet received in the wrong order

Synopsis	Ensure that when the DUT receives a series of unordered IPv4 Fragments of a large IPv4 packet then the DUT reassembles and accept the IPv4 packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset = 0, containing an ICMPv4 Echo Request where the Total length is less than the <MTU> 2. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset correctly incremented, but belonging to the third fragment, and the ID being the same as the first fragment 3. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset correctly incremented, but belonging to the second fragment, and the ID being the same as the first fragment 4. TESTER: Send an IPv4 packet with MF = 0 (Last Fragment) and Offset correctly incremented, and the ID being the same as the first fragment 5. DUT: Sends an ICMPv4 Echo Reply
Pass Criteria	The DUT reassembles and accept the IPv4 packet and answers correctly with an ICMPv4 Echo Request.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	

DEPRECATED - IPv4_REASSEMBLY_05: Ensure that the DUT discards a duplicate of an IPv4Fragment during the reassembly

Synopsis	Ensure that when the DUT receives a series of IPv4 Fragments of a large IPv4 packet including a duplicated fragment then the DUT discards the duplicated fragment, reassembles and accepts the IPv4 packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset = 0, containing an ICMPv4 Echo Request where the Total length is less than the <MTU> 2. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset correctly incremented, and the ID being the same as the first fragment 3. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset indicating the same value as in test step 2, and the ID being the same as the first fragment 4. TESTER: Send an IPv4 packet with MF = 0 (Last Fragment) and Offset correctly incremented, and the ID being the same as the first fragment 5. DUT: Sends an ICMPv4 Echo Reply, not including any data from the fragment sent in test step 3
Pass Criteria	The DUT discards the duplicated fragment, reassembles and accepts the IPv4 packet and answers correctly with an ICMPv4 Echo Reply not including any data from the duplicated fragment.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	Deprected in version 2 - test case is duplicated to IPv4_ REASSEMBLY_13.

IPv4_REASSEMBLY_06: Ensure that the DUT does not reassemble fragments of an IPv4 Packet if no first fragment is sent

Synopsis	Ensure that when the DUT receives a series of IPv4 Fragments of a large IPv4 packet containing an IPv4 Header containing an Offset indicating a value different than 0 but with correct increment then the DUT does not reassemble and accept the IPv4 packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset not equal to 0, containing an ICMPv4 Echo Request where the Total length is less than the MTU 2. TESTER: Send an IPv4 packet with MF = 0 (Last Fragment) and Offset correctly incremented, and the ID being the same as the first fragment 3. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT does not reassemble and accept the IPv4 packet and does not answer with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	

IPv4_REASSEMBLY_07: Ensure that the DUT does not reassemble fragments of an IPv4 Packet if some IPv4 Fragments are missing

Synopsis	Ensure that when the DUT receives a series of IPv4 Fragments of a large IPv4 packet including the first and the last fragment but missing some in between then the DUT does not reassemble and accept the IPv4 packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset = 0, containing an ICMPv4 Echo Request where the Total length is less than the <MTU> 2. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset correctly incremented, and the ID being the same as the first fragment 3. TESTER: Send an IPv4 packet with MF = 0 (Last Fragment) and Offset is incremented as if one more fragment would have been sent, and the ID being the same as the first fragment 4. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT does not reassemble and accept the IPv4 packet and does not answer with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	

DEPRECATED - IPv4_REASSEMBLY_08: Ensure that the DUT does not reassemble fragments of an IPv4 Packet if no last fragment is sent

Synopsis	Ensure that when the DUT receives a series of IPv4 Fragments of a large IPv4 packet but no last fragment then the DUT does not reassemble and accept the IPv4 packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset = 0, containing an ICMPv4 Echo Request where the Total length is less than the <MTU> 2. TESTER: Send an IPv4 packet with MF = 1 (More Fragments) and Offset correctly incremented, and the ID being the same as the first fragment 3. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT does not reassemble and accept the IPv4 packet and does not answer with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	Deprected in version 2 - test case is duplicated to IPv4_REASSEMBLY_10.

IPv4_REASSEMBLY_09: Ensure that DUT discards IPv4 Packet MF = 1

Synopsis	Ensure that when the DUT receives an IPv4 packet containing an IPv4 Header containing Flags and containing a MF flag indicating that there are more fragments coming: MF = 1, then the DUT discards the IPv4 Packet silently.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMPv4 Echo Request with MF = 1 (More Fragments) 2. DUT: Does not send an ICMPv4 Echo Reply
Pass Criteria	The DUT discards the IPv4 Packet silently and does not answer with an ICMPv4 Echo Reply.
Test Iterations	
Reference	Derived from RFC791 section 3.2
Notes	

IPv4_REASSEMBLY_10: IP Reassembly default time check

Synopsis	The current recommendation for the initial timer setting is 15 seconds.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - TTL field set to 15 - first half of the constructed ICMP packet whose size is multiple of 8-octets 2. TESTER: Wait for (<ipIniReassembleTimeout> - <ParamToleranceTime>) seconds 3. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - TTL field set to 15 - last half of the constructed ICMP packet 4. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 5. DUT: Send ICMP Echo Reply 6. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments. 7. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - TTL field set to 15 - first half of the constructed ICMP packet which is multiple of 8-octets 8. TESTER: Wait for (<ipIniReassembleTimeout> + <ParamToleranceTime>) seconds 9. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1

	- Destination Address field set to address of DUT - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - TTL field set to 15 - last half of the constructed ICMP packet 10. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 11. DUT: Do not send ICMP Echo Reply 12. TESTER: Wait for <ipIniReassembleTimeout> seconds 13. NOTE: TESTER has to wait for <ipIniReassembleTimeout> seconds so that timeout occurs for fragment-2 stored in DUT's reassembly buffer. If this does not happen then it may lead to a scenario where fragment-2 stored in the DUT's reassembly buffer may be mistaken for a fragment-1 of another test case. So a timeout needs to occur for fragment-2.
Pass Criteria	5. DUT: Send ICMP Echo Reply 6. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments. 11. DUT: Do not send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.2 p27 Discussion (An Example Reassembly Procedure) (SHOULD)

IPv4_REASSEMBLY_11: Check fragment with Large TTL value

Synopsis	The initial setting of the timer is a lower bound on the reassembly waiting time. This is because the waiting time will be increased if the Time to Live in the arriving fragment is greater than the current timer value. (Note: Here we are assuming that initial timer setting is 15 seconds)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<LargeTTLValue> Check section general Input Parameters
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - TTL field set to <LargeTTLValue> - first half of the constructed ICMP packet which is multiple of 8-octets 2. TESTER: Wait for 15 seconds 3. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - TTL field set to <LargeTTLValue> - last half of the constructed ICMP packet 4. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 5. DUT: Send ICMP Echo Reply 6. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Pass Criteria	5. DUT: Send ICMP Echo Reply 6. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Test Iterations	
Notes	Derived from RFC 791 s3.2 p27 Discussion (An Example Reassembly Procedure) (MUST)

IPv4_REASSEMBLY_12: Check fragment with Low TTL value

Synopsis	The initial setting of the timer is a lower bound on the reassembly waiting time. The waiting time will not be decreased if it is less the Time to Live in the arriving fragment.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<LowTTLValue> Check section general Input Parameters
Test Procedure	1. TESTER: Construct an ICMP Echo Request. Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - TTL field set to <LowTTLValue> - first half of the constructed ICMP packet which is multiple of 8-octets 2. TESTER: Wait for <LowTTLValue> seconds 3. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to data size sent in first IP packet in unit of 8-octets - Flags field, containing: - MF bit set to zero - TTL field set to <LowTTLValue> - last half of the constructed ICMP packet 4. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 5. DUT: Send ICMP Echo Reply 6. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Pass Criteria	5. DUT: Send ICMP Echo Reply 6. TESTER: Verify that Identifier, Sequence Number and Data of ICMP Echo Reply are same as those of ICMP Echo Request sent in two fragments.
Test Iterations	
Notes	Derived from RFC 791 s3.2 p27 Discussion (An Example Reassembly Procedure) (MUST)

IPv4_REASSEMBLY_13: IP Fragments overlap check

Synopsis	In the case that two or more fragments contain the same data either identically or through a partial overlap, this procedure will use the more recently arrived copy in the data buffer and datagram delivered.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Construct an ICMP Echo Request, containing: - Data field set to "ECU NETWORK VALIDATION TEST" Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to zero - Flags field, containing: - MF bit set to 1 - first 16 octets of the constructed ICMP packet 2. TESTER: Listen (for upto <FragReassemlyTimeout>/4 seconds) on <DIface-0> 3. DUT: Do not send ICMP Echo Reply 4. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to 2 - Flags field, containing: - MF bit set to 1 - Data field set to "DUPLICATE FRAGMENTS TEST" 5. TESTER: Listen (for upto <FragReassemlyTimeout>/4 seconds) on <DIface-0> 6. DUT: Do not send ICMP Echo Reply 7. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set to 2 - Flags field, containing: - MF bit set to 1 - second 8 octets of the constructed ICMP packet 8. TESTER: Listen (for upto <FragReassemlyTimeout>/4 seconds) on <DIface-0> 9. DUT: Do not send ICMP Echo Reply 10. TESTER: Send an IP packet to <DIface-0>, containing: - Source Address field set to address of host-1 - Destination Address field set to address of DUT - Fragment Offset field set 3 - Flags field, containing: - MF bit set to zero

	- remaining portion of the constructed ICMP packet 11. TESTER: Listen (for upto <FragReassemlyTimeout>/4 seconds) on <DIface-0> 12. DUT: Send ICMP Echo Reply 13. TESTER: Verify that the received ICMP Echo Reply contains: - Data field set to "ECU NETWORK VALIDATION TEST"
Pass Criteria	3. DUT: Do not send ICMP Echo Reply 6. DUT: Do not send ICMP Echo Reply 9. DUT: Do not send ICMP Echo Reply 12. DUT: Send ICMP Echo Reply 13. TESTER: Verify that the received ICMP Echo Reply contains: - Data field set to "ECU NETWORK VALIDATION TEST"
Test Iterations	
Notes	Derived from RFC 791 s3.2 p29 Discussion (An Example Reassembly Procedure) (MUST)

4.4.3.8 IPv4 Options

IPv4_OPTIONS_01: Multiple End of Options check

Synopsis	End of Option List option indicates the end of the option list. (Note: Here we are verifying that DUT does not crash on receiving multiple End of Options)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - Four End of Option List options 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply
Pass Criteria	3. DUT: Send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p16 Internet Header Format (End of Option List), RFC 791 s3.2 p31 Options (MUST)

IPv4_OPTIONS_02: Multiple No Operation option check

Synopsis	No Operation - This option may be used between options, for example, to alignthe beginning of a subsequent option on a 32 bit boundary. (Note: Here we are verifying that DUT does not crash on receiving multiple No Operation Options)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Data field set to "ECU NETWORK VALIDATION TEST" - IP Options, containing: - One Record Route option, containing: - length field set to 7 - pointer field set to 4 - Two No Operation options - One End of Option List option 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply
Pass Criteria	3. DUT: Send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p17 Internet Header Format (No Operation) (MUST)

IPv4_OPTIONS_03: Multiple No Operation and End of Option check

Synopsis	No Operation - This option may be used between options, for example, to alignthe beginning of a subsequent option on a 32 bit boundary. (Note: Here we are verifying that DUT does not crash on receiving multiple No Operation and End of Options)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - Data field set to "ECU NETWORK VALIDATION TEST" - IP Options, containing: - One Security option - One No Operation option - One No Operation option - One End of Option List option - One End of Option List option 2. TESTER: Listen (for up to <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply
Pass Criteria	3. DUT: Send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p17 Internet Header Format (No Operation) (MUST)

IPv4_OPTIONS_04: Security Option check

Synopsis	Security Option appears at most once in a datagram.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - Two Security Options 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Do not send ICMP Echo Reply
Pass Criteria	3. DUT: Do not send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p18 Internet Header Format (Security) (MAY)

IPv4_OPTIONS_05: Overflow Bit in timestamp option check

Synopsis	The Overflow (oflw) [4 bits] is the number of IP modules that cannot register timestamps due to lack of space.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 8 - pointer field set to 9 (one timestamp sent by TESTER) - oflw field set to zero - flg field set to zero - one timestamp value 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - oflw field set to 1
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - oflw field set to 1
Test Iterations	
Notes	Derived from RFC 791 s3.1 p22 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_06: Timestamp value check

Synopsis	0 -- time stamps only, stored in consecutive 32-bit words.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 12 - pointer field set to 9 (one timestamp) - flg field set to zero - one timestamp - octets 8-11 set to zero 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - octets 8-11 set to timestamp value (non-zero)
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - octets 8-11 set to timestamp value (non-zero)
Test Iterations	
Notes	Derived from RFC 791 s3.1 p22 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_07: Ttimestamp value, internet address and sequence check

Synopsis	1 -- each timestamp is preceded with internet address of the registering entity.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 20 - pointer field set to 13 (one timestamp and one internet address) - flg field set to 1 - one internet address set to address of host-1 - one timestamp - octets 12-19 set to zero 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - octets 12-15 set to address of DUT - octets 16-19 set to timestamp value (non-zero)
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - octets 12-15 set to address of DUT - octets 16-19 set to timestamp value (non-zero)
Test Iterations	
Notes	Derived from RFC 791 s3.1 p22 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_08: Relation between timestamp & its own address

Synopsis	3 -- the internet address fields are prespecified. An IP module only registers its timestamp if it matches its own address with the next specified internet address.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 12 - pointer field set to 5 - flg field set to 3 - one internet address set to address of DUT - octets 4-7 set to zero 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - octets 4-7 set to timestamp value (non-zero)
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - octets 4-7 set to timestamp value (non-zero)
Test Iterations	
Notes	Derived from RFC 791 s3.1 p22 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_09:Relation between timestamp & its own address, negative

Synopsis	3 -- the internet address fields are prespecified. An IP module only registers its timestamp if it matches its own address with the next specified internet address.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 12 - pointer field set to 5 - flg field set to 3 - one internet address set to different address of DUT - octets 4-7 set to zero 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - octets 4-7 set to zero (DUT did not register its timestamp)
Pass Criteria	3. DUT: Send ICMP Echo Reply 4. TESTER: Verify that received ICMP Echo Reply contains: - IP Internet Timestamp option, containing: - octets 4-7 set to zero (DUT did not register its timestamp)
Test Iterations	
Notes	Derived from RFC 791 s3.1 p22 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_10: Room for timestamp check

Synopsis	If there is some room but not enough room for a full timestamp to be inserted the original datagram is considered to be in error and is discarded.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 12 - pointer field set to 11 - flg field set to zero - one timestamp value 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Discard ICMP Echo Request and do not send ICMP Echo Reply
Pass Criteria	3. DUT: Discard ICMP Echo Request and do not send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p23 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_11: Overflow bits validation

Synopsis	If the overflow count itself overflows, the original datagram is considered to be in error and is discarded.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 8 - pointer field set to 9 (one timestamp sent by TESTER) - oflw field set to 15 (oflw is a 4 bits field) - flg field set to zero - one timestamp value 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Discard ICMP Echo Request and do not send ICMP Echo Reply
Pass Criteria	3. DUT: Discard ICMP Echo Request and do not send ICMP Echo Reply
Test Iterations	
Notes	Derived from RFC 791 s3.1 p23 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_12: Check room for timestamp

Synopsis	If there is some room but not enough room for a full timestamp to be inserted, an ICMP parameter problem message may be sent to the source host.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 12 - pointer field set to 11 - flg field set to zero - one timestamp value 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 22 (Basic IP Header length (20) + third octet (pointer field) of Internet Timestamp option)
Pass Criteria	3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 22 (Basic IP Header length (20) + third octet (pointer field) of Internet Timestamp option)
Test Iterations	
Notes	Derived from RFC 791 s3.1 p23 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_13: Validate overflow bits

Synopsis	If the overflow count itself overflows, an ICMP parameter problem message may be sent to the source host.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - One Internet Timestamp option, containing: - length field set to 8 - pointer field set to 9 (one timestamp sent by TESTER) - oflw field set to 15 (oflw is a 4 bits field) - flg field set to zero - one timestamp value 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 23 (Basic IP Header length (20) + fourth octet (oflw field))
Pass Criteria	3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 23 (Basic IP Header length (20) + fourth octet (oflw field))
Test Iterations	
Notes	Derived from RFC 791 s3.1 p23 Internet Header Format (Internet Timestamp) (MAY)

IPv4_OPTIONS_14: Timestamp option occurrence check

Synopsis	The timestamp option appears at most once in a datagram.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an ICMP Echo Request to <DIface-0>, containing: - IP Source Address field set to address of host-1 - IP Destination Address field set to address of DUT - IP Options, containing: - Two Internet Timestamp options, each containing: - length field set to 12 - pointer field set to 9 - flg field set to zero - one timestamp value 2. TESTER: Listen (for upto <ListenTime> seconds) on <DIface-0> 3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 32 (Basic IP Header length (20) + one internet Timestamp option length (12))
Pass Criteria	3. DUT: Send ICMP Parameter Problem Message 4. TESTER: Verify that the received ICMP Parameter Problem Message contains: - Pointer field set to 32 (Basic IP Header length (20) + one internet Timestamp option length (12))
Test Iterations	
Notes	Derived from RFC 791 s3.1 p23 Internet Header Format (Internet Timestamp) (MAY)

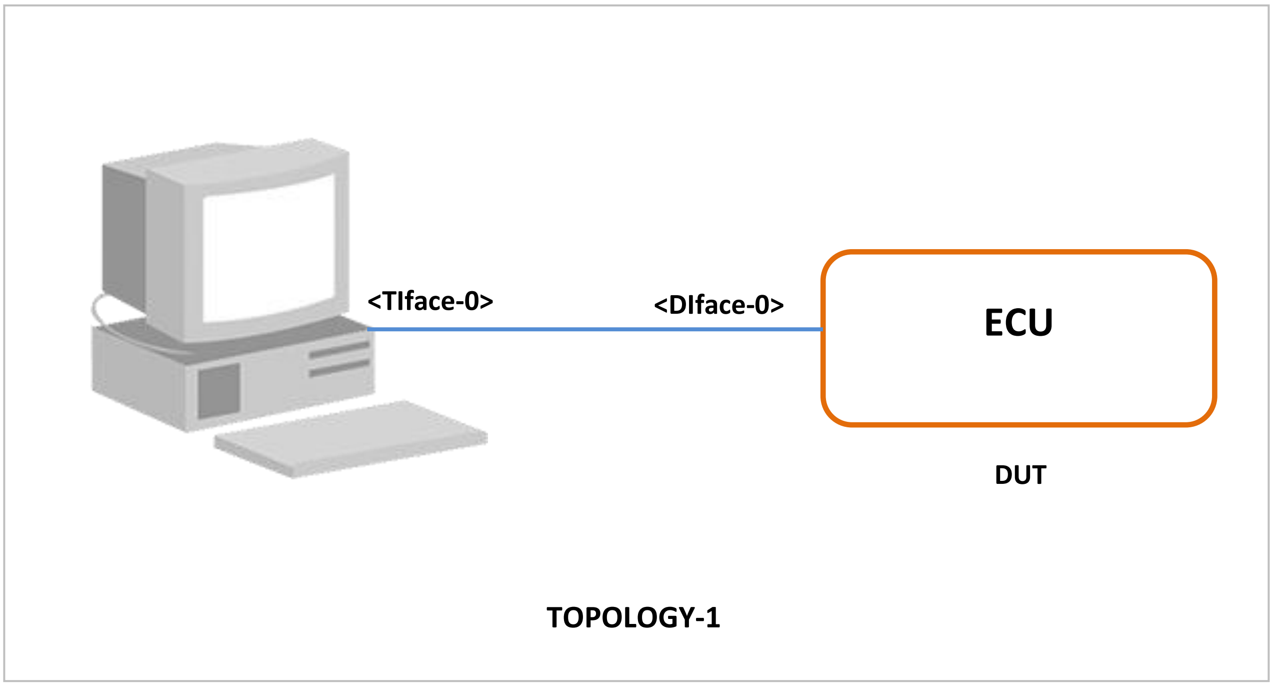
4.5 Dynamic configuration of IPv4 Link Local Address

4.5.1 General

The scope of this chapter is to specify test cases for the Dynamic IPv4 address Autoconfiguraron Protocol (IPv4 Autoconfig) based on the following standards:

- RFC 3927 - Dynamic Configuration of IPv4 Link-Local Addresses

4.5.2 Simulated topologies



4.5.3 Required topology related configuration

This suite expects to be running against any IP enabled network interface which supports acquisitiong of IP address through the following methods :

- IPv4 Link Local Autoconfiguration
- DHCP

The following information are obtained from the unused IP network configurations on the TESTER:

- IP Address of all the emulated servers
- IP Address Pool to be offered by these emulated DHCP Servers

4.5.4 Coverage

Specification Document	Sub-section	Test Category	Test Number(s)
RFC 3927: Dynamic Configuration of IP Link-Local Addresses	1.9	When to configure a link-local address	IPv4_AUTOCONF_INTRO_1 to IPv4_AUTOCONF_INTRO_6
RFC 3927: Dynamic Configuration of IP Link-Local Addresses	2.1 2.2.1 2.2	Address Selection, Defense and Delivery	IPv4_AUTOCONF_ADDRESS_SELECTION_1 to IPv4_AUTOCONF_ADDRESS_SELECTION_16
RFC 3927: Dynamic Configuration of IP Link-Local Addresses	2.4 1.2	Announcing an Address	IPv4_AUTOCONF_ANNOUNCING_1 to IPv4_AUTOCONF_ANNOUNCING_7
RFC 3927: Dynamic Configuration of IP Link-Local Addresses	2.5	Conflict Detection and Defense	IPv4_AUTOCONF_CONFLICT_1 to IPv4_AUTOCONF_CONFLICT_12
RFC 3927: Dynamic Configuration of IP Link-Local Addresses	2.6.2	Forwarding Rules	IPv4_AUTOCONF_FORWARDING_1 to IPv4_AUTOCONF_FORWARDING_8
RFC 3927: Dynamic Configuration of IP Link-Local Addresses	2.7	Link-Local Packets Are Not Forward	IPv4_AUTOCONF_LINKLOCAL_PACKETS_1 to IPv4_AUTOCONF_LINKLOCAL_PACKETS_4
RFC 3927: Dynamic Configuration of IP Link-Local Addresses	3.3	Interaction with Hosts with Routable Addresses	IPv4_AUTOCONF_ROUTABLE_ADDRESSES_1, IPv4_AUTOCONF_ROUTABLE_ADDRESSES_2
RFC 3927: Dynamic Configuration of IP Link-Local Addresses	4	Healing of Network Partitions	IPv4_AUTOCONF_NETWORKPARTITIONS_1
TOTAL			56

4.5.5 Parameters/constants used in the tests

Parameters used in test	Description
<SERVER-1>	This denotes 1st DHCP Server simulated by TESTER.
<ROUTABLE_IP_ADDR_1>	This denotes IP address of 1st DHCP Server simulated by TESTER
<ROUTABLE_IP_ADDR_2>	This denotes IP address offered to the 1st DUT interface by the 1st DHCP Server
<DIFACE-0-IP-LINKLOCAL-ADDR>	This denotes Link-local IP address of 1st DUT interface
<AIFACE-0-IP-LINKLOCAL-ADDR	This denotes link local IP Address of 1st TESTER interface
<ARBITRARY-IP-LINKLOCAL-ADDR>	This denotes an IP link-local address with value 169.254.10.10
<PARAM_PROCESS_TIME>	Amount of time TESTER will wait after sending a packet for which there is no immediate manifestation but later some other

	event will decide whether the DUT has correctly accepted the packet or not.
<PARAM_TOLERANCE_TIME>	Tolerance time associated with an event. When waiting or listening then this number will be added with the actual wait-time or listen-time.
<PARAM_LISTEN_TIME>	This is the maximum time interval for which TESTER waits for a packet for cases when a certain event has been triggered on the DUT either by some protocol timer or using some external mechanism (script).
<DIface-0>	This denotes 1st DUT interface.
<MAC-ADDR1>	This value is equal to the value provided for "IEEE First Unused MAC Address"
<MAC-ADDR2>	This is another value for MAC address which is auto-generated from "IEEE First Unused MAC Address"
<DIFACE_O_MAC_ADDR>	This is the MAC address of 1st DUT interface.
<LINK-LOCAL-NET-ADDR>	This denotes the Link local network ID i.e. 169.254.0.0
CONSTANT	Description
PROBE_MIN_IN_MILLISEC	This indicates the value of PROBE_MIN constant specified in RFC 3927 pg 26, in milliseconds i.e. the value is 1000 milliseconds.
PROBE_MAX_IN_MILLISEC	This indicates the value of PROBE_MAX constant specified in RFC 3927 pg 26, in milliseconds i.e. the value is 2000 milliseconds.
ANNOUNCE_WAIT	This indicates the value of ANNOUNCE_WAIT constant specified in RFC 3927 pg 26, i.e. the value is 2 seconds.
ANNOUNCE_INTERVAL	This indicates the value of ANNOUNCE_INTERVAL constant specified in RFC 3927 pg 26, i.e. the value is 2 seconds.
ANNOUNCE_INTERVAL_IN_MILLISEC	This indicates the value of ANNOUNCE_INTERVAL constant specified in RFC 3927 pg 26, in milliseconds i.e. the value is 2000 milliseconds.
PROBE_MIN_MILLISEC	This indicates the value of PROBE_MIN constant specified in RFC 3927 pg 26, in milliseconds i.e. the value is 1000.
PROBE_MIN_MILLISEC	This indicates the value of PROBE_MIN constant specified in RFC 3927 pg 26, in milliseconds i.e. the value is 1000.
RATE_LIMIT_INTERVAL	This indicates the value of RATE_LIMIT_INTERVAL constant specified in RFC 3927 pg 26, i.e. the value is 60 seconds.
DEFEND_INTERVAL	This indicates the value of DEFEND_INTERVAL constant specified in RFC 3927 pg 26, i.e. the value is 10 seconds.
ETHERNET_BROADCAST_ADDRESS	This indicates the Ethernet Broadcast Address. This value is equal to FF:FF:FF:FF:FF:FF
BROADCAST-IP	Broadcast IP address of the network
NULL-MAC-ADDRESS	This indicates a MAC address with value 00:00:00:00:00:00

4.5.6 Tests

4.5.6.1 Introduction

IPv4_AUTOCONF_INTRO_01: Link local address configurability condition (in presence of operable routable address)

Synopsis	When an operable routable address is available on an interface, the host SHOULD NOT also assign an IPv4 Link-Local address on that interface.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 6. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 10. DUT: Does not send ARP Request Message

	11. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 10. DUT: Does not send ARP Request Message
Test Iterations	
Notes	Derived from RFC 3927 p8 Section 1.9 (SHOULD)

IPv4_AUTOCONF_INTRO_02: Link local address preference condition (in presense of operable routable address)

Synopsis	If a host finds that an interface that was previously configured with an IPv4 Link-Local address now has an operable routable address available, the host MUST use the routable address when initiating new communications, and MUST cease advertising the availability of the IPv4 Link-Local address.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> <AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK

	- Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Destination Hardware Address set to <DIFACE-0-MAC-ADDR> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ARP Request Message 13. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 14. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 15. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 16. DUT: Sends DHCPDISCOVER Message 17. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 18. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 19. DUT: Sends DHCPREQUEST Message 20. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 21. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> 22. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <0.0.0.0> - Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 23. DUT: Sends ARP Probe or ARP Response 24. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1>

	- Destination IP Address set to <ROUTABLE_IP_ADDR_2> 25. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 26. DUT: Sends ARP Request Message 27. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ARP Request Message 16. DUT: Sends DHCPDISCOVER Message 19. DUT: Sends DHCPREQUEST Message 23. DUT: Sends ARP Probe or ARP Response 26. DUT: Sends ARP Request Message
Test Iterations	
Notes	Derived from RFC 3927 p9 Section 1.9 (MUST)

IPv4_AUTOCONF_INTRO_03: Link local address (usage continuation condition)

Synopsis	The host SHOULD continue to use the IPv4 Link-Local address for communications already underway, and MAY continue to accept new communications addressed to the IPv4 Link-Local address. (Note: In this test case TESTER checks that at that time when DUT is assigning a routable address on its interface it will continue to use its linklocal address for any undergoing communications, until the routable address assignment, through DHCP for example, is complete.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> <AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message

	5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Destination Hardware Address set to <DIFACE-0-MAC-ADDR> to verify that DUT has assigned the link-local ip 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ICMP Echo Reply Message 13. TESTER: <SERVER-1> Keeps on sending ICMP Echo Request Messages to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 14. TESTER: <SERVER-1> Listens (up to <20 seconds>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 15. DUT: Keeps on sending ICMP Echo Reply Messages 16. TESTER: Stores the count of ICMP echo replies received from <DIface-0> - in LINK_LOCAL_ECHO_REPLY_COUNT1 17. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 18. TESTER: Externally cause DUT to bring up <DIface-0> 19. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 20. DUT: Sends DHCPDISCOVER Message 21. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3>

	22. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 23. DUT: Sends DHCPREQUEST Message 24. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 25. TESTER: <SERVER-1> Keeps on sending ICMP Echo Request Messages to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 26. TESTER: <SERVER-1> Keeps on sending ICMP Echo Request Messages to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> 27. TESTER: <SERVER-1> Listens (up to <20 seconds>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> and - Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 28. DUT: Keeps on sending ICMP Echo Reply Messages 29. TESTER: Counts the number of ICMP echo replies received from <DIface-0> with -i) Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - increments LINK_LOCAL_ECHO_REPLY_COUNT1 and stores - the cumulative count in LINK_LOCAL_ECHO_REPLY_COUNT2 -ii) Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> - and stores the count in ROUTABLE_ECHO_REPLY_COUNT 30. TESTER: Verify that (LINK_LOCAL_ECHO_REPLY_COUNT2-LINK_LOCAL_ECHO_REPLY_COUNT1) is greater than 0 31. TESTER: Verify that ROUTABLE_ECHO_REPLY_COUNT is greater than 0 32. TESTER: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ICMP Echo Reply Message 15. DUT: Keeps on sending ICMP Echo Reply Messages 20. DUT: Sends DHCPDISCOVER Message 23. DUT: Sends DHCPREQUEST Message 28. DUT: Keeps on sending ICMP Echo Reply Messages 30. TESTER: Verify that (LINK_LOCAL_ECHO_REPLY_COUNT2-LINK_LOCAL_ECHO_REPLY_COUNT1) is greater than 0

	31. TESTER: Verify that ROUTABLE_ECHO_REPLY_COUNT is greater than 0
Test Iterations	
Notes	Derived from RFC 3927 p9 Section 1.9 (SHOULD)

IPv4_AUTOCONF_INTRO_04: Link local address (usage for new communication)

Synopsis	The host SHOULD continue to use the IPv4 Link-Local address for communications already underway, and MAY continue to accept new communications addressed to the IPv4 Link-Local address. (Note: In this test case TESTER checks that at that time when DUT is assigning a routable address on its interface it will continue to use its linklocal address for any undergoing communications, till the routable address assignment, and also it will accept new communications addresses to the link local address until the routable address assignment is complete.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> <AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR>

	- Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR+1> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 11. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Destination Hardware Address set to <DIFACE-0-MAC-ADDR> to verify that DUT has assigned the link-local ip 12. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 13. DUT: Sends ICMP Echo Reply Message 14. TESTER: <SERVER-1> Keeps on sending ICMP Echo Request Messages to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Destination Hardware Address set to <DIFACE-0-MAC-ADDR> 15. TESTER: <SERVER-1> Listens (up to <20 seconds>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 16. DUT: Keeps on sending ICMP Echo Reply Messages 17. TESTER: Stores the count of ICMP echo replies received from <DIface-0> - in LINK_LOCAL_ECHO_REPLY_COUNT1 18. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 19. TESTER: Externally cause DUT to bring up <DIface-0> 20. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 21. DUT: Sends DHCPDISCOVER Message 22. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 23. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 24. DUT: Sends DHCPREQUEST Message 25. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1>

	- 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 26. TESTER: <SERVER-1> Keeps on sending ICMP Echo Request Messages to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 27. TESTER: <SERVER-1> Keeps on sending ICMP Echo Request Messages to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR+1> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 28. TESTER: <SERVER-1> Keeps on sending ICMP Echo Request Messages to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> 29. TESTER: <SERVER-1> Listens (up to <20 seconds>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> and - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR+1> and - Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 30. DUT: Keeps on sending ICMP Echo Reply Messages 31. TESTER: Counts the number of ICMP echo replies received from <DIface-0> with -i) Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - increments LINK_LOCAL_ECHO_REPLY_COUNT1 and stores - the cumulative count in LINK_LOCAL_ECHO_REPLY_COUNT2 -ii) Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR+1> - and stores it in LINK_LOCAL_ECHO_REPLY_COUNT3 -iii) Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> - and stores it in ROUTABLE_ECHO_REPLY_COUNT 32. TESTER: Verify that (LINK_LOCAL_ECHO_REPLY_COUNT2-LINK_LOCAL_ECHO_REPLY_COUNT1) is greater than 0 33. TESTER: Verify that LINK_LOCAL_ECHO_REPLY_COUNT3 is greater than 0 34. TESTER: Verify that ROUTABLE_ECHO_REPLY_COUNT is greater than 0 35. TESTER: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 13. DUT: Sends ICMP Echo Reply Message

	16. DUT: Keeps on sending ICMP Echo Reply Messages 21. DUT: Sends DHCPDISCOVER Message 24. DUT: Sends DHCPREQUEST Message 30. DUT: Keeps on sending ICMP Echo Reply Messages 32. TESTER: Verify that (LINK_LOCAL_ECHO_REPLY_COUNT2-LINK_LOCAL_ECHO_REPLY_COUNT1) is greater than 0 33. TESTER: Verify that LINK_LOCAL_ECHO_REPLY_COUNT3 is greater than 0 34. TESTER: Verify that ROUTABLE_ECHO_REPLY_COUNT is greater than 0
Test Iterations	
Notes	Derived from RFC 3927 p9 Section 1.9 (MAY)

IPv4_AUTOCONF_INTRO_05: Link local address configurability condition-I (in absense of operable routable address)

Synopsis	If a host finds that an interface no longer has an operable routable address available, the host MAY identify a usable IPv4 Link-Local address and assign that address to the interface.Ways in which an operable routable address might cease to be available on an interface include: *Removal of the address from the interface through manual configuration
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to assign any static IP on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to remove the static IP on <DIface-0> 3. DUT CONFIGURE: Externally cause DUT to reboot <DIface-0> 4. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 5. DUT: Sends ARP Request Message 6. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 7. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 8. DUT CONFIGURE: Configure DUT to add a static route with

	- Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 9. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Destination Hardware Address set to <DIFACE-0-MAC-ADDR> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 11. DUT: Sends ARP Request Message 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	5. DUT: Sends ARP Request Message 11. DUT: Sends ARP Request Message
Test Iterations	
Notes	Derived from RFC 3927 p9 Section 1.9 (MAY)

IPv4_AUTOCONF_INTRO_06: Link local address configurability condition-II (in absense of operable routable address)

Synopsis	If a host finds that an interface no longer has an operable routable address available, the host MAY identify a usable IPv4 Link-Local address and assign that address to the interface.Ways in which an operable routable address might cease to be available on an interface include: *Expiration of the lease on the address assigned through DHCP
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> <AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0>

Procedure	3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME/3> 6. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME/3> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <0.0.0.0> - Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 11. DUT: Sends ARP Probe or ARP Response 12. TESTER: Wait till <(PARAM_LISTEN_TIME/3)-PARAM_TOLERANCE_TIME> so that the lease time of the routable IP address of <DIFace-0> gets expired 13. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 14. DUT: Sends ARP Request Message 15. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 16. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 17. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 18. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through

	<DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 19. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 20. DUT: Sends ARP Request Message 21. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ARP Probe or ARP Response 14. DUT: Sends ARP Request Message 20. DUT: Sends ARP Request Message
Test Iterations	
Notes	Derived from RFC 3927 p9 Section 1.9 (MAY)

4.5.6.2 Address Selection, Defense and Delivery

IPv4_AUTOCONF_ADDRESS_SELECTION_01: Future use of first 256 and last 256 addresses in the 169.254/16 prefix

Synopsis	The first 256 and last 256 addresses in the 169.254/16 prefix are reserved for future use and MUST NOT be selected by a host using this dynamic configuration mechanism.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Target IP Address is greater than <169.254.0.255> - Target IP Address is less than or equal to <169.254.254.255> 8. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Target IP Address is greater than <169.254.0.255> - Target IP Address is less than or equal to <169.254.254.255>
Test Iterations	
Notes	Derived from RFC 3927 p10 Section 2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_02: Use of persistent storage for each interface record

Synopsis	Hosts that are equipped with persistent storage MAY, for each interface, record the IPv4 address they have selected. On booting, hosts with a previously recorded address SHOULD use that address as their first candidate when probing.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Externally cause DUT to reboot <DIface-0> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 11. DUT: Sends ARP Request Message 12. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> 13. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 11. DUT: Sends ARP Request Message 12. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p10 Section 2.1 (SHOULD)

IPv4_AUTOCONF_ADDRESS_SELECTION_03: Need for probing to detect address already in use

Synopsis	A host probes to see if an address is already in use by broadcasting an ARP Request for the desired address. (Note: this is an ARP Probe Request)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Probe Request Message contains: - Ethernet Destination Hardware Address is set to ETHERNET_BROADCAST_ADDR 8. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Ethernet Destination Hardware Address is set to ETHERNET_BROADCAST_ADDR
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_04: Need for not probing periodically to detect address already in use

Synopsis	A host MUST NOT perform this check periodically as a matter of course.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Destination Hardware Address set to <DIFACE-0-MAC-ADDR> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ARP Request Message 13. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 14. DUT: Does not send ARP Request Message 15. CLEANUP: Externally cause DUT to shutdown <DIface-0>

Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ARP Request Message 14. DUT: Does not send ARP Request Message
Test Iterations	
Notes	Derived from RFC 3927 p11 Section 2.2 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_05: Sender hardware address field usage

Synopsis	The client MUST fill in the sender hardware address field of the ARP Request with the hardware address of the interface through which it is sending the packet. (Note: This holds true for all kinds of ARP frames: Request, Response, Probe)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Source Hardware Address is set to <DIFACE-0-MAC-ADDR> 8. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Source Hardware Address is set to <DIFACE-0-MAC-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_06: Sender IP address setting

Synopsis	The sender IP address field MUST be set to all zeroes. (Note: this is testing for the ARP Probe frame).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Sender IP Address is set to <0.0.0.0> 8. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Sender IP Address is set to <0.0.0.0>
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_07: Target hardware address setting and receive check

Synopsis	The target hardware address field is ignored and SHOULD be set to all zeroes.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Target Hardware Address is set to <NULL-MAC-ADDRESS> 8. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - Target Hardware Address is set to <NULL-MAC-ADDRESS>
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (SHOULD)

IPv4_AUTOCONF_ADDRESS_SELECTION_08: Target IP address field setting

Synopsis	The target IP address field MUST be set to the address being probed.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_09: Probing time interval and packet count - I

Synopsis	When ready to begin probing, the host should then wait for a random time interval selected uniformly in the range zero to PROBE_WAIT seconds, and should then send PROBE_NUM probe packets, each of these probe packets spaced randomly, PROBE_MIN to PROBE_MAX seconds apart. (Note : Here TESTER is verifying that the number of PROBES received is equal to PROBE_NUM)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <0.0.0.0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_10: Probing time interval and packet count - II

Synopsis	When ready to begin probing, the host should then wait for a random time interval selected uniformly in the range zero to PROBE_WAIT seconds, and should then send PROBE_NUM probe packets, each of these probe packets spaced randomly, PROBE_MIN to PROBE_MAX seconds apart. (Note : Here TESTER is verifying that the time gap between consecutive probes falls in the range <PROBE_MIN_IN_MILLISEC> to <PROBE_MAX_IN_MILLISEC>. Tolerance is 50 milliseconds).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: Verify that the time interval between reception of last two ARP Requests is greater than (<PROBE_MIN_IN_MILLISEC> - 50) milli second 8. TESTER: Verify that the time interval between reception of last two ARP Requests is less than (<PROBE_MAX_IN_MILLISEC> + 50) milli second 9. TESTER: Verify that the time interval between reception of 1st ARP Request and second last ARP Request is greater than (<PROBE_MIN_IN_MILLISEC> - 50) milli second 10. TESTER: Verify that the time interval between reception of 1st ARP Request and second last ARP Request is less than (<PROBE_MAX_IN_MILLISEC> + 50) milli second 11. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 7. TESTER: Verify that the time interval between

	reception of last two ARP Requests is greater than (<PROBE_MIN_IN_MILLISEC> - 50) milli second 8. TESTER: Verify that the time interval between reception of last two ARP Requests is less than (<PROBE_MAX_IN_MILLISEC> + 50) milli second 9. TESTER: Verify that the time interval between reception of 1st ARP Request and second last ARP Request is greater than (<PROBE_MIN_IN_MILLISEC> - 50) milli second 10. TESTER: Verify that the time interval between reception of 1st ARP Request and second last ARP Request is less than (<PROBE_MAX_IN_MILLISEC> + 50) milli second
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_11: Probing and reception of ARP packet - I

Synopsis	If during this period, from the beginning of the probing process until ANNOUNCE_WAIT seconds after the last probe packet is sent, the host receives any ARP packet (Request *or* Reply) on the interface where the probe is being performed where the packet's 'sender IP address' is the address being probed for, then the host MUST treat this address as being in use by some other host, and MUST select a new pseudo-random address and repeat the process. (Note : Here TESTER is sending a conflicting ARP request packet with sender IP address equal to the link-local probed address and verifies that DUT sends another ARP Probe with a different link-local probe address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <255.255.255.255> 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains:

	- Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_12: Probing and reception of ARP packet - II

Synopsis	If during this period, from the beginning of the probing process until ANNOUNCE_WAIT seconds after the last probe packet is sent, the host receives any ARP packet (Request *or* Reply) on the interface where the probe is being performed where the packet's 'sender IP address' is the address being probed for, then the host MUST treat this address as being in use by some other host, and MUST select a new pseudo-random address and repeat the process. (Note : Here TESTER is sending a conflicting ARP Response packet with sender IP address equal to the link-local probed address and verifies that DUT sends another ARP Probe with a different link-local probe address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <255.255.255.255> 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message

	11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_13: Probing and reception of ARP packet - III

Synopsis	In addition, if during this period the host receives any ARP Probe where the packet's 'target IP address' is the address being probed for, and the packet's 'sender hardware address' is not the hardware address of the interface the host is attempting to configure, then the host MUST similarly treat this as an address conflict and select a new address as above.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <0.0.0.0> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>

Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_14: Conflict resolution - I

Synopsis	If the number of conflicts exceeds MAX_CONFLICTS then the host MUST limit the rate at which it probes for new addresses to no more than one new address per RATE_LIMIT_INTERVAL. (Note : Here TESTER checks that if number of conflicts reach MAX_CONFLICTS(10) then the host should not send any ARP request within RATE_LIMIT_INTERVAL seconds after the 10th conflict)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to

	<LINK-LOCAL-NET-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 13. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 14. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 15. DUT: Sends ARP Request Message 16. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 17. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 18. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 19. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 20. DUT: Sends ARP Request Message 21. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 22. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 23. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 24. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 25. DUT: Sends ARP Request Message 26. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 27. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 28. TESTER: <SERVER-1> Sends ARP Request Message to DUT through

	<DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 29. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 30. DUT: Sends ARP Request Message 31. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 32. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 33. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 34. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 35. DUT: Sends ARP Request Message 36. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 37. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 38. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 39. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 40. DUT: Sends ARP Request Message 41. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 42. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 43. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 44. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 45. DUT: Sends ARP Request Message 46. TESTER: Verify that received ARP Request Message contains:

	- Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 47. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 48. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 49. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 50. DUT: Sends ARP Request Message 51. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 52. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 53. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 54. TESTER: <SERVER-1> Listens (up to <RATE_LIMIT_INTERVAL> second) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 55. DUT: Does not send ARP Request Message 56. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 15. DUT: Sends ARP Request Message 16. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 20. DUT: Sends ARP Request Message 21. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 25. DUT: Sends ARP Request Message 26. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 30. DUT: Sends ARP Request Message 31. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>

	35. DUT: Sends ARP Request Message 36. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 40. DUT: Sends ARP Request Message 41. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 45. DUT: Sends ARP Request Message 46. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 50. DUT: Sends ARP Request Message 51. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 55. DUT: Does not send ARP Request Message
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_15: Conflict resolution - II

Synopsis	If the number of conflicts exceeds MAX_CONFLICTS then the host MUST limit the rate at which it probes for new addresses to no more than one new address per RATE_LIMIT_INTERVAL. (Note : Here TESTER checks that if number of conflicts reach MAX_CONFLICTS(10) then the host should send only 1 ARP request after the RATE_LIMIT_INTERVAL seconds get over, within the next RATE_LIMIT_INTERVAL seconds)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 13. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 14. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0>

	- Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 15. DUT: Sends ARP Request Message 16. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 17. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 18. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 19. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 20. DUT: Sends ARP Request Message 21. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 22. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 23. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 24. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 25. DUT: Sends ARP Request Message 26. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 27. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 28. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 29. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 30. DUT: Sends ARP Request Message 31. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 32. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR>

	33. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 34. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 35. DUT: Sends ARP Request Message 36. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 37. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 38. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 39. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 40. DUT: Sends ARP Request Message 41. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 42. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 43. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 44. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 45. DUT: Sends ARP Request Message 46. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 47. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 48. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 49. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 50. DUT: Sends ARP Request Message

	51. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 52. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 53. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 54. TESTER: <SERVER-1> Listens (up to <RATE_LIMIT_INTERVAL> second) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 55. DUT: Does not send ARP Request Message 56. TESTER: <SERVER-1> Listens (up to <RATE_LIMIT_INTERVAL> second) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 57. DUT: Sends ARP Request Message 58. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 59. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 60. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 61. TESTER: <SERVER-1> Listens (up to <RATE_LIMIT_INTERVAL> second) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 62. DUT: Does not send ARP Request Message 63. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 15. DUT: Sends ARP Request Message 16. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 20. DUT: Sends ARP Request Message 21. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>

	25. DUT: Sends ARP Request Message 26. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 30. DUT: Sends ARP Request Message 31. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 35. DUT: Sends ARP Request Message 36. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 40. DUT: Sends ARP Request Message 41. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 45. DUT: Sends ARP Request Message 46. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 50. DUT: Sends ARP Request Message 51. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 55. DUT: Does not send ARP Request Message 57. DUT: Sends ARP Request Message 58. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 62. DUT: Does not send ARP Request Message
Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

IPv4_AUTOCONF_ADDRESS_SELECTION_16: IPv4 Link-Local address claim condition - I

Synopsis	If, by ANNOUNCE_WAIT seconds after the transmission of the last ARP Probe no conflicting ARP Reply or ARP Probe has been received, then the host has successfully claimed the desired IPv4 Link-Local address.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> <MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Sender Hardware Address set to <MAC-ADDR1> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 11. DUT: Sends ARP Response Message 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 11. DUT: Sends ARP Response Message

Test Iterations	
Notes	Derived from RFC 3927 p12 Section 2.2.1 (MUST)

4.5.6.3 Announcing an Address

IPv4_AUTOCONF_ANNOUNCING_01: An ARP announcement - I

Synopsis	An ARP announcement is identical to the ARP Probe described above, except that now the sender and target IP addresses are both set to the host's newly selected IPv4 address. In this document, the term 'ARP Announcement' is used to refer to an ARP Request packet, broadcast on the local link, identical to the ARP Probe described above, except that both the sender and target IP address fields contain the IP address being announced. (Note: Here TESTER checks that after receiving 3 ARP Probes from DUT interface, TESTER receives an ARP Announcement(Request) packet with Destination MAC address set to Ethernet Broadcast Address.A tolerance time of -0.05 seconds is used at the step where TESTER is waiting up to ANNOUNCE_WAIT seconds)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT-0.05> i.e. the delay time before DUT sends the ARP Announcement message 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Ethernet Destination Hardware Address is set to ETHERNET_BROADCAST_ADDR 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>

Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Ethernet Destination Hardware Address is set to ETHERNET_BROADCAST_ADDR
Test Iterations	
Notes	Derived from RFC 3927 p13 section 2.4 (MUST)

IPv4_AUTOCONF_ANNOUNCING_02: An ARP announcement - II

Synopsis	An ARP announcement is identical to the ARP Probe described above, except that now the sender and target IP addresses are both set to the host's newly selected IPv4 address. In this document, the term 'ARP Announcement' is used to refer to an ARP Request packet, broadcast on the local link, identical to the ARP Probe described above, except that both the sender and target IP address fields contain the IP address being announced. (Note: Here TESTER checks that after receiving 3 ARP Probes from DUT interface, TESTER receives an ARP Announcement(Request) packet with Source IP address = Target IP address = Announced IP address. A tolerance time of -0.05 seconds is used at the step where TESTER is waiting up to ANNOUNCE_WAIT seconds)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT-0.05> for DUT to send ARP

	Announcement Message 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13 section 2.4 (MUST)

IPv4_AUTOCONF_ANNOUNCING_03: An ARP announcement - III

Synopsis	An ARP announcement is identical to the ARP Probe described above, except that now the sender and target IP addresses are both set to the host's newly selected IPv4 address. In this document, the term 'ARP Announcement' is used to refer to an ARP Request packet, broadcast on the local link, identical to the ARP Probe described above, except that both the sender and target IP address fields contain the IP address being announced. (Note: Here TESTER checks that after receiving 3 ARP Probes from DUT interface, TESTER receives an ARP Announcement(Request) packet with Sender Hardware address set to DUT interface MAC Address. A tolerance time of -0.05 seconds is used at the step where TESTER is waiting up to ANNOUNCE_WAIT seconds)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0>

Procedure	3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT-0.05> for DUT to send ARP Announcement Message 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Source Hardware Address is set to <DIFACE-0-MAC-ADDR> 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Source Hardware Address is set to <DIFACE-0-MAC-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13 section 2.4 (MUST)

IPv4_AUTOCONF_ANNOUNCING_04: An ARP announcement - IV

Synopsis	An ARP announcement is identical to the ARP Probe described above, except that now the sender and target IP addresses are both set to the host's newly selected IPv4 address. In this document, the term 'ARP Announcement' is used to refer to an ARP Request packet, broadcast on the local link, identical to the ARP Probe described above, except that both the sender and target IP address fields contain the IP address being announced. (Note: Here TESTER checks that after receiving 3 ARP Probes from DUT interface, TESTER receives an ARP Announcement(Request) packet with Target Hardware address set to all zeroes. A tolerance time of -0.05 seconds is used at the step where TESTER is waiting up to ANNOUNCE_WAIT seconds)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT-0.05> for DUT to send ARP Announcement Message 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT: Sends ARP Request Message 11. TESTER: Verify that received ARP Request Message contains: - Target Hardware Address is set to <NULL-MAC-ADDRESS> 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends ARP Request Message

	11. TESTER: Verify that received ARP Request Message contains: - Target Hardware Address is set to <NULL-MAC-ADDRESS>
Test Iterations	
Notes	Derived from RFC 3927 p13 section 2.4 (SHOULD)

IPv4_AUTOCONF_ANNOUNCING_05: Announcing claimed address

Synopsis	Having probed to determine a unique address to use, the host MUST then announce its claimed address by broadcasting ANNOUNCE_NUM ARP announcements, spaced ANNOUNCE_INTERVAL seconds apart. (Note : A tolerance time of -0.05 seconds is used at the step where TESTER is waiting up to ANNOUNCE_WAIT seconds)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT-0.05> for DUT to send ARP Announcement Message 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT: Sends 2 ARP Request Messages 11. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 10. DUT: Sends 2 ARP Request Messages
Test Iterations	
Notes	Derived from RFC 3927 p13 section 2.4 (MUST)

IPv4_AUTOCONF_ANNOUNCING_06: Announcing claimed address (interval and packet count)

Synopsis	Having probed to determine a unique address to use, the host MUST then announce its claimed address by broadcasting ANNOUNCE_NUM ARP announcements, spaced ANNOUNCE_INTERVAL seconds apart. (Note : Here TESTER is verifying that the time gap between the two announce messages is <ANNOUNCE_INTERVAL_IN_MILLISEC>. Tolerance is 50 milliseconds. A tolerance time of -0.05 seconds is used at the step where TESTER is waiting up to ANNOUNCE_WAIT seconds).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT-0.05> for DUT to send ARP Announcement Message 9. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT: Sends 2 ARP Request Messages 11. TESTER: Verify that the time interval between reception of last two ARP Requests is greater than (<ANNOUNCE_INTERVAL_IN_MILLISEC> - 50) milli second 12. TESTER: Verify that the time interval between reception of last two ARP Requests is less than (<ANNOUNCE_INTERVAL_IN_MILLISEC> + 50) milli second 13. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages

	10. DUT: Sends 2 ARP Request Messages 11. TESTER: Verify that the time interval between reception of last two ARP Requests is greater than (<ANNOUNCE_INTERVAL_IN_MILLISEC> - 50) milli second 12. TESTER: Verify that the time interval between reception of last two ARP Requests is less than (<ANNOUNCE_INTERVAL_IN_MILLISEC> + 50) milli second
Test Iterations	
Notes	Derived from RFC 3927 p13 section 2.4 (MUST)

IPv4_AUTOCONF_ANNOUNCING_07: Announcing claimed address (non-conflict)

Synopsis	The purpose of these ARP announcements is to make sure that other hosts on the link do not have stale ARP cache entries left over from some other host that may previously have been using the same address.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> <PARAM_PROCESS_TIME> <MAC-ADDR1> <MAC-ADDR2> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL>

	for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Source Hardware Address set to <MAC-ADDR1> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ARP Request Message 13. TESTER: <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Sender Hardware Address set to <MAC-ADDR1> 14. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Source Hardware Address set to <MAC-ADDR1> 15. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 16. DUT: Sends ICMP Echo Reply Message 17. TESTER: Verify that received ICMP Echo Reply Message contains: - Ethernet Destination Hardware Address is set to <MAC-ADDR1> 18. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Sender Hardware Address set to <MAC-ADDR2> - Ethernet Source Hardware Address set to <MAC-ADDR2> 19. TESTER: Wait till <PARAM_PROCESS_TIME> for DUT to update its arp cache entry 20. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 21. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 22. DUT: Sends ICMP Echo Reply Message

	23. TESTER: Verify that received ICMP Echo Reply Message contains: - Ethernet Destination Hardware Address is set to <MAC-ADDR2> 24. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ARP Request Message 16. DUT: Sends ICMP Echo Reply Message 17. TESTER: Verify that received ICMP Echo Reply Message contains: - Ethernet Destination Hardware Address is set to <MAC-ADDR1> 22. DUT: Sends ICMP Echo Reply Message 23. TESTER: Verify that received ICMP Echo Reply Message contains: - Ethernet Destination Hardware Address is set to <MAC-ADDR2>
Test Iterations	
Notes	Derived from RFC 3927 p13 section 2.4 (MUST)

4.5.6.4 Conflict Detection and Defense

IPv4_AUTOCONF_CONFLICT_01: Conflicting ARP packet reception

Synopsis	Upon receiving a conflicting ARP packet, a host MAY elect to immediately configure a new IPv4 Link-Local address. (Note: In this test case, TESTER is sending a conflicting ARP Request message with source IP Address and target IP Address both set to probed link-local address sent by the DUT and checks whether DUT sends a probe with a different link local address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 11. DUT: Sends ARP Request Message 12. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 13. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 11. DUT: Sends ARP Request Message

	12. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_02: Defending own address - I

Synopsis	If a host currently has active TCP connections or other reasons to prefer to keep the same IPv4 address, and it has not seen any other conflicting ARP packets within the last DEFEND_INTERVAL seconds, then it MAY elect to attempt to defend its address by recording the time that the conflicting ARP packet was received, and then broadcasting one single ARP announcement, giving its own IP and hardware addresses as the sender addresses of the ARP (Note : Here TESTER sends a conflicting ARP request message with both source and target IP address set to the probed link local address and checks that the ARP announcement that DUT has send to defend its address has both the Source IP Address and Target IP address set to DUT's LinkLocal IP Address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR>

	- Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: Wait till <DEFEND_INTERVAL-PARAM_TOLERANCE_TIME> for DUT to process the conflicting packet and defend its link-local address at <DIface-0> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_03: Defending own address - II

Synopsis	If a host currently has active TCP connections or other reasons to prefer to keep the same IPv4 address, and it has not seen any other conflicting ARP packets within the last DEFEND_INTERVAL seconds, then it MAY elect to attempt to defend its address by recording the time that the conflicting ARP packet was received, and then broadcasting one single ARP announcement, giving its own IP and hardware addresses as the sender addresses of the ARP (Note : Here TESTER sends a conflicting ARP Request message with both source and target IP Address set to the advertised link local address and checks that the ARP announcement that DUT has send to defend its address has the Sender Hardware Address set to DUT interface MAC Address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0>

Procedure	2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: Wait till <DEFEND_INTERVAL-PARAM_TOLERANCE_TIME> for DUT to process the conflicting packet and defend its link-local address at <DIface-0> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Source Hardware Address is set to <DIFACE-0-MAC-ADDR> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Source Hardware Address is set to <DIFACE-0-MAC-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_04: Defending own address - III

Synopsis	If a host currently has active TCP connections or other reasons to prefer to keep the same IPv4 address, and it has not seen any other conflicting ARP packets within the last DEFEND_INTERVAL seconds, then it MAY elect to attempt to defend its address by recording the time that the conflicting ARP packet was received, and then broadcasting one single ARP announcement, giving its own IP and hardware addresses as the sender addresses of the ARP (Note : Here TESTER sends a conflicting ARP Response message with both source and target IP address set to the advertised link local address and checks that the ARP announcement that DUT has send to defend its address has both the Source IP Address and Target IP address set to DUT's LinkLocal IP Address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: Wait till <DEFEND_INTERVAL-PARAM_TOLERANCE_TIME> for DUT to process the conflicting packet and defend it link-local address at <DIface-0> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>

	- Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_05: Defending own address - IV

Synopsis	If a host currently has active TCP connections or other reasons to prefer to keep the same IPv4 address, and it has not seen any other conflicting ARP packets within the last DEFEND_INTERVAL seconds, then it MAY elect to attempt to defend its address by recording the time that the conflicting ARP packet was received, and then broadcasting one single ARP announcement, giving its own IP and hardware addresses as the sender addresses of the ARP (Note : Here TESTER sends a conflicting ARP Response packet with both source and target IP Address set to the advertised link local address and checks that the ARP announcement that DUT has send to defend its address has the Sender Hardware Address set to DUT interface MAC Address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol

	Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: Wait till <DEFEND_INTERVAL-PARAM_TOLERANCE_TIME> for DUT to process the conflicting packet and defend it link-local address at <DIface-0> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Source Hardware Address is set to <DIFACE-0-MAC-ADDR> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Source Hardware Address is set to <DIFACE-0-MAC-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_06: Link local address (usage cease condition - I)

Synopsis	If this is not the first conflicting ARP packet the host has seen, and the time recorded for the previous conflicting ARP packet is recent, within DEFEND_INTERVAL seconds, then the host MUST immediately cease using this address and configure a new IPv4 Link-Local address. (Note:In this test case, TESTER sends 2 conflicting ARP Request mesages with both source and target IP address set to the advertised link local address, within DEFEND_INTERVAL seconds, and check if DUT probes with a new advertised link local address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: Wait till <DEFEND_INTERVAL/2> before sending the next conflicting packet to <DIface-0> 11. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 13. DUT: Sends ARP Request Message 14. TESTER: Verify that received ARP Request Message contains:

	- Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 15. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 13. DUT: Sends ARP Request Message 14. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13-14 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_07: Link local address (usage cease condition - II)

Synopsis	If this is not the first conflicting ARP packet the host has seen, and the time recorded for the previous conflicting ARP packet is recent, within DEFEND_INTERVAL seconds, then the host MUST immediately cease using this address and configure a new IPv4 Link-Local address. (Note:In this test case, TESTER sends 2 conflicting ARP Response messages with both source and target IP address set to the advertised link local address, within DEFEND_INTERVAL seconds and check if DUT probes with a new advertised link local address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Response Message to DUT through

	<DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: Wait till <DEFEND_INTERVAL/2> before sending the next conflicting packet to <DIface-0> 11. TESTER: <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 13. DUT: Sends ARP Request Message 14. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 15. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 13. DUT: Sends ARP Request Message 14. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13-14 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_08: Link local address (usage cease condition - III)

Synopsis	If this is not the first conflicting ARP packet the host has seen, and the time recorded for the previous conflicting ARP packet is recent, within DEFEND_INTERVAL seconds, then the host MUST immediately cease using this address and configure a new IPv4 Link-Local address. (Note:In this test case, TESTER first sends 1 conflicting ARP Request mesages and then sends 1 conflicting ARP Response message, within DEFEND_INTERVAL seconds, and check if DUT probes with a new advertised link local address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: Wait till <DEFEND_INTERVAL/2> before sending the next conflicting packet to <DIface-0> 11. TESTER: <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 13. DUT: Sends ARP Request Message 14. TESTER: Verify that received ARP Request Message contains:

	- Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 15. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 13. DUT: Sends ARP Request Message 14. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13-14 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_09: Link local address (usage cease condition - IV)

Synopsis	If this is not the first conflicting ARP packet the host has seen, and the time recorded for the previous conflicting ARP packet is recent, within DEFEND_INTERVAL seconds, then the host MUST immediately cease using this address and configure a new IPv4 Link-Local address. (Note:In this test case, TESTER first sends 1 conflicting ARP Response message and then sends 1 conflicting ARP request message, within DEFEND_INTERVAL seconds and check if DUT probes with a new advertised link local address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Response Message to DUT through

	<DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: Wait till <DEFEND_INTERVAL/2> before sending the next conflicting packet to <DIface-0> 11. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 12. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 13. DUT: Sends ARP Request Message 14. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 15. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 13. DUT: Sends ARP Request Message 14. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13-14 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_10: Receiving a conflicting ARP packet

Synopsis	Upon receiving a conflicting ARP packet, a host MAY elect to immediately configure a new IPv4 Link-Local address. (Note: In this test case, TESTER is sending a conflicting ARP Response message with source IP Address and target IP Address both set to probed link-local address sent by the DUT and checks whether DUT sends a probe with a different link local address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing: - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 11. DUT: Sends ARP Request Message 12. TESTER: Verify that received ARP Request Message contains: - Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR> 13. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 11. DUT: Sends ARP Request Message 12. TESTER: Verify that received ARP Request Message contains:

	- Target IP Address is not set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p13 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_11: ARP packets containing (Link- Local 'sender IP address') rule - I

Synopsis	All ARP packets (*replies* as well as requests) that contain a Link- Local 'sender IP address' MUST be sent using link-layer broadcast instead of link-layer unicast. (Note:Here TESTER checks that DUT sends ARP Probe Response with Ethernet destination address set to broadcast address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Probe Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 11. DUT: Sends ARP Response Message 12. TESTER: Verify that received ARP Response Message contains: - Target IP Address is set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>

	- Ethernet Destination Hardware Address is set to ETHERNET_BROADCAST_ADDR 13. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Probe Request Messages 11. DUT: Sends ARP Response Message 12. TESTER: Verify that received ARP Response Message contains: - Target IP Address is set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Destination Hardware Address is set to ETHERNET_BROADCAST_ADDR
Test Iterations	
Notes	Derived from RFC 3927 p14 Section 2.5 (MUST)

IPv4_AUTOCONF_CONFLICT_12: ARP packets containing (Link- Local 'sender IP address') rule - II

Synopsis	All ARP packets (*replies* as well as requests) that contain a Link- Local 'sender IP address' MUST be sent using link-layer broadcast instead of link-layer unicast. (Note:Here TESTER checks that DUT sends ARP Requests with Ethernet destination address set to broadcast address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Ethernet Destination Hardware Address is set to ETHERNET_BROADCAST_ADDR

	14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Ethernet Destination Hardware Address is set to ETHERNET_BROADCAST_ADDR
Test Iterations	
Notes	Derived from RFC 3927 p14 Section 2.5 (MUST)

4.5.6.5 Forwarding Rules

IPv4_AUTOCONF_FORWARDING_01: Forwarding Rules (Destination address is in the 169.254/16 prefix - I)

Synopsis	If the destination address is in the 169.254/16 prefix (excluding the address 169.254.255.255, which is the broadcast address for the Link-Local prefix), then the sender MUST ARP for the destination address and then send its packet directly to the destination on the same physical link. (Note : Here TESTER checks that the DUT interface, which has a linklocal ip address, sends an ARP request when it is configured to send an ICMP echo request to a link-local destination IP address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains:

	- Target IP Address is set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ARP Request Message 13. TESTER: Verify that received ARP Request Message contains: - Target IP Address is set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p15 Section 2.6.2 (MUST)

IPv4_AUTOCONF_FORWARDING_02: Forwarding Rules (Destination address is in the 169.254/16 prefix - II)

Synopsis	If the destination address is in the 169.254/16 prefix (excluding the address 169.254.255.255, which is the broadcast address for the Link-Local prefix), then the sender MUST ARP for the destination address and then send its packet directly to the destination on the same physical link. (Note : Here TESTER checks that DUT sends an ICMP echo request with destination IP address set to TESTER interface's link local ip address and source IP address set to its own linklocal IP Address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol

	Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Source IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address is set to <AIFACE-0-IP-LINKLOCAL-ADDR> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Source IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address is set to <AIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p15 Section 2.6.2 (MUST)

IPv4_AUTOCONF_FORWARDING_03: Forwarding Rules (Destination address is in the 169.254/16 prefix - III)

Synopsis	If the destination address is in the 169.254/16 prefix (excluding the address 169.254.255.255, which is the broadcast address for the Link-Local prefix), then the sender MUST ARP for the destination address and then send its packet directly to the destination on the same physical link. (Note : Here TESTER checks that DUT sends an ICMP echo request with ethernet destination address set to TESTER interface's MAC Address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> <MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains:

	- Ethernet Destination Hardware Address is set to <MAC-ADDR1> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Ethernet Destination Hardware Address is set to <MAC-ADDR1>
Test Iterations	
Notes	Derived from RFC 3927 p15 Section 2.6.2 (MUST)

IPv4_AUTOCONF_FORWARDING_04: Forwarding Rules (Host sending the packet with an IPv4 Link-Local source address - I)

Synopsis	If for any reason the host chooses to send the packet with an IPv4 Link-Local source address (e.g., no routable address is available on the selected interface), then it MUST ARP for the destination address and then send its packet, with an IPv4 Link-Local source address and a routable destination IPv4 address, directly to its destination on the same physical link. (Note : Here TESTER checks that DUT sends an ICMP echo request with source IP address as its own linklocal IP Address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL>

	for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <ROUTABLE_IP_ADDR_1> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Source IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Source IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p15-16 Section 2.6.2 (MUST)

IPv4_AUTOCONF_FORWARDING_05: Forwarding Rules (Host sending the packet with an IPv4 Link-Local source address - II)

Synopsis	If for any reason the host chooses to send the packet with an IPv4 Link-Local source address (e.g., no routable address is available on the selected interface), then it MUST ARP for the destination address and then send its packet, with an IPv4 Link-Local source address and a routable destination IPv4 address, directly to its destination on the same physical link. (Note : Here TESTER checks that DUT sends an ICMP echo request with destination IP address as the routable address its trying to ping)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <ROUTABLE_IP_ADDR_1> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Destination IP Address is set to <ROUTABLE_IP_ADDR_1> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>

Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Destination IP Address is set to <ROUTABLE_IP_ADDR_1>
Test Iterations	
Notes	Derived from RFC 3927 p15-16 Section 2.6.2 (MUST)

IPv4_AUTOCONF_FORWARDING_06: Forwarding Rules (Host sending the packet with an IPv4 Link-Local source address - III)

Synopsis	If for any reason the host chooses to send the packet with an IPv4 Link-Local source address (e.g., no routable address is available on the selected interface), then it MUST ARP for the destination address and then send its packet, with an IPv4 Link-Local source address and a routable destination IPv4 address, directly to its destination on the same physical link. (Note : Here TESTER checks that DUT sends an ICMP echo request with ethernet destination address set to TESTER interface mac address))
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0>

	9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <ROUTABLE_IP_ADDR_1> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Ethernet Destination Hardware Address is set to <MAC-ADDR1> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ICMP Echo Request Message 13. TESTER: Verify that received ICMP Echo Request Message contains: - Ethernet Destination Hardware Address is set to <MAC-ADDR1>
Test Iterations	
Notes	Derived from RFC 3927 p15-16 Section 2.6.2 (MUST)

IPv4_AUTOCONF_FORWARDING_07: Forwarding Rules (Destination address is a unicast address outside the 169.254/16 prefix)

Synopsis	If the destination address is a unicast address outside the 169.254/16 prefix, then the host SHOULD use an appropriate routable IPv4 source address, if it can. (Note : Here TESTER checks that DUT sends an ICMP echo request with source IP address set to routable address)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 6. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <0.0.0.0>

	- Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 11. DUT: Sends ARP Probe or ARP Response 12. DUT CONFIGURE: Configure DUT to send an ICMP Echo Request Message from <DIface-0> with - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 13. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 14. DUT: Sends ICMP Echo Request Message 15. TESTER: Verify that received ICMP Echo Request Message contains: - Source IP Address is set to <ROUTABLE_IP_ADDR_2> - Destination IP Address is set to <ROUTABLE_IP_ADDR_1> 16. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ARP Probe or ARP Response 14. DUT: Sends ICMP Echo Request Message 15. TESTER: Verify that received ICMP Echo Request Message contains: - Source IP Address is set to <ROUTABLE_IP_ADDR_2> - Destination IP Address is set to <ROUTABLE_IP_ADDR_1>
Test Iterations	
Notes	Derived from RFC 3927 p15 Section 2.6.2 (SHOULD)

IPv4_AUTOCONF_FORWARDING_08: Forwarding Rules (host MUST NOT send a packet with an IPv4 Link-Local destination address to any router for forwarding)

Synopsis	The host MUST NOT send a packet with an IPv4 Link-Local destination address to any router for forwarding. (Note : Here TESTER sends a ping with destination IP address set to an arbitrary link-local address and checks that DUT should not send the ping to a router after consulting the static route entry in its routing table)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> <ARBITRARY-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 6. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ROUTABLE_IP_ADDR_2>

	10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <0.0.0.0> - Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 11. DUT: Sends ARP Probe or ARP Response 12. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <ARBITRARY-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <ROUTABLE_IP_ADDR_2+1> 13. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ARBITRARY-IP-LINKLOCAL-ADDR> 14. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <ARBITRARY-IP-LINKLOCAL-ADDR> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 15. DUT: Does not send ICMP Echo Request Message 16. CLEANUP: Configure DUT to delete a static route on <DIface-0> with - Destination Prefix set to <ARBITRARY-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <ROUTABLE_IP_ADDR_2+1> 17. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ARP Probe or ARP Response 15. DUT: Does not send ICMP Echo Request Message
Test Iterations	
Notes	Derived from RFC 3927 p15 Section 2.6.2 (MUST)

4.5.6.6 Link-Local Packets Are Not Forwarded

IPv4_AUTOCONF_LINKLOCAL_PACKETS_01: Link-Local Packets Are Not Forwarded (TTL)

Synopsis	A sensible default for applications which are sending from an IPv4 Link-Local address is to explicitly set the IPv4 TTL to 1.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> <MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Ethernet Source Hardware Address set to <MAC-ADDR1> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 12. DUT: Sends ICMP Echo Reply Message 13. TESTER: Verify that received ICMP Echo Reply Message contains:

	- TTL field is set to 1 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Sends ICMP Echo Reply Message 13. TESTER: Verify that received ICMP Echo Reply Message contains: - TTL field is set to 1
Test Iterations	
Notes	Derived from RFC 3927 p16 section 2.7 (MAY)

IPv4_AUTOCONF_LINKLOCAL_PACKETS_02: Link-Local Packets Are Not Forwarded - (Source and/or destination address is in the 169.254/16 prefix)

Synopsis	An IPv4 packet whose source and/or destination address is in the 169.254/16 prefix MUST NOT be sent to any router for forwarding. (Note : Here TESTER sends a ping with destination IP address set to DUT interface link-local address and checks that DUT should not send a ping reply after consulting the static route entry in its routing table)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ARBITRARY-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. DUT CONFIGURE: Configure DUT to add a static route with

	- Destination Prefix set to <ROUTABLE_IP_ADDR_1> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <ARBITRARY-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 11. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 12. DUT: Does not send ICMP Echo Reply Message 13. CLEANUP: Configure DUT to delete a static route on <DIface-0> with - Destination Prefix set to <ROUTABLE_IP_ADDR_1> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <ARBITRARY-IP-LINKLOCAL-ADDR> 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 12. DUT: Does not send ICMP Echo Reply Message
Test Iterations	
Notes	Derived from RFC 3927 p16 Section 2.7 (MUST)

IPv4_AUTOCONF_LINKLOCAL_PACKETS_03: Link-Local Packets Are Not Forwarded (Router response)

Synopsis	A router may of course answer ARP Requests for one or more IPv4 Link-Local address(es) that it has legitimately claimed for its own use according to the claim-and-defend protocol described in this document. (Note : Here TESTER checks that the router/host must answer to the ARP request for the Link-local address it is using)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> <MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Target Hardware Address set to <NULL-MAC-ADDRESS> - Sender Hardware Address set to <MAC-ADDR1> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 11. DUT: Sends ARP Response Message 12. TESTER: Verify that received ARP Response Message contains: - Target IP Address is set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR> 13. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message

	11. DUT: Sends ARP Response Message 12. TESTER: Verify that received ARP Response Message contains: - Target IP Address is set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Sender IP Address is set to <DIFACE-0-IP-LINKLOCAL-ADDR>
Test Iterations	
Notes	Derived from RFC 3927 p16 Section 2.7 (MAY)

IPv4_AUTOCONF_LINKLOCAL_PACKETS_04: Link-Local Packets Are Not Forwarded (router or other host response for addresses in the 169.254/16 prefix)

Synopsis	A router or other host MUST NOT indiscriminately answer all ARP Requests for addresses in the 169.254/16 prefix. A router may of course answer ARP Requests for one or more IPv4 Link-Local address(es) that it has legitimately claimed for its own use according to the claim-and-defend protocol described in this document. (Note : Here TESTER checks that the router/host will not answer to the ARP request for an arbitrary link-local address which it is not using)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> <ARBITRARY-IP-LINKLOCAL-ADDR> <MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0>

	9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <ARBITRARY-IP-LINKLOCAL-ADDR> - Target Hardware Address set to <NULL-MAC-ADDRESS> - Sender Hardware Address set to <MAC-ADDR1> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <ARBITRARY-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 11. DUT: Does not send ARP Response Message 12. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Request Message 11. DUT: Does not send ARP Response Message
Test Iterations	
Notes	Derived from RFC 3927 p16 Section 2.7 (MUST)

4.5.6.7 Interaction with Hosts with Routable Addresses

IPv4_AUTOCONF_ROUTABLE_ADDRESSES_01: Host acquires a routable address

Synopsis	When a host acquires a routable address, it does not need to retain its Link-Local address for the purpose of communicating with other devices on the link that are themselves using only Link-Local addresses:
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> <AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends ARP Probe Request Message 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores Target IP address in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <(PROBE_MAX*2)+ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Target Hardware Address set to <NULL-MAC-ADDRESS> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 11. DUT: Sends ARP Response Message 12. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 13. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 14. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 15. DUT: Sends DHCPDISCOVER Message 16. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0>

	containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 17. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 18. DUT: Sends DHCPREQUEST Message 19. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 20. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> 21. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> for ARP response message 22. DUT: Sends ARP Response - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 23. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Target Hardware Address set to <NULL-MAC-ADDRESS> - Ethernet Destination Hardware Address set to ETHERNET_BROADCAST_ADDR 24. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 25. DUT: Does not send ARP Response Message 26. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends ARP Probe Request Message 11. DUT: Sends ARP Response Message 15. DUT: Sends DHCPDISCOVER Message 18. DUT: Sends DHCPREQUEST Message 22. DUT: Sends ARP Response 25. DUT: Does not send ARP Response Message

Test Iterations	
Notes	Derived from RFC 3927 p20 Section 3.3 (MAY)

IPv4_AUTOCONF_ROUTABLE_ADDRESSES_02: Interaction with Hosts with Routable Addresses

Synopsis	It is not necessary for that host to have a Link-Local source address in order to send to a Link-Local destination address
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<ROUTABLE_IP_ADDR_1> <ROUTABLE_IP_ADDR_2> <ARBITRARY-IP-LINKLOCAL-ADDR> <MAC-ADDR1> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 6. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - 'yiaddr' field set to <ROUTABLE_IP_ADDR_2> - Message Option containing: - Type field set to DHCP_IP_ADDRESS_LEASE_TIME - Length field set to 4 - Value set to <PARAM_LISTEN_TIME*3> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through

	<DIface-0> containing: - Source IP Address set to <ROUTABLE_IP_ADDR_1> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> for ARP response message 11. DUT: Sends ARP Response - Destination IP Address set to <ROUTABLE_IP_ADDR_1> 12. DUT CONFIGURE: Configure DUT to add a static route with - Destination Prefix set to <ARBITRARY-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <ROUTABLE_IP_ADDR_2> 13. TESTER: <SERVER-1> Sends ICMP Echo Request Message to DUT through <DIface-0> containing: - Source IP Address set to <ARBITRARY-IP-LINKLOCAL-ADDR> - Destination IP Address set to <ROUTABLE_IP_ADDR_2> - Ethernet Destination Hardware Address set to <DIFACE-0-MAC-ADDR> - Ethernet Source Hardware Address set to <MAC-ADDR1> 14. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <ROUTABLE_IP_ADDR_2> - Destination IP Address set to <ARBITRARY-IP-LINKLOCAL-ADDR> 15. DUT: Sends ICMP Echo Reply Message 16. CLEANUP: Configure DUT to delete a static route on <DIface-0> with - Destination Prefix set to <ARBITRARY-IP-LINKLOCAL-ADDR> - Netmask set to HOST_MASK - Outgoing interface set to DUT_IFACE_0 - Gateway address set to <ROUTABLE_IP_ADDR_2> 17. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ARP Response 15. DUT: Sends ICMP Echo Reply Message
Test Iterations	
Notes	Derived from RFC 3927 p20 Section 3.3 (MAY)

4.5.6.8 Healing of Network Partitions

IPv4_AUTOCONF_NETWORK_PARTITIONS_01: Healing of Network Partitions Hosts

Synopsis	Hosts SHOULD NOT send periodic gratuitous ARPs (Note: for Link-Local Addresses).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIFACE-0-IP-LINKLOCAL-ADDR> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally cause DUT to bring up <DIface-0> 3. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Expected Network Address of Target IP Address set to <LINK-LOCAL-NET-ADDR> 6. DUT: Sends 3 ARP Request Messages 7. TESTER: <SERVER-1> retrieves the value of Destination Protocol Address field of ARP Request Message received from <DIface-0> and stores it in <DIFACE-0-IP-LINKLOCAL-ADDR> 8. TESTER: Wait till <ANNOUNCE_WAIT+ANNOUNCE_INTERVAL> for DUT to assign the link-local address to <DIface-0> 9. TESTER: <SERVER-1> Sends ARP Request Message to DUT through <DIface-0> containing: - Source IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 10. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <AIFACE-0-IP-LINKLOCAL-ADDR> 11. DUT: Sends ARP Response Message 12. TESTER: <SERVER-1> Listens (up to <ParamListenTime>) on <DIface-0> - Source IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> - Destination IP Address set to <DIFACE-0-IP-LINKLOCAL-ADDR> 13. DUT: Does not send ARP Request Message 14. CLEANUP: Externally cause DUT to shutdown <DIface-0>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends 3 ARP Request Messages 11. DUT: Sends ARP Response Message 13. DUT: Does not send ARP Request Message

Test Iterations	
Notes	Derived from RFC 3927 p23 Section 4 (SHOULD)

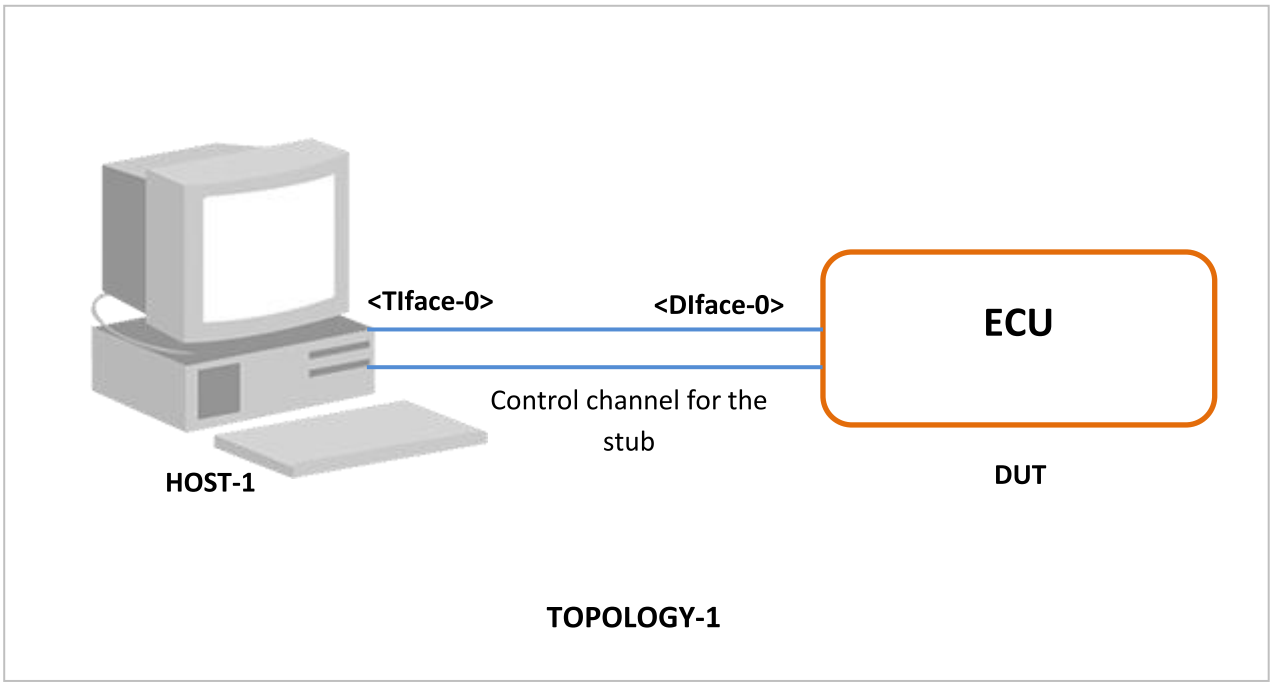
4.6 User Datagram Protocol (UDP)

4.6.1 General

The scope of this chapter is to specify test cases for the User Datagram Protocol (UDP) from the following standards:

- RFC 768 - User Datagram Protocol

4.6.2 Simulated topologies



4.6.3 Required topology related configuration

- This test suite expects to be running against an UDP stack
- This test suite runs over Ethernet
- All tests run with one interface, except 2 (these 2 needs discussion)

4.6.4 Parameters used in the tests

Parameter used in test	Description
<testerUDPPort>	Port number used for UDP in TESTER

<testerUDPPort2>	Another port number used for UDP in TESTER
<unusedUDPSrcPort>	Unused UDP-source port number used in TESTER [Value : 20000]
<unusedUDPDstPort1>	Unused UDP-destination port number used in TESTER [Value : 20001]
<udpUserDataSize>	UDP-data size used by the TESTER [Value : 101]
<calculatedUDPChecksum>	UDP checksum of a UDP packet
<incorrectUDPChecksum>	Incorrect UDP checksum [Value : 0xffff]
<UDPDefaultData>	Default data used in UDP message by TESTER
<UDPDefaultDataLen>	Length of UDP default data
<UDPData>	UDP data used by TESTER [Value : TESTERTESTERTESTERTESTER\0]
<UDPDataLen>	UDP data length [Value : 17]
<API_SUCCESS>	Indication to API status is successful [Value : SUCCESS]
<allSystemMCastAddr>	Multicast address for all systems
<AIface-0-BcastIP>	Broadcast address of the TESTER's 0th interface's network.
<Host-1-IP>	IP address from unused network
<Host-2-IP>	Another IP address from unused network
<DUTSupportsDynamicInterface>	Automotive ECUs may not support dynamic user interface for new port creation, either for performance or security reasons. TRUE indicates DUT supports dynamic user interface FALSE indicates DUT does not support dynamic user interface Default: FALSE

4.6.5 Tests

4.6.5.1 UDP Message Format

DEPRECATED – UDP_MessageFormat_01: To verify that IUT generates an UDP packet containing a well-formed UDP header.

Synopsis	Ensure that when the DUT is requested to generate a UDP packet, then the DUT generates a UDP Packet containing a well-formed Header containing a Source Port field containing a Total Length containing a Destination Port containing a Length field containing a Checksum field
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send <sendUdpPacket> command to the DUT to request a UDP packet. 2. DUT: Generates a UDP packet.
Pass Criteria	The DUT generates a UDP Packet containing a well-formed Header containing a Source Port field containing a Destination Port indicating a value equal to the TS's UDP port. containing a Length field indicating a valid value equal to the size of the received datagram containing a Checksum field indicating a value equal to the <calculatedUDPChecksum>
Test Iterations	
Reference	Derived from RFC 768, section Format
Notes	

UDP_MessageFormat_02: To verify that IUT accepts an UDP packet containing a well-formed UDP header.

Synopsis	Ensure that

	when the DUT receives a UDP packet containing a well-formed Header containing a Source Port field containing a Destination Port indicating a value equal to the DUT's UDP port containing a Length field indicating a valid value equal to the size of the sent datagram containing a Checksum field indicating a value equal to the <calculatedUDPChecksum> by the DUT then the DUT accepts the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet containing a well-formed UDP Header 2. DUT: Sends <Indication> containing the same content of the UDP packet received.
Pass Criteria	The DUT accepts the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Format
Notes	

4.6.5.2 UDP Port Handling

DEPRECATED – UDP_PortHandling_01: To verify that the DUT accepts the datagram in case UDP Header has a non-zero source port.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Source Port Field indicating a value different from zero then the DUT accepts the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with a non-zero Source Port field. 2. DUT: Sends <Indication> containing the same content of the UDP packet received.
Pass Criteria	The DUT accepts the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

DEPRECATED – UDP_PortHandling_02: To verify that the DUT accepts the datagram in case UDP Header has a source port with a value equal to zero.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Source Port Field indicating a value equal to zero then the DUT accepts the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with a Source Port field equal to zero. 2. DUT: Sends <Indication> containing the same content of the UDP packet received.
Pass Criteria	The DUT accepts the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

DEPRECATED – UDP_PortHandling_03: To verify that the DUT accepts the datagram in case UDP Header has a specific source port value.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Source Field indicating an acceptable specific value then the DUT accepts the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with a specific Source Port field value. 2. DUT: Sends <Indication> containing the same content of the UDP packet received.
Pass Criteria	The DUT accepts the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

DEPRECATED – UDP_PortHandling_04: To verify that the DUT discards a datagram in case UDP Header has a non-acceptable source port value.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Source Field indicating a value different from the <rangeOfAcceptablePorts> then the DUT discards the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with a valid Source Port field. 2. DUT: Discards the UDP packet.
Pass Criteria	The DUT discards the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

4.6.5.3 UDP Datagram Length

UDP_DatagramLength_01: To verify that IUT discards a truncated UDP datagram.

Synopsis	Ensure that when the DUT receives a truncated UDP packet (a packet with the length field smaller than the actual size of the data coming from the Ethernet frame) then the DUT discards the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a truncated UDP packet. 2. DUT: Discards the UDP packet and sends no <Indication>.
Pass Criteria	The DUT discards the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Format and automotive specific requirements on frame integrity
Notes	

DEPRECATED – UDP_DatagramLength_02: To verify that IUT discards a UDP datagram in case datagram total length less than 8.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Length field indicating a value less than 8 then the DUT discards the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with Length greater than or equal to 8. 2. DUT: Discards the UDP packet and sends no <Indication>.
Pass Criteria	The DUT discards the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

DEPRECATED – UDP_DatagramLength_03: To verify that IUT accepts an UDP datagram in case datagram total length greater than 8.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Length field indicating a value greater than or equal to 8 then the DUT accepts the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with Length greater than or equal to 8. 2. DUT: Sends <Indication> containing the same content of the UDP packet received.
Pass Criteria	The DUT accepts the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

DEPRECATED – UDP_DatagramLength_04: To verify that IUT discards a UDP datagram in case datagram total length equal to 0.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Length field indicating a value equal to 0 then the DUT discards the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with Length equal to zero. 2. DUT: Discards UDP packet and sends no <Indication>.
Pass Criteria	The DUT discards the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

DEPRECATED – UDP_DatagramLength_05: To verify that IUT discards a UDP datagram in case datagram total length greater than the size of the datagram.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Length field indicating a value greater than the size of the packet then the DUT discards the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with Length greater than the packet size. 2. DUT: Discards the UDP packet and sends no <Indication>.
Pass Criteria	The DUT discards the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

DEPRECATED – UDP_DatagramLength_06: To verify that IUT discards a UDP datagram in case datagram total length is less than the size of the datagram.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Length field indicating a value less than the size of the packet then the DUT discards the UDP packet.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with Length field less than the size of the packet. 2. DUT: Discards the UDP packet and sends no <Indication>.
Pass Criteria	The DUT discards the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

DEPRECATED – UDP_DatagramLength_07: To verify that IUT discards an UDP datagram in case datagram total length is set to maximum number allowed by the field.

Synopsis	Ensure that when the DUT receives a UDP packet containing a UDP Header containing a Length field indicating a value equal to the maximum size then the DUT discards the UDP packet and sends no <Indication>.
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a UDP packet with Length field set to maximum size. 2. DUT: discards the UDP packet.
Pass Criteria	The DUT discards the UDP packet.
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

4.6.5.4 UDP Padding

DEPRECATED – UDP_Padding_01: To verify that IUT generates UDP datagram with odd size of payload and adds padding at the end.

Synopsis	Ensure that when the DUT is requested to generate a UDP packet with an odd <udpUserDataSize> then the DUT generates a UDP packet containing Data indicating value of the received odd payload size plus padding bytes
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send <sendUdpPacketOddSize> to the DUT to request a UDP packet with odd <udpUserDataSize>. 2. DUT: Generates a UDP packet with padding.
Pass Criteria	The DUT generates a UDP packet containing Data indicating value of the received odd <udpUserDataSize> plus padding bytes
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

UDP_Padding_02: To verify that IUT generates UDP datagram with even size of payload and no padding at the end.

Synopsis	Ensure that when the DUT is requested to generate a UDP packet with an even payload size then the DUT generates a UDP packet containing Data indicating value of the received even <udpUserDataSize> with no padding bytes
Prerequisites	None
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a <sendUdpPacketEvenSize> to the DUT to request a UDP packet with an even <udpUserDataSize>. 2. DUT: Generates a UDP packet without padding.
Pass Criteria	The DUT generates a UDP packet containing Data indicating value of the received an even <udpUserDataSize> with no padding bytes
Test Iterations	
Reference	Derived from RFC768, section Fields
Notes	

4.6.5.5 UDP Fields

UDP_FIELDS_01: Fields – Specify Source Port

Synopsis	Source Port is an optional field, when meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed in the absence of any other information. A user interface should allow the creation of new receive ports [Note: In this test, we verify that application can specify source port]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with source port set to <unusedUDPDstPort1> through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Source UDP Port field is set to <unusedUDPDstPort1>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Source UDP Port field is set to <unusedUDPDstPort1>
Test Iterations	
Notes	Derived from RFC 768 Page 1 'Fields' (SHOULD)

UDP_FIELDS_02: Fields – Specify Destination Port

Synopsis	Source Port is an optional field, when meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed in the absence of any other information. [Note: In this test, we verify that DUT can respond respond back (if required) to source port of received message. This has two parts: being able to read the source port of a received message, and being able to set the destination port of a sent message - these operations are available on the AUTOSAR testability protocol. In this case, the tester is basically implementing the application logic of issuing the reply by properly reading the source port and replying to that port.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<unusedUDPSrcPort> <UDPDefaultData> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Source UDP Port field set to <unusedUDPSrcPort> - Destination UDP Port field set to <unusedUDPDstPort1> 4. TESTER: Externally cause DUT to send a UDP Message with <UDPDefaultData> as data through <DIface-0> 5. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 6. DUT: Sends Message 7. TESTER: Verify that received UDP Message contains: - Destination UDP Port field is set to <unusedUDPSrcPort>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 6. DUT: Sends Message 7. TESTER: Verify that received UDP Message contains: - Destination UDP Port field is set to <unusedUDPSrcPort>
Test Iterations	
Notes	Derived from RFC 768 Page 1 'Fields' (MUST)

UDP_FIELDS_03: Fields - Accept Source Port set to zero

Synopsis	Source Port is an optional field, when meaningful, it indicates the port of the sending process, and may be assumed to be the port to which a reply should be addressed in the absence of any other information. If not used, a value of zero is inserted. [Note: In this test, we verify that using the value 0 for UDP Source Port is valid and DUT accepts such a Message]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Source UDP Port field set to 0 - Destination UDP Port field set to <unusedUDPDstPort1> 4. TESTER: Verify using Upper Tester that DUT received UDP Message containing: - UDP Source Port field set to 0
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that DUT received UDP Message containing: - UDP Source Port field set to 0
Test Iterations	
Notes	Derived from RFC 768 Page 1 'Fields' (MUST)

UDP_FIELDS_04: Fields - Same Destination Port with Different IP Address (send)

Synopsis	Destination Port has a meaning within the context of a particular internet destination address. [Note: In this test, we verify that DUT can send UDP message at same destination port to more than one different IP addresses.]
Prerequisites	Check section prerequisites
Test setup	Topology 2
Test Input Parameters	<testerUDPPort> <Host-1-IP> <Host-2-IP> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to send a UDP Message at <testerUDPPort> as the destination port and to <Host-1-IP> IP Address through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Destination IP Address field is set to <Host-1-IP> - Destination UDP Port field is set to <testerUDPPort> 5. DUT CONFIGURE: Externally cause DUT to send a UDP Message at <testerUDPPort> as the destination port and to <Host-2-IP> IP Address through <DIface-0> 6. TESTER: <HOST-2> Listens (up to<ParamListenTime>) on <DIface-0> 7. DUT: Sends Message 8. TESTER: Verify that received UDP Message contains: - Destination IP Address field is set to <Host-2-IP> - Destination UDP Port field is set to <testerUDPPort>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Destination IP Address field is set to <Host-1-IP> - Destination UDP Port field is set to <testerUDPPort> 7. DUT: Sends Message 8. TESTER: Verify that received UDP Message contains: - Destination IP Address field is set to <Host-2-IP> - Destination UDP Port field is set to <testerUDPPort>
Notes	Derived from RFC 768 Page 1 'Fields' (MUST)

UDP_FIELDS_05: Fields - Same Port with Different IP Address (receive and send)

Synopsis	Destination Port has a meaning within the context of a particular internet destination address. [Note: In this test, we verify that DUT can receive send UDP message at same destination port to more than one differrent IP addresses.]
Prerequisites	Check section prerequisites
Test setup	Topology 2
Test Input Parameters	<UDPDefaultData> <UDPData> <Host-1-IP> <Host-2-IP> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Source IP Address field set to <Host-1-IP> - Destination IP Address field set to <DIface-0-IP> - Destination UDP Port field set to <unusedUDPDstPort1> - UDP send data set to <UDPData> 4. TESTER: Verify using Upper Tester that application layer has got UDP Message containing: - UDP data equal to <UDPData> 5. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 6. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 7. TESTER: <HOST-2> Sends Message to DUT through <DIface-0> containing: - Source IP Address field set to <Host-2-IP> - Destination IP Address field set to <DIface-0-IP> - Destination UDP Port field set to <unusedUDPDstPort1> - UDP send data set to <UDPDefaultData> 8. TESTER: Verify using Upper Tester that application layer has got UDP Message containing: - UDP data equal to <UDPDefaultData>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer has got UDP Message containing: - UDP data equal to <UDPData> 6. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 8. TESTER: Verify using Upper Tester that application layer has got UDP Message containing: - UDP data equal to <UDPDefaultData>

Test Iterations	
Notes	Derived from RFC 768 Page 1 'Fields' (MUST)

UDP_FIELDS_06: Fields - Total Length

Synopsis	Length is the length in octets of this user datagram including this header and the data. [Note: In this test, we configure DUT to send <udpUserDataSize> size of data so that the length in Header is (<udpUserDataSize > + 8(UDP Header Size)) bytes]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<udpUserDataSize> Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send UDP Message with <udpUserDataSize> of data through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Length field is set to (<udpUserDataSize>+8)
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Length field is set to (<udpUserDataSize>+8)
Test Iterations	
Notes	Derived from RFC 768 Page 2 'Fields' (MUST)

UDP_FIELDS_07: Fields - Total Length (no data)

Synopsis	Length is the length in octets of this user datagram including this header and the data. [Note: In this test, we configure DUT to send 0 (zero) size of data so that the length in Header is 8(UDP Header Size)) bytes]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send UDP Message with 0 of data through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Length field is set to 8
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Length field is set to 8
Test Iterations	
Notes	Derived from RFC 768 Page 2 'Fields' (MUST)

UDP_FIELDS_08: Fields - Total Length (less than 8 bytes)

Synopsis	Length is the length in octets of this user datagram including this header and the data. [Note: Check that the DUT discards the received datagram in case the total length of the datagram is less than 8 bytes]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - truncated message: length less than 8 bytes - No UDP Data set. 4. TESTER: Verify using Upper Tester that DUT discards that UDP message.
Pass Criteria	4. TESTER: Verify using Upper Tester that DUT discards that UDP message.
Test Iterations	
Notes	Derived from RFC 768 Page 2 'Fields' (MUST)

UDP_FIELDS_09: Fields - Total Length (equal to zero)

Synopsis	Length is the length in octets of this user datagram including this header and the data. [Note: Check that the DUT discards the received datagram in case the total length of the datagram is zero (‘0’)]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - UDP Data set to <UDPData>. - length field set to 0 4. TESTER: Verify using Upper Tester that DUT discards that UDP message.
Pass Criteria	4. TESTER: Verify using Upper Tester that DUT discards that UDP message.
Test Iterations	
Notes	Derived from RFC 768 Page 2 'Fields' (MUST)

UDP_FIELDS_10: Fields - Total Length (greater than actual)

Synopsis	Length is the length in octets of this user datagram including this header and the data. [Note: Check that the DUT discards the received datagram in case the length value in the header is greater than the actual length of the datagram]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - length field set to (Length of <UDPData> + 8) + 1 - UDP Data set to <UDPData>. 4. TESTER: Verify using Upper Tester that DUT discards that UDP message.
Pass Criteria	4. TESTER: Verify using Upper Tester that DUT discards that UDP message.
Test Iterations	
Notes	Derived from RFC 768 Page 2 'Fields' (MUST)

UDP_FIELDS_12: Fields - Total Length (maximum)

Synopsis	Length is the length in octets of this user datagram including this header and the data. [Note: Check that the DUT accepts the received datagram in case the length value in the header is set to the maximum allowed value. - IPv4: 65,507 bytes (65,535 – 8 byte UDP header – 20 byte IP header) - IPv6: 65,535 bytes]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Length field set to maximum supported value - data set to maximum supported octet size 4. TESTER: Verify using Upper Tester that DUT has received the UDP message.
Pass Criteria	4. TESTER: Verify using Upper Tester that DUT has received the UDP message.
Test Iterations	
Notes	Derived from RFC 768 Page 2 'Fields' (MUST)

UDP_FIELDS_13: Fields - Checksum (with padding)

Synopsis	Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets. [Note: In this test, we verify that DUT calculates UDP checksum correctly. While calculating UDP checksum the padded byte is needed to be considered.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<udpUserDataSize> <calculatedUDPChecksum> Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send UDP Message with <udpUserDataSize> of data through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to <calculatedUDPChecksum>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to <calculatedUDPChecksum>
Test Iterations	
Notes	Derived from RFC 768 Page 1 'Fields' (MUST)

UDP_FIELDS_14: Fields - Checksum (no padding)

Synopsis	Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets. [Note: In this test, we verify that DUT calculates UDP checksum correctly. While calculating UDP checksum the padded byte is not required.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<calculatedUDPChecksum> Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send UDP Message with 100 of data through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to <calculatedUDPChecksum>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to <calculatedUDPChecksum>
Test Iterations	
Notes	Derived from RFC 768 Page 1 'Fields' (MUST)

UDP_FIELDS_15: Fields - Checksum (incorrect)

Synopsis	If a UDP datagram is received with a checksum that is non-zero and invalid, UDP MUST silently discard the datagram. [Note: In this test, we verify that DUT will not accept UDP message with incorrect checksum]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<incorrectUDPChecksum> <UDPDefaultData> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Destination UDP Port field set to <unusedUDPDstPort1> - UDP Header Checksum field set to <incorrectUDPChecksum> - UDP send data set to <UDPDefaultData> 4. TESTER: Verify using Upper Tester that application layer did not get UDP Message containing: - UDP data equal to <UDPDefaultData>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer did not get UDP Message containing: - UDP data equal to <UDPDefaultData>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.4 Page 78 'UDP Checksums' (MUST)

UDP_FIELDS_16: Fields - Checksum (zero checksum)

Synopsis	An all zero transmitted checksum value means that the transmitter generated no checksum (for debugging or for higher level protocols that don't care). An application MAY optionally be able to control whether a UDP checksum will be generated, but it MUST default to checksumming on. [Note: In this test, we verify that DUT accepts UDP datagram with zero checksum]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<UDPDefaultData> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Destination UDP Port field set to <unusedUDPDstPort1> - UDP Header Checksum field set to 0 - UDP send data set to <UDPDefaultData> 4. TESTER: Verify using Upper Tester that application layer has got UDP Message containing: - UDP data equal to <UDPDefaultData>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer has got UDP Message containing: - UDP data equal to <UDPDefaultData>
Test Iterations	
Notes	Derived from RFC 768 Page 1 'Fields' (MUST)

DEPRECATED – UDP_FIELDS_17: Fields - Checksum (no checksum)

Synopsis	An all zero transmitted checksum value means that the transmitter generated no checksum (for debugging or for higher level protocols that don't care). An application MAY optionally be able to control whether a UDP checksum will be generated, but it MUST default to checksumming on. [Note: In this test, we verify that application on DUT can generate UDP datagram with no-checksum]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with no-checksum through <DIface-0>. 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to 0
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to 0
Test Iterations	
Notes	Derived from RFC 768 Page 1 'Fields' (MUST)

DEPRECATED – UDP_FIELDS_18: Fields - Checksum (controlled)

Synopsis	An application MAY optionally be able to control whether a UDP checksum will be generated, but it MUST default to checksumming on. [Note: In this test, we verify that application on DUT can turn UDP checksum on while generating UDP datagram.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<calculatedUDPChecksum> Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with no-checksum through <DIface-0>. 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to 0 5. TESTER: Externally cause DUT to send a UDP Message with UDP-checksum turn on through <DIface-0>. 6. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 7. DUT: Sends Message 8. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to <calculatedUDPChecksum>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to 0 7. DUT: Sends Message 8. TESTER: Verify that received UDP Message contains: - UDP Header Checksum field is set to <calculatedUDPChecksum>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.4 Page 78 'UDP Checksums' (MUST)

4.6.5.6 User Interface

UDP_USER_INTERFACE_01: User Interface - New Receive Port

Synopsis	A user interface should allow the creation of new receive ports, receive operations on the receive ports that return the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent. [Note: In this test, we verify that user interface allows creation of new receive ports. This test is only ran when <DUTSupportsDynamicInterface> is TRUE]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to create 10 receive ports on <DIface-0> 2. DUT: Create 10 receive ports on <DIface-0> 3. TESTER: Verify using Upper Tester that application layer of DUT has created 10 receive ports on <DIface-0>
Pass Criteria	2. DUT: Create 10 receive ports on <DIface-0> 3. TESTER: Verify using Upper Tester that application layer of DUT has created 10 receive ports on <DIface-0>
Test Iterations	
Notes	Derived from RFC 768 Page 2 'User Interface' (MUST)

UDP_USER_INTERFACE_02: User Interface - Data octets

Synopsis	A user interface should allow the creation of new receive ports, receive operations on the receive ports that return the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent. [Note: In this test, we verify that receive operations on the receive ports return the data octets correctly. This test is only ran when <DUTSupportsDynamicInterface> is TRUE]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<UDPDefaultData> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Destination UDP Port field set to <unusedUDPDstPort1> - UDP send data set to <UDPDefaultData> 4. TESTER: Verify using Upper Tester that application layer has got UDP Message containing: - UDP data equal to <UDPDefaultData>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer has got UDP Message containing: - UDP data equal to <UDPDefaultData>
Test Iterations	
Notes	Derived from RFC 768 Page 2 'User Interface' (MUST)

UDP_USER_INTERFACE_03: User Interface – Return Source Port

Synopsis	A user interface should allow the creation of new receive ports, receive operations on the receive ports that return the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent. [Note: In this test, we verify that receive operations on the receive ports return the source port correctly. This test is only ran when <DUTSupportsDynamicInterface> is TRUE]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<unusedUDPSrcPort> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Source UDP Port field set to <unusedUDPSrcPort> - Destination UDP Port field set to <unusedUDPDstPort1> 4. TESTER: Verify using Upper Tester that DUT received UDP Message containing: - UDP Source Port field set to <unusedUDPSrcPort>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that DUT received UDP Message containing: - UDP Source Port field set to <unusedUDPSrcPort>
Test Iterations	
Notes	Derived from RFC 768 Page 2 'User Interface' (MUST)

UDP_USER_INTERFACE_04: User Interface – Return Source IP Address

Synopsis	A user interface should allow the creation of new receive ports, receive operations on the receive ports that return the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent. [Note: In this test, we verify that receive operations on the receive ports return the source address correctly. This test is only ran when <DUTSupportsDynamicInterface> is TRUE]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Source IP Address field set to <AIface-0-IP> - Destination UDP Port field set to <unusedUDPDstPort1> 4. TESTER: Verify using Upper Tester that application layer received an IP Packet containing a UDP Message containing: - IP source address equal to <AIface-0-IP>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer received UDP Message containing: - UDP source address equal to <AIface-0-IP>
Test Iterations	
Notes	Derived from RFC 768 Page 2 'User Interface' (MUST)

UDP_USER_INTERFACE_05: User Interface - Source Port (to be sent)

Synopsis	A user interface should allow the creation of new receive ports, receive operations on the receive ports that return the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent. [Note: In this test, we verify that an operation that allows a datagram to be sent, specifies the source port to be sent. This test is only ran when <DUTSupportsDynamicInterface> is TRUE]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<unusedUDPSrcPort> Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with source port set to <unusedUDPSrcPort> through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Source UDP Port field is set to <unusedUDPSrcPort>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Source UDP Port field is set to <unusedUDPSrcPort>
Test Iterations	
Notes	Derived from RFC 768 Page 2 'User Interface' (MUST)

UDP_USER_INTERFACE_06: User Interface - Destination Port (to be sent)

Synopsis	A user interface should allow the creation of new receive ports, receive operations on the receive ports that return the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent. [Note: In this test, we verify that an operation that allows a datagram to be sent, specifies the destination port to be sent. This test is only ran when <DUTSupportsDynamicInterface> is TRUE]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<testerUDPPort> Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with destination port set to <testerUDPPort> through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Destination UDP Port field is set to <testerUDPPort>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Destination UDP Port field is set to <testerUDPPort>
Test Iterations	
Notes	Derived from RFC 768 Page 2 'User Interface' (MUST)

UDP_USER_INTERFACE_07: User Interface - Source IP Address (to be sent)

Synopsis	A user interface should allow the creation of new receive ports, receive operations on the receive ports that return the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent. [Note: In this test, we verify that an operation that allows a datagram to be sent, specifies the source address to be sent. This test is only ran when <DUTSupportsDynamicInterface> is TRUE]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with source address set to <DIface-0-IP> through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Source IP Address field is set to <DIface-0-IP>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Source IP Address field is set to <DIface-0-IP>
Test Iterations	
Notes	Derived from RFC 768 Page 2 'User Interface' (MUST)

UDP_USER_INTERFACE_08: User Interface - Destination Address (to be sent)

Synopsis	A user interface should allow the creation of new receive ports, receive operations on the receive ports that return the data octets and an indication of source port and source address, and an operation that allows a datagram to be sent, specifying the data, source and destination ports and addresses to be sent. [Note: In this test, we verify that an operation that allows a datagram to be sent, specifies the destination address to be sent. This test is only ran when <DUTSupportsDynamicInterface> is TRUE]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with destination address set to <AIface-0-IP> through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received IP Packet containing a UDP Message containing: - Destination IP Address field is set to <AIface-0-IP>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received IP Packet containing a UDP Message containing: - Destination IP Address field is set to <AIface-0-IP>
Test Iterations	
Notes	Derived from RFC 768 Page 2 'User Interface' (MUST)

4.6.5.7 IP Interface

UDP_IP_INTERFACE_01: IP Interface – TTL Field

Synopsis	One possible UDP/IP interface would return the whole internet datagram including all of the internet header in response to a receive operation. Such an interface would also allow the UDP to pass a full internet datagram complete with header to the IP to send. The IP would verify certain fields for consistency and compute the internet header checksum. [Note: In this test we verify that application is notified about TTL value of received UDP message]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: Set IP/IPv6 TTL/Hoplimit value with <defaultIPTTL> 4. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Destination UDP Port field set to <unusedUDPDstPort1> 5. TESTER: Verify using Upper Tester that DUT application layer received UDP Message containing: - IP TTL equal to <defaultIPTTL>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 5. TESTER: Verify using Upper Tester that DUT application layer received UDP Message containing: - IP TTL equal to <defaultIPTTL>
Test Iterations	
Notes	Derived from RFC 768 Page 3 'IP Interface' (MUST)

4.6.5.8 Introduction

UDP_INTRODUCTION_01: Introduction – Broadcast Destination Address

Synopsis	UDP is used by applications that do not require the level of service of TCP or that wish to use communications services (e.g., multicast or broadcast delivery) not available from TCP. [Note: In this test we verify that DUT will accept UDP message with broadcast destination Address.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<AIface-0-BcastIP> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Destination IP Address field set to <AIface-0-BcastIP> - Destination UDP Port field set to <unusedUDPDstPort1> 4. TESTER: Verify using Upper Tester that application layer received UDP Message containing: - destination address equal to <AIface-0-BcastIP>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer received UDP Message containing: - destination address equal to <AIface-0-BcastIP>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.1 Page 77 'Introduction' (SHOULD)

UDP_INTRODUCTION_02: Introduction – Multicast Destination Address

Synopsis	UDP is used by applications that do not require the level of service of TCP or that wish to use communications services (e.g., multicast or broadcast delivery) not available from TCP. [Note: In this test we verify that DUT will accept UDP message with multicast destination Address.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<allSystemMCastAddr> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Destination IP Address field set to <allSystemMCastAddr> - Destination UDP Port field set to <unusedUDPDstPort1> 4. TESTER: Verify using Upper Tester that application layer received UDP Message containing: - destination address equal to <allSystemMCastAddr>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer received UDP Message containing: - destination address equal to <allSystemMCastAddr>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.1 Page 77 'Introduction' (SHOULD)

UDP_INTRODUCTION_03: Introduction – Pending Listen Call

Synopsis	If a datagram arrives addressed to a UDP port for which there is no pending LISTEN call, UDP SHOULD send an ICMP Port Unreachable message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Destination IP Address field set to <DIface-0-IP> - Destination UDP Port field set to <unusedUDPDstPort1> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends <ICMP-Dest-Unrchbl> Message
Pass Criteria	3. DUT: Sends <ICMP-Dest-Unrchbl> Message
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.1 Page 77 'Ports' (SHOULD)

4.6.5.9 IP Option

UDP_IP_OPTION_01: IP Option - Time Stamp Option (pass any option)

Synopsis	UDP MUST pass any IP option that it receives from the IP layer transparently to the application layer. [Note: In this test, we verify 'Time Stamp' IP Option]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Destination UDP Port field set to <unusedUDPDstPort1> - Time Stamp IP Option 4. TESTER: Verify using Upper Tester that application layer has received UDP message containing: - IP Header contains: - <IP_OPT_TIME_STAMP> IP option 5.
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer has received UDP message containing: - IP Header contains: - <IP_OPT_TIME_STAMP> IP option
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.2 Page 77 'IP Option' (MUST)

UDP_IP_OPTION_02: IP Option - Time Stamp Option (to be sent)

Synopsis	An application MUST be able to specify IP options to be sent in its UDP datagrams, and UDP MUST pass these options to the IP layer. [Note: In this test, we verify 'Time Stamp' IP Option]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message containig <IP_OPT_TIME_STAMP> IP Option through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Time Stamp IP Option is set to <IP_OPT_TIME_STAMP>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Time Stamp IP Option is set to <IP_OPT_TIME_STAMP>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.2 Page 77 'IP Option' (MUST)

4.6.5.10 UDP Multihoming

UDP_MULTI_HOMING_01: UDP Multihoming - appropriate source address

Synopsis	An application program MUST be able to specify the IP source address to be used for sending a UDP datagram or to leave it unspecified (in which case the networking software will choose an appropriate source address). [Note: In this test, we verify that the networking software has chosen an an appropriate source address.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with destination address set to <AIface-0-IP> through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Source IP Address field is set to <DIface-0-IP>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - Source IP Address field is set to <DIface-0-IP>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.5 Page 79 'UDP Multihoming' (MUST)

UDP_MULTI_HOMING_02: UDP Multihoming - specific destination address

Synopsis	A request/response application that uses UDP should use a source address for the response that is the same as the specific destination address of the request.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<UDPDefaultData> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Source IP Address field set to <AIface-0-IP> - Destination IP Address field set to <DIface-0-IP> - Destination UDP Port field set to <unusedUDPDstPort1> 4. TESTER: Externally cause DUT to send a UDP Message with <UDPDefaultData> as data through <DIface-0> 5. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 6. DUT: Sends Message 7. TESTER: Verify that received UDP Message contains: - Destination IP Address field is set to <AIface-0-IP>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 6. DUT: Sends Message 7. TESTER: Verify that received UDP Message contains: - Destination IP Address field is set to <AIface-0-IP>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.5 Page 79 'UDP Multihoming' (MUST)

4.6.5.11 Invalid Addresses

UDP_INVALID_ADDRESSES_01: Invalid Addresses - multicast source address

Synopsis	A UDP datagram received with an invalid IP source address (e.g., a broadcast or multicast address) must be discarded by UDP or by the IP layer (see Section 3.2.1.3). [Note: In this test, we verify UDP Message with multicast address as source address.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<UDPDefaultData> <allSystemMCastAddr> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Source IP Address field set to <allSystemMCastAddr> - Destination UDP Port field set to <unusedUDPDstPort1> - UDP send data set to <UDPDefaultData> 4. TESTER: Verify using Upper Tester that application layer did not get UDP Message containing: - UDP data equal to <UDPDefaultData>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer did not get UDP Message containing: - UDP data equal to <UDPDefaultData>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.6 Page 79 'Invalid Addresses' (MUST)

UDP_INVALID_ADDRESSES_02: Invalid Addresses - broadcast source address

Synopsis	A UDP datagram received with an invalid IP source address (e.g., a broadcast or multicast address) must be discarded by UDP or by the IP layer (see Section 3.2.1.3). [Note: In this test, we verify UDP Message with broadcast address as source address.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<UDPDefaultData> <AIface-0-BcastIP> Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally cause DUT to listen on port <unusedUDPDstPort1> on <DIface-0> 2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 3. TESTER: <HOST-1> Sends Message to DUT through <DIface-0> containing: - Source IP Address field set to <AIface-0-BcastIP> - Destination UDP Port field set to <unusedUDPDstPort1> - UDP send data set to <UDPDefaultData> 4. TESTER: Verify using Upper Tester that application layer did not get UDP Message containing: - UDP data equal to <UDPDefaultData>
Pass Criteria	2. DUT: Listen on port <unusedUDPDstPort1> on <DIface-0> 4. TESTER: Verify using Upper Tester that application layer did not get UDP Message containing: - UDP data equal to <UDPDefaultData>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.6 Page 79 'Invalid Addresses' (MUST)

4.6.5.12 UDP/Application layer interface

UDP_APP_INTERFACE_01: UDP/Application layer interface - TTL

Synopsis	An application-layer program MUST be able to set the TTL and TOS values as well as IP options for sending a UDP datagram, and these values must be passed transparently to the IP layer. [Note: In this test, we verify that UDP-application on DUT can specify TTL value for IP header.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with <defaultIPTTL> as TTL through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - TTL/Hoplimit in IP/IPv6 header is set to <defaultIPTTL>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - TTL/Hoplimit in IP/IPv6 header is set to <defaultIPTTL>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.4 Page 79 'UDP/APPLICATION LAYER INTERFACE' (MUST)

UDP_APP_INTERFACE_02: UDP/Application layer interface - TOS

Synopsis	An application-layer program MUST be able to set the TTL and TOS values as well as IP options for sending a UDP datagram, and these values must be passed transparently to the IP layer. [Note: In this test, we verify that UDP-application on DUT can specify TOS value for IP header.]
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with <defaultIPTOS> as TOS through <DIface-0> 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - TOS in IP header is set to <defaultIPTOS>
Pass Criteria	3. DUT: Sends Message 4. TESTER: Verify that received UDP Message contains: - TOS in IP header is set to <defaultIPTOS>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.4 Page 79 'UDP/APPLICATION LAYER INTERFACE' (MUST)

4.6.5.13 ICMP Messages

UDP_ICMP_MESSAGES_01: ICMP Messages – Error

Synopsis	UDP MUST pass to the application layer all ICMP error messages that it receives from the IP layer.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	<testerUDPPort2> Check section general Input Parameters
Test Procedure	1. TESTER: Externally cause DUT to send a UDP Message with destination port set to <testerUDPPort2> through <DIface-0> and reuse the socket 2. TESTER: <HOST-1> Listens (up to<ParamListenTime>) on <DIface-0> 3. DUT: Sends Message 4. DUT CONFIGURE: Externally cause DUT to listen on port <testerUDPPort2> on <DIface-0> 5. DUT: Listen on port <testerUDPPort2> on <DIface-0> 6. TESTER: <HOST-1> Sends <ICMP-Dest-Unrchbl> Message to DUT through <DIface-0> containing: - Destination IP Address field set to <DIface-0-IP> 7. TESTER: Verify using Upper Tester that DUT application layer received ICMP Error Message containing: - error code equal to <ICMP-Dest-Unrchbl>
Pass Criteria	3. DUT: Sends Message 5. DUT: Listen on port <testerUDPPort2> on <DIface-0> 7. TESTER: Verify using Upper Tester that DUT application layer received ICMP Error Message containing: - error code equal to <ICMP-Dest-Unrchbl>
Test Iterations	
Notes	Derived from RFC 1122 Section 4.1.3.3 Page 78 'ICMP Messages' (MUST)

4.7 Dynamic Host configuration Protocol Version 4 (DHCPv4) Server

DHCPv4 Server Tests are not relevant for Automotive Use Cases.

4.8 Dynamic Host configuration Protocol Version 4 (DHCPv4) Client

4.8.1 General

The scope of this chapter is to specify test cases for the Dynamic Host configuration Protocol Version 4 (DHCPv4) from the following standards:

- RFC 2131 - Dynamic Host Configuration Protocol

Though the focus of conformance testing has been limited to the above mentioned specification document we have relied heavily upon the following document as specification for DHCP Message Options:

- RFC 2132 - DHCP Options and BOOTP Vendor Extensions

4.8.2 Simulated topologies

The number of DUT interfaces required by the topologies described below are as follows:

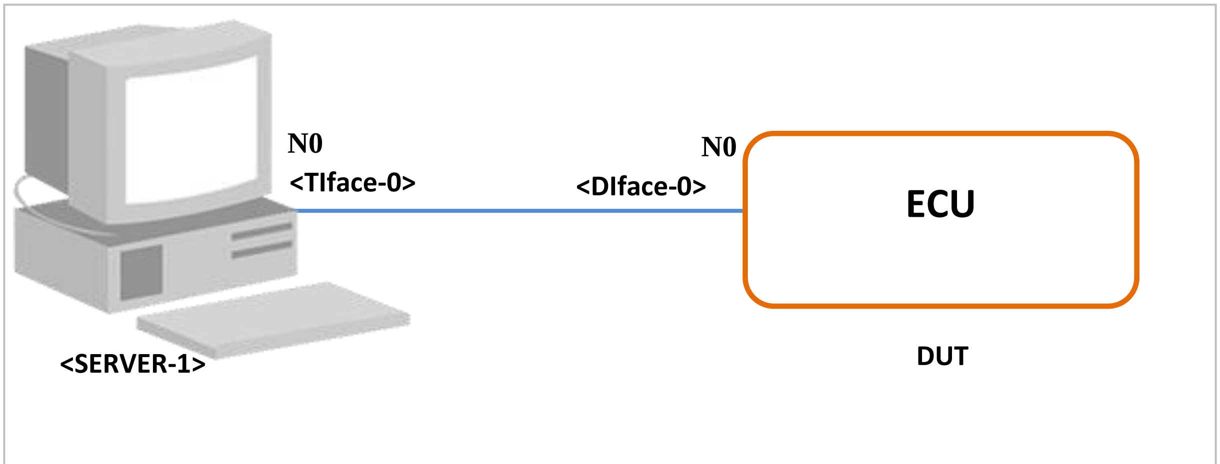
TOPOLOGY ID	Number of DUT Interfaces	Emulated Actors
Topology-1	1	1 Server
Topology-2	2	1 Server
Topology-3	1	2 Servers
Topology-4	1	1 Server 1 Static IP assigned non-DHCP Client

In each test, TESTER simulates a portion of exactly one of the 4 topologies shown below. Each node shown in a topology is represented with a unique identifier. The test methods follow the same representation of these nodes.

- **TOPOLOGY-1**

In this topology, following is being emulated:

- DHCP Server <SERVER-1> connected to DUT through <DIface-0> belonging to network N0

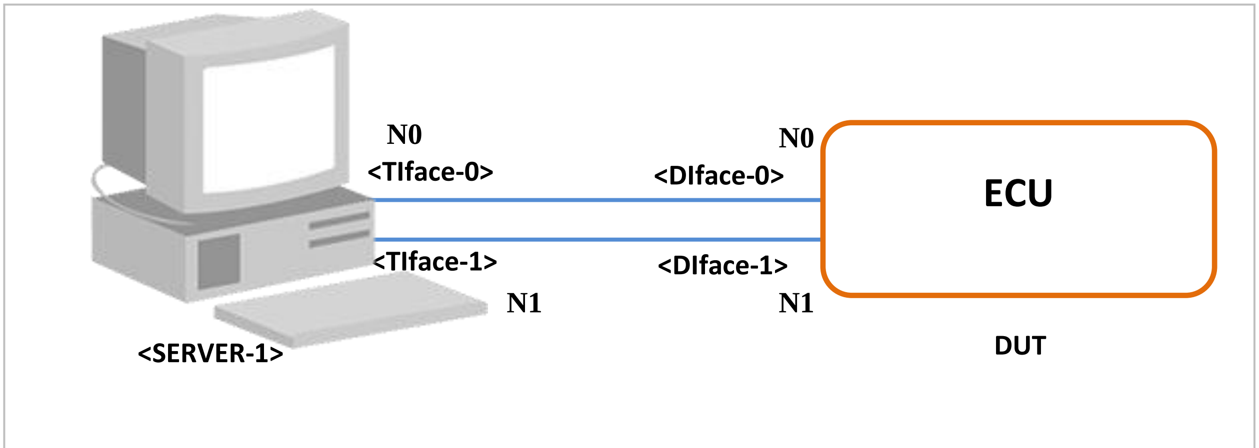


TOPOLOGY-1

- **TOPOLOGY-2**

In this topology, following is being emulated:

- DHCP Server <SERVER-1> connected to DUT through <DIface-0> belonging to network N0
- DHCP Server <SERVER-1> connected to DUT through <DIface-1> belonging to network N1

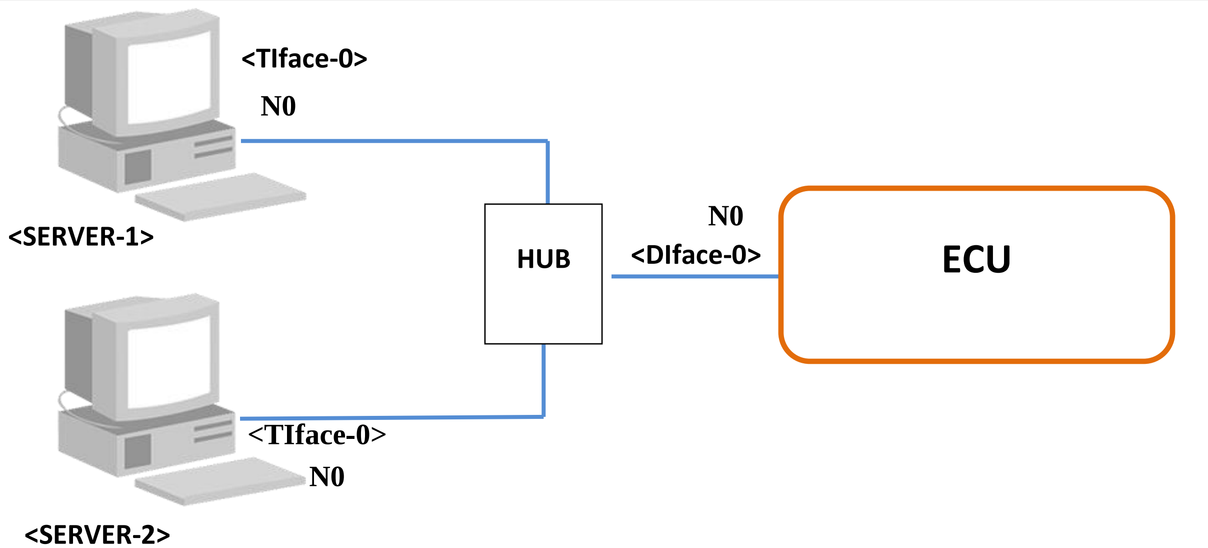


TOPOLOGY-2

- **TOPOLOGY-3**

  In this topology, following is being emulated:
  - DHCP Server <SERVER-1> connected to DUT through <DIface-0> belonging to network N0
  - DHCP Server <SERVER-2> connected to DUT through <DIface-0> belonging to network N0.

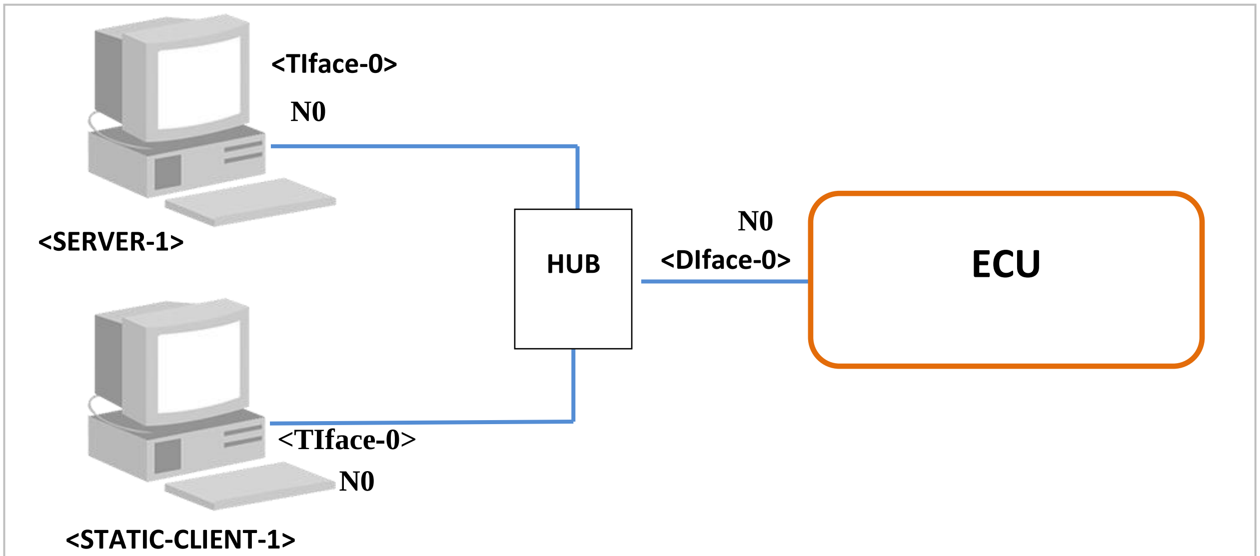
  These two servers are connected to the same DUT interface <DIface-0> through an emulated broadcast device (e.g., HUB. See below)



TOPOLOGY-3

- **TOPOLOGY-4**

  In this topology, following is being emulated:
  - DHCP Server <SERVER-1> connected to DUT through <DIface-0> belonging to network N0
  - Non DHCP Client (with Statically assigned IP Address). These two objects are connected to the same DUT interface <DIface-0> through an emulated broadcast device (e.g., HUB. See above)



TOPOLOGY-4

4.8.3 Required topology related configuration

This suite expects to be running against any IP enabled network interface which supports acquisitiong of IP address using DHCP. Most of the tests can be run against a single interface, only a few tests require that the TESTER be connected to 2 interfaces that are connected to the DUT.

The highest number of DUT interfaces required in the DHCP Client Test Suite is 2 (two).

The following information are obtained from the unused IP network configurations:

- IP Address of all the emulated servers
- IP Address Pool to be offered by these emulated DHCP Servers

4.8.4 Coverage

Section No.	Test Category	Test Number(s)
1.6, 2	Design Goals and Protocol Summary The Client-Server Protocol	DHCPv4_CLIENT_SUMMARY_01 to DHCPv4_CLIENT_SUMMARY_04
3	The Client-Server Protocol	DHCPv4_CLIENT_PROTOCOL_01 to DHCPv4_CLIENT_PROTOCOL_06
3.1	Client-server interaction - allocating a network address	DHCPv4_CLIENT_ALLOCATING_01 to DHCPv4_CLIENT_ALLOCATING_14
3.2	Client-server interaction - reusing a previously allocated network address	DHCPv4_CLIENT_REUSING_01 to DHCPv4_CLIENT_REUSING_08
3.4, 3.5	Client parameters in DHCP	DHCPv4_CLIENT_PARAMETERS_01 to DHCPv4_CLIENT_PARAMETERS_06
3.6, 3.7, 4.4.4	DHCP Usage	DHCPv4_CLIENT_USAGE_01 to DHCPv4_CLIENT_USAGE_03
4.1	Constructing and sending DHCP messages	DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_01 to DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_14
4.3.2	DHCPREQUEST message	DHCPv4_CLIENT_REQUEST_01 to DHCPv4_CLIENT_REQUEST_12
4.4.1	Initialization and allocation of network address	DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_01 to DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_10
4.4.3	Initialization with an externally assigned network address	DHCPv4_CLIENT_INITIALIZATION_EXTERNAL_01 to DHCPv4_CLIENT_INITIALIZATION_EXTERNAL_03
4.4.5	Reacquisition and expiration	DHCPv4_CLIENT_REACQUISITION_01 to DHCPv4_CLIENT_REACQUISITION_10
TOTAL		90
	(Negative)	10

4.8.5 Parameters and constants used in the tests

Parameters/Constants	Description
<ParamLeaseTime>	Value of DHCP IP Address Lease Time in seconds which will be offered to the DUT (DHCP Client) unless overridden by specific value in a particular test. It is advised that a too small value (less than 10 seconds) can create some critical timeout related issues possibly leading to undesirable test results since other timers like T1 (Renewal Time) and T2 (Rebinding Time) are based on this entries. [Default : 15 seconds]
<ParamToleranceTime>	The amount of tolerance to be provided so that we do not miss DUT packets. Increasing the value of this entry too much will mean adding too much tolerance to DUT behaviour which can mean more tests passing but actually with not sufficient conformance with the specification.[Default : 1 second]
<ParamProcessTime>	The amount of time TESTER will usually wait for the DUT to process some PDU sent by TESTER before continuing with the test. The purpose of this entry also is to provide some amount of latency consideration for the DUT so that the DUT gets ample time to process some PDU and take action in accordance to that. [Default : 2 seconds]
<ParamListenTime>	When we listen for a packet from the DUT either as response to a packet we have senht or when the specification does not define any specific time to listen for, this entry is used. It is also very important that it should have a large value since we start listening for a packet and in the background some automated script may be fired to trigger some event of the DUT which should cause the DUT to send our desirable packet. [Default: 10 seconds]
<ParamFirstRetransmissionInterval>	The number of seconds taken by the DUT to re-send a DHCPDISCOVER after TESTER acting as a DHCP-Server did not send a DHCPOFFER corresponding to the first DHCPDISCOVER sent by the DUT. This parameter is used to calculate the next retransmission interval by the TESTER and is different in case of different DUTs. "The delay between retransmissions SHOULD be chosen to allow sufficient time for replies from the server to be delivered based on the characteristics of the internetwork between the client and the server. For example, in a 10Mb/sec Ethernet internetwork, the delay before the first retransmission SHOULD be 4 seconds randomized by the value of a uniform random number chosen from the range -1 to +1." -RFC 2131,Sec 4.1, Pg 24. [Default: 4 seconds]
<DHCP-MAGIC-COOKIE>	The first 4 octets of DHCP Options which represent the value 99, 130, 83 and 99 respectively.[Ref: RFC2131 Section 3 Page 13]
<IP-BROADCAST-ADDRESS>	The all 1 IP broadcast address to which if any IP packet is sent, all the nodes within the physical LAN (and logical LAN is policy permits) will receive the packet. [Ref: RFC2131 Section 4.4.3 Page 39]
<SERVERn-IP-ADDRESS>	IP Address of the emulated 'n'th DHCP Server. (n = 1, 2)

SERVERn-IP-POOL-L-M>	'M'th IP Address in the pool of IP Address that the 'n'th server has to offer to clients requesting IP address through associated 'L'th interface connection between DUT and TESTER.
<SERVERn-IP-POOL-NETMASK>	IP Subnet Mask for all the pools of IP Addresses (as above) associated globally with the 'n'th emualated DHCP Server.
<MAC-UNUSED-ADDRESS>	A MAC Address (Layer 2 address) which is not of any of the interfaces of TESTER nor DUT.
<IP-UNUSED-ADDRESS>	IP Address which is of a different subnet than all the emulated servers and all other objects within the simulated topologies.
<IP-UNUSED-NET-MASK>	IP Subnet Mask of the above IP Unused Address which is basically the IP Netmask provided for the configuration entry 'IP Unused Net Mask'
<HIGH-LEASE-TIME>	In some tests which require the DUT to retransmit its DHCPREQUEST Message in Renewing state must have sufficiently high lease time so that the calculated value of T1 is such that half-way through T2 from T1 is greater than 60 seconds. [Value : 350 seconds] [Ref: RFC2131 Section 4.4.5 Page 41]
<VERY-HIGH-LEASE-TIME>	In some tests which require the DUT to retransmit its DHCPREQUEST Message in Rebinding state must have sufficiently high lease time so that the calculated value of T2 is such that half-way through Lease Time from T2 is greater than 60 seconds. [Value : 1000 seconds] [Ref: RFC2131 Section 4.4.5 Page 41]
<REMOTE-CLIENTn-LEASE-TIME>	The 'n'th remote DHCP Client (DUT) which is served by emulated DHCP Server if identified as <REMOTE-CLIENTn>, then the lease time for which the client is expected to use the offered IP Address is identified by this entry.
<REMOTE-CLIENTn-T1>	The 'n'th remote DHCP Client (DUT) which is served by emulated DHCP Server if identified as <REMOTE-CLIENTn>, then the renewal time after which client is expected to attempt automated renewal of IP address (as per RFC2131 Section 4.4 Page 35) is identified by this entry. [Expected value : 0.5 * Offered Lease Time]
<REMOTE-CLIENTn-T2>	The 'n'th remote DHCP Client (DUT) which is served by emulated DHCP Server if identified as <REMOTE-CLIENTn>, then the rebinding time after which client is expected to attempt broadcasting of DHCPREQUEST message after failing to get response in Renewing state (as per RFC2131 Section 4.4 Page 35) is identified by this entry. [Expected value : 0.875 * Offered Lease Time]

4.8.6 Tests

5.7.6.1 Summary

DHCPv4_CLIENT_SUMMARY_01: Setup Verification (DHCP Client Listens on UDP port 68)

Synopsis	A DHCP Client Listens on UDP port 68
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message
Test Iterations	
Notes	Derived from (MUST)

DHCPv4_CLIENT_SUMMARY_02: Discard DHCP Offsers whose xid is not the one of the latest DHCPDISCOVER sent

Synopsis	A DHCP client must be prepared to receive multiple responses to a request for configuration parameters (Note: If the 'xid' of an arriving DHCPOFFER message does not match the 'xid' of the most recent DHCPDISCOVER message, the DHCPOFFER message must be silently discarded.)
Prerequisites	Check section prerequisites
Test setup	Topology 3
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Extracts the content of 'xid' field to <extractedXID> 6. TESTER: DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - 'xid' field set to (extractedXID+1) 7. TESTER:DHCP Server <SERVER-2> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - 'xid' field set to extractedXID 8. TESTER:DHCP Server <SERVER-2> Listens (upto <ParamListenTime>) on <DIface-0> 9. DUT: Sends DHCPREQUEST Message 10. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 54 (Server Identifier Option) - Length field set to 4 - Value set to <SERVER2-IP-ADDRESS>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 9. DUT: Sends DHCPREQUEST Message 10. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 54 (Server Identifier Option) - Length field set to 4 - Value set to <SERVER2-IP-ADDRESS>

Test Iterations	
Notes	Derived from (MUST)

DHCPv4_CLIENT_SUMMARY_03: Receive DHCP messages with an 'options' field of at least length 312 octets

Synopsis	A DHCP client must be prepared to receive DHCP messages with an 'options' field of at least length 312 octets. This requirement implies that a DHCP client must be prepared to receive a message of up to 576 octets
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message
Test Iterations	
Notes	Derived from RFC 2131 Section 2 Page 10 'Protocol Summary' (MUST)

DHCPv4_CLIENT_SUMMARY_04: The flags field must be set to zero by clients

Synopsis	The remaining bits of the flags field are reserved for future use. They MUST be set to zero by clients and ignored by servers and relay agents
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Bits 2 to 16 of 'flags' field is set to 0
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Bits 2 to 16 of 'flags' field is set to 0
Test Iterations	
Notes	Derived from RFC 2131 Section 2 Page 11 'Protocol Summary' (MUST)

4.8.6.1 The Client-Server Protocol

DHCPv4_CLIENT_PROTOCOL_01: First four octets of the 'options' field of the DHCP message

Synopsis	The first four octets of the 'options' field of the DHCP message contain the (decimal) values 99, 130, 83 and 99, respectively (this is the same magic cookie as is defined in RFC 1497)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - First four octets of DHCP Options is set to <DHCP-MAGIC-COOKIE>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - First four octets of DHCP Options is set to <DHCP-MAGIC-COOKIE>
Test Iterations	
Notes	Derived from RFC 2131 Section 3 Page 13 'The Client-Server Protocol' (MUST)

DHCPv4_CLIENT_PROTOCOL_02: "DHCP message type" option present in DHCPDISCOVER Message.

Synopsis	One particular option - the \"DHCP message type\" option - must be included in every DHCP message (Note: This test verifies the above statement for DHCPDISCOVER Message)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 3 Page 13 'The Client-Server Protocol' (MUST)

DHCPv4_CLIENT_PROTOCOL_03: "DHCP message type" option present in DHCPREQUEST Message.

Synopsis	One particular option - the \"DHCP message type\" option - must be included in every DHCP message (Note: This test verifies the above statement for DHCPREQUEST Message)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_REQUESTING 4. DUT: Transit finite state to DHCPCLIENT_STATE_REQUESTING 5. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_REQUESTING 5. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 3 Page 13 'The Client-Server Protocol' (MUST)

DHCPv4_CLIENT_PROTOCOL_04: "DHCP message type" option present in DHCPRELEASE Message.

Synopsis	One particular option - the \"DHCP message type\" option - must be included in every DHCP message (Note: This test verifies the above statement for DHCPRELEASE Message)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to release <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPRELEASE Message 8. TESTER: Verify that received DHCPRELEASE Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPRELEASE Message 8. TESTER: Verify that received DHCPRELEASE Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 3 Page 13 'The Client-Server Protocol' (MUST)

DHCPv4_CLIENT_PROTOCOL_05: "DHCP message type" option present in DHCPDECLINE Message.

Synopsis	One particular option - the \"DHCP message type\" option - must be included in every DHCP message (Note: This test verifies the above statement for DHCPDECLINE Message)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0> 8. TESTER:DHCP Server <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing : - ARP Sender MAC Address field set to <MAC-UNUSED-ADDRESS> - ARP Sender IP Address field set to <SERVER1-IP-POOL-0-0>(*INVALID*) 9. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 10. DUT: Sends DHCPDECLINE Message 11. TESTER: Verify that received DHCPDECLINE Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0> 10. DUT: Sends DHCPDECLINE Message 11. TESTER: Verify that received DHCPDECLINE Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)

DHCPv4_CLIENT_PROTOCOL_06: "DHCP message type" option present in DHCPINFORM Message.

Synopsis	One particular option - the \"DHCP message type\" option - must be included in every DHCP message (Note: This test verifies the above statement for DHCPINFORM Message)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally assign IP Address <SERVER1-IP-POOL-0-0> with mask <SERVER1-IP-POOL-NETMASK> on DUT for interface <DIface-0> 3. TESTER: Externally cause DUT to send DHCPINFORM Message through <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Pass Criteria	5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Message Option containing: - Type field set to 53 (Message Type Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 3 Page 13 'The Client-Server Protocol' (MUST)

4.8.6.2 Client-server interaction - allocating a network address

DHCPv4_CLIENT_ALLOCATING_01: Broadcast DHCPDISCOVER message on its local physical subnet

Synopsis	The client broadcasts a DHCPDISCOVER message on its local physical subnet
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Destination IP Address field is set to <IP-BROADCAST-ADDRESS>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Destination IP Address field is set to <IP-BROADCAST-ADDRESS>
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 13 (MUST)

DHCPv4_CLIENT_ALLOCATING_02: Send DHCPDISCOVER (may suggest a value for lease time)

Synopsis	The DHCPDISCOVER message MAY include options that suggest values for the network address and lease duration (Note: This test checks that DUT may suggest a value for lease time)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally configure DHCP option 51 (IP Address Lease Time Option) with value 60 for interface <DIface-0> 3. TESTER: Externally cause DUT to bring up <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPDISCOVER Message 6. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to 60
Pass Criteria	5. DUT: Sends DHCPDISCOVER Message 6. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to 60
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 13 (MAY)

DHCPv4_CLIENT_ALLOCATING_03: Send DHCPREQUEST - must include the 'server identifier'

Synopsis	The client broadcasts a DHCPREQUEST message that MUST include the 'server identifier' option to indicate which server it has selected
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 54 (Server Identifier Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 54 (Server Identifier Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS>
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 16 (MUST)

DHCPv4_CLIENT_ALLOCATING_04: Send DHCPREQUEST - header value 'secs' field

Synopsis	the DHCPREQUEST message MUST use the same value in the DHCP message header's 'secs' field ... as the original DHCPDISCOVER message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Extracts the content of 'secs' field to <extractedSeconds> 6. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - 'secs' field is set to extractedSeconds
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - 'secs' field is set to extractedSeconds
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 16 (MUST)

DHCPv4_CLIENT_ALLOCATING_05: Send DHCPREQUEST to the same IP broadcast address

Synopsis	the DHCPREQUEST message MUST ... be sent to the same IP broadcast address as the original DHCPDISCOVER message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Destination IP Address field is set to <IP-BROADCAST-ADDRESS> 6. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <IP-BROADCAST-ADDRESS>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Destination IP Address field is set to <IP-BROADCAST-ADDRESS> 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <IP-BROADCAST-ADDRESS>
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 16 (MUST)

DHCPv4_CLIENT_ALLOCATING_06: Send DHCPDISCOVER message - timeout and resend on no DHCPOFFER messages

Synopsis	The client times out and retransmits the DHCPDISCOVER message if the client receives no DHCPOFFER messages
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPDISCOVER Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. DUT: Sends DHCPDISCOVER Message
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 16 (MUST)

DHCPv4_CLIENT_ALLOCATING_07: Send DHCPDECLINE Message and restart configuration process

Synopsis	If the client detects that the address is already in use (e.g., through the use of ARP), the client MUST send a DHCPDECLINE message to the server and restarts the configuration process (Note: In this test we check that the DUT does restart the configuration process after sending the DHCPDECLINE Message)
Prerequisites	Check section prerequisites
Test setup	Topology 4
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER:DHCP Client <STATIC-CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0> 8. TESTER:DHCP Client <STATIC-CLIENT-1> Sends ARP Response Message to DUT through <DIface-0> containing : - ARP Sender MAC Address field set to <MAC-UNUSED-ADDRESS> - ARP Target MAC Address field set to <DIface-0-MAC-ADDRESS> - ARP Sender IP Address field set to <SERVER1-IP-POOL-0-0> - ARP Target IP Address field set to <SERVER1-IP-POOL-0-0> 9. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 10. DUT: Sends DHCPDECLINE Message 11. TESTER:DHCP Server <SERVER-1> Listens (upto (10 + <ParamToleranceTime>) second) on <DIface-0> 12. DUT: Sends DHCPDISCOVER Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0> 10. DUT: Sends DHCPDECLINE Message 12. DUT: Sends DHCPDISCOVER Message

Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (MUST)

DHCPv4_CLIENT_ALLOCATING_08: Wait minimum 10 seconds before restarting configuration

Synopsis	The client SHOULD wait a minimum of ten seconds before restarting the configuration process to avoid excessive network traffic in case of looping
Prerequisites	Check section prerequisites
Test setup	Topology 4
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER:DHCP Client <STATIC-CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0> 8. TESTER:DHCP Client <STATIC-CLIENT-1> Sends ARP Response Message to DUT through <DIface-0> containing : - ARP Sender MAC Address field set to <MAC-UNUSED-ADDRESS> - ARP Target MAC Address field set to <DIface-0-MAC-ADDRESS> - ARP Sender IP Address field set to <SERVER1-IP-POOL-0-0> - ARP Target IP Address field set to <SERVER1-IP-POOL-0-0> 9. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 10. DUT: Sends DHCPDECLINE Message 11. TESTER:DHCP Server <SERVER-1> Listens (upto (10 + <ParamToleranceTime>) second) on <DIface-0> 12. DUT: Sends DHCPDISCOVER Message 13. TESTER: Verify that the time interval between reception of last DHCPDECLINE Message and reception of last DHCPDISCOVER Message is greater than (10 - <ParamToleranceTime>) second
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0> 10. DUT: Sends DHCPDECLINE Message

	12. DUT: Sends DHCPDISCOVER Message 13. TESTER: Verify that the time interval between reception of last DHCPDECLINE Message and reception of last DHCPDISCOVER Message is greater than (10 - <ParamToleranceTime>) second
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (SHOULD)

DHCPv4_CLIENT_ALLOCATING_09: Receive DHCPNAK - restart the configuration process

Synopsis	If the client receives a DHCPNAK message, the client restarts the configuration process
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPREQUEST Message 7. TESTER:DHCP Server <SERVER-1> Sends DHCPNAK Message to DUT through <DIface-0> 8. TESTER:DHCP Server <SERVER-1> Listens (upto (10 + <ParamToleranceTime>) second) on <DIface-0> 9. DUT: Sends DHCPDISCOVER Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 6. DUT: Sends DHCPREQUEST Message 9. DUT: Sends DHCPDISCOVER Message
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (MUST)

DHCPv4_CLIENT_ALLOCATING_10: Resend DHCPREQUEST message if timeout on no DHCPACK or a DHCPNAK message

Synopsis	The client times out and retransmits the DHCPREQUEST message if the client receives neither a DHCPACK or a DHCPNAK message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: Initialize server1 to offer IP Address Lease Time of <HIGH-LEASE-TIME> seconds to the DUT 4. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 5. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 6. TESTER:DHCP Server <SERVER-1> Listens (upto ((<REMOTE-CLIENT1-T2> - <REMOTE-CLIENT1-T1>) + <ParamToleranceTime>) second) on <DIface-0> 7. DUT: Sends 2 DHCPREQUEST Messages
Pass Criteria	5. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 7. DUT: Sends 2 DHCPREQUEST Messages
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (MUST)

DHCPv4_CLIENT_ALLOCATING_11: Notify the user that the initialization process has failed and is restarting

Synopsis	The client SHOULD notify the user that the initialization process has failed and is restarting
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER:DHCP Server <SERVER-1> Listens (upto (<REMOTE-CLIENT1-LEASE-TIME> + <ParamToleranceTime>) second) on <DIface-0> 6. DUT: Sends DHCPDISCOVER Message 7. TESTER: Verify that the DUT has correctly logged that initialization process has failed and is restarting for
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Sends DHCPDISCOVER Message 7. TESTER: Verify that the DUT has correctly logged that initialization process has failed and is restarting for
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (SHOULD)

DHCPv4_CLIENT_ALLOCATING_12: Send DHCPRELEASE message

Synopsis	The client may choose to relinquish its lease on a network address by sending a DHCPRELEASE message to the server
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to release <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPRELEASE Message 8. TESTER: Wait till <ParamProcessTime> for DUT to actually stop using the released address 9. TESTER:DHCP Server <SERVER-1> Sends ICMP Echo Message to DUT through <DIface-0> containing : - Destination Harware Address field set to <DIface-0-MAC-ADDRESS> - Destination IP Address field set to <SERVER1-IP-POOL-0-0> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Does not send ICMP Echo Reply Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPRELEASE Message 11. DUT: Does not send ICMP Echo Reply Message
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (MAY)

DHCPv4_CLIENT_ALLOCATING_13: Send DHCPRELEASE - 'client identifier' or 'chaddr' fields and network address

Synopsis	The client identifies the lease to be released with its 'client identifier', or 'chaddr' and network address in the DHCPRELEASE message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Extracts the content of 'chaddr' field to <extractedChaddr> 6. DUT CONFIGURE: Externally cause DUT to release <DIface-0> 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPRELEASE Message 9. TESTER: Verify that received DHCPRELEASE Message contains: - 'chaddr' field is set to extractedChaddr
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 8. DUT: Sends DHCPRELEASE Message 9. TESTER: Verify that received DHCPRELEASE Message contains: - 'chaddr' field is set to extractedChaddr
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (MUST)

DHCPv4_CLIENT_ALLOCATING_14: Consisent use of 'client identifier' field

Synopsis	If the client used a 'client identifier' when it obtained the lease, it MUST use the same 'client identifier' in the DHCPRELEASE message If the client supplies a 'client identifier', the client MUST use the same 'client identifier' in all subsequent messages
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally configure DHCP option 61 (Client-identifier Option) with value \"CONFORMANCE-DUT-0\" for interface <DIface-0> 3. TESTER: Externally cause DUT to bring up <DIface-0> 4. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_SELECTING 5. DUT: Transit finite state to DHCPCLIENT_STATE_SELECTING 6. TESTER: Extracts the content of 61 (Client-identifier Option) to <extractedClientID> 7. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 8. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 9. DUT CONFIGURE: Externally cause DUT to release <DIface-0> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends DHCPRELEASE Message 12. TESTER: Verify that received DHCPRELEASE Message contains: - Message Option containing extractedClientID
Pass Criteria	5. DUT: Transit finite state to DHCPCLIENT_STATE_SELECTING 8. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 11. DUT: Sends DHCPRELEASE Message 12. TESTER: Verify that received DHCPRELEASE Message contains: - Message Option containing extractedClientID
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (MUST)

4.8.6.3 Client-server interaction - reusing a previously allocated network address

DHCPv4_CLIENT_REUSING_01: Restarts initialization process retransmission algorithm timeout

Synopsis	If the client receives neither a DHCPACK or a DHCPNAK message after employing the retransmission algorithm, the client reverts to INIT state and restarts the initialization process
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: Initialize server1 to offer IP Address Lease Time of <HIGH-LEASE-TIME> seconds to the DUT 4. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 5. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. TESTER: Wait till <ParamProcessTime> for DUT to stabilize in BOUND state 7. DUT CONFIGURE: Externally cause DUT to renew <DIface-0> 8. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 9. DUT: Sends 2 DHCPREQUEST Messages 10. TESTER:DHCP Server <SERVER-1> Listens (upto (130 + 10 + <ParamToleranceTime>) second) on <DIface-0> 11. DUT: Sends DHCPDISCOVER Message 12. (Note: We have derived 130 seconds keeping in mind the total retransmission time in RFC 2131 Section 4.1 Page 24)
Pass Criteria	5. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 9. DUT: Sends 2 DHCPREQUEST Messages 11. DUT: Sends DHCPDISCOVER Message
Test Iterations	
Notes	Derived from RFC 2131 Section 3.1 Page 17 (MUST)

DHCPv4_CLIENT_REUSING_02: Reusing a previously allocated network address

Synopsis	If a client remembers and wishes to reuse a previously allocated network address, a client may choose to omit some of the steps described in the previous section
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till <ParamProcessTime> for DUT to stabilize in BOUND state 6. DUT CONFIGURE: Externally cause DUT to renew <DIface-0> 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPREQUEST Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 8. DUT: Sends DHCPREQUEST Message
Test Iterations	
Notes	Derived from RFC 2131 Section 3.2 Page 17 (MAY)

DHCPv4_CLIENT_REUSING_03: Broadcast a DHCPREQUEST if reusing a previously allocated network address

Synopsis	If a client remembers and wishes to reuse a previously allocated network address, ...The client broadcasts a DHCPREQUEST message on its local subnet
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to renew <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS>
Test Iterations	
Notes	Derived from RFC 2131 Section 3.2 Page 17-18 (MUST)

DHCPv4_CLIENT_REUSING_04: Reusing a previously allocated network address - include 'requested IP address' option

Synopsis	If a client remembers and wishes to reuse a previously allocated network address, ...The message includes the client's network address in the 'requested IP address' option
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to renew <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 50 (Requested IP Address Option) - Length field set to 4 - Value set to <SERVER1-IP-POOL-0-0>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 50 (Requested IP Address Option) - Length field set to 4 - Value set to <SERVER1-IP-POOL-0-0>
Test Iterations	
Notes	Derived from RFC 2131 Section 3.2 Page 18 (MUST)

DHCPv4_CLIENT_REUSING_05: If not received an adress do not fill in the 'ciaddr' field

Synopsis	As the client has not received its network address, it MUST NOT fill in the 'ciaddr' field
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to renew <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to 0
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to 0
Test Iterations	
Notes	Derived from RFC 2131 Section 3.2 Page 18 (MUST)

DHCPv4_CLIENT_REUSING_06: Use same 'client identifier' in all subsequent messages DHCPREQUEST message

Synopsis	If the client used a 'client identifier' to obtain its address, the client MUST use the same 'client identifier' in the If the client supplies a 'client identifier', the client MUST use the same 'client identifier' in all subsequent messages DHCPREQUEST message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally configure DHCP option 61 (Client-identifier Option) with value \"CONFORMANCE-DUT-0\" for interface <DIface-0> 3. TESTER: Externally cause DUT to bring up <DIface-0> 4. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 5. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. TESTER: Extracts the content of 61 (Client-identifier Option) to <extractedClientID> 7. DUT CONFIGURE: Externally cause DUT to renew <DIface-0> 8. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 9. DUT: Sends DHCPREQUEST Message 10. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing extractedClientID
Pass Criteria	5. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 9. DUT: Sends DHCPREQUEST Message 10. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing extractedClientID
Test Iterations	
Notes	Derived from RFC 2131 Section 3.2 Page 18 (MUST)

DHCPv4_CLIENT_REUSING_07: Respond to ICMP Echo Request messages when reusing address

Synopsis	the client may respond to ICMP Echo Request messages at this point
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to renew <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS> 9. TESTER:DHCP Server <SERVER-1> Sends ICMP Echo Message to DUT through <DIface-0> containing : - Destination Harware Address field set to <DIface-0-MAC-ADDRESS> - Destination IP Address field set to <SERVER1-IP-POOL-0-0> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends ICMP Echo Reply Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS> 11. DUT: Sends ICMP Echo Reply Message
Test Iterations	
Notes	Derived from RFC 2131 Section 3.2 Page 18 (MAY)

DHCPv4_CLIENT_REUSING_08: On DHCPNAK fo not reuse network address and request new using non abbreviated process

Synopsis	If the client receives a DHCPNAK message, it cannot reuse its remembered network address. It must instead request a new address by restarting the configuration process, this time using the (non-abbreviated) procedure described in section 3.1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to renew <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER:DHCP Server <SERVER-1> Sends DHCPNAK Message to DUT through <DIface-0> 9. TESTER:DHCP Server <SERVER-1> Listens (upto (10 + <ParamToleranceTime>) second) on <DIface-0> 10. DUT: Sends DHCPDISCOVER Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 10. DUT: Sends DHCPDISCOVER Message
Test Iterations	
Notes	Derived from RFC 2131 Section 3.2 Page 19 (MUST)

4.8.6.4 Client parameters in DHCP

DHCPv4_CLIENT_PARAMETERS_01: Interpreration of time values

Synopsis	If a client has obtained a network address through some other means (e.g., manual configuration), it may use a DHCPINFORM request message to obtain other local configuration parameters
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally assign IP Address <IP-UNUSED-ADDRESS> with mask <IP-UNUSED-NET-MASK> on DUT for interface <DIface-0> 3. TESTER: Externally cause DUT to bring up <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPINFORM Message
Pass Criteria	5. DUT: Sends DHCPINFORM Message
Test Iterations	
Notes	Derived from RFC 2131 Section 3.4 Page 20 (MAY)

DHCPv4_CLIENT_PARAMETERS_02: Default value for TTL

Synopsis	First, most of the parameters have defaults defined in the Host Requirements RFCs; if the client receives no parameters from the server that override the defaults, a client uses those default values (Note: This test checks that if default value is not overridden, DUT uses the default value for Time To Live as per RFC1770 'IANA Numbering')
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till <ParamProcessTime> for DUT to perform necessary verification of offered IP address 6. TESTER:DHCP Server <SERVER-1> Sends ICMP Echo Message to DUT through <DIface-0> 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends ICMP Echo Reply Message 9. TESTER: Verify that received ICMP Echo Reply Message contains: - Time to Live field is set to 64
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 8. DUT: Sends ICMP Echo Reply Message 9. TESTER: Verify that received ICMP Echo Reply Message contains: - Time to Live field is set to 64
Test Iterations	
Notes	Derived from RFC 2131 Section 3.5 Page 21 'Client parameters in DHCP' (MUST)

DHCPv4_CLIENT_PARAMETERS_03: Override value for TTL

Synopsis	First, most of the parameters have defaults defined in the Host Requirements RFCs; if the client receives no parameters from the server that override the defaults, a client uses those default values (Note: This test checks that if default value is overridden, DUT uses the overridden value for Time To Live)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to <ParamLeaseTime> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to <ParamLeaseTime>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to <ParamLeaseTime>
Notes	Derived from RFC 2131 Section 3.5 Page 21 'Client parameters in DHCP' (MAY)

DHCPv4_CLIENT_PARAMETERS_04: Use same parameters in DHCPREQUEST message as in DHCPDISCOVER

Synopsis	If the client includes a list of parameters in a DHCPDISCOVER message, it MUST include that list in any subsequent DHCPREQUEST messages.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Extracts the content of 55 (Parameter Request List Option) to <extractedParamReq> 6. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing extractedParamReq
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing extractedParamReq
Test Iterations	
Notes	Derived from RFC 2131 Section 3.5 Page 21 'Client parameters in DHCP' (MUST)

DHCPv4_CLIENT_PARAMETERS_05: Set 'maximum DHCP message size' option

Synopsis	The client SHOULD include the 'maximum DHCP message size' option to let the server know how large the server may make its DHCP messages
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally configure DHCP option 57 (Maximum DHCP Size Option) with value 600 for interface <DIface-0> 3. TESTER: Externally cause DUT to bring up <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPDISCOVER Message 6. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - Type field set to 57 (Maximum DHCP Size Option)
Pass Criteria	5. DUT: Sends DHCPDISCOVER Message 6. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - Type field set to 57 (Maximum DHCP Size Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 3.5 Page 21 'Client parameters in DHCP' (SHOULD)

DHCPv4_CLIENT_PARAMETERS_06: Send DHCPDISCOVER message - include the 'IP address lease time' option

Synopsis	In addition, the client may suggest values for the network address and lease time in the DHCPDISCOVER message. The client may ... include the 'IP address lease time' option to suggest the lease time it would like
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally configure DHCP option 51 (IP Address Lease Time Option) with value 60 for interface <DIface-0> 3. TESTER: Externally cause DUT to bring up <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPDISCOVER Message 6. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to 60
Pass Criteria	5. DUT: Sends DHCPDISCOVER Message 6. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to 60
Test Iterations	
Notes	Derived from RFC 2131 Section 3.5 Page 21 'Client parameters in DHCP' (MAY)

4.8.6.5 DHCP usage

DHCPv4_CLIENT_USAGE_01: Use of DHCP in clients with multiple interfaces

Synopsis	A client with multiple network interfaces must use DHCP through each interface independently to obtain configuration information parameters for those separate interfaces
Prerequisites	Check section prerequisites
Test setup	Topology 2
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. DUT CONFIGURE: Externally configure DHCP Client on <DIface-1> 3. TESTER: Externally cause DUT to bring up <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPDISCOVER Message 6. TESTER: Extracts the content of 'chaddr' field to <extractedChaddr> 7. TESTER: Externally cause DUT to bring up <DIface-1> 8. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-1> 9. DUT: Sends DHCPDISCOVER Message 10. TESTER: Verify that received DHCPDISCOVER Message contains: - 'chaddr' field is not set to extractedChaddr
Pass Criteria	5. DUT: Sends DHCPDISCOVER Message 9. DUT: Sends DHCPDISCOVER Message 10. TESTER: Verify that received DHCPDISCOVER Message contains: - 'chaddr' field is not set to extractedChaddr
Test Iterations	
Notes	Derived from RFC 2131 Section 3.6 Page 22 (MUST)

DHCPv4_CLIENT_USAGE_02: Use DHCP at system boot time

Synopsis	A client SHOULD use DHCP to reacquire or verify its IP address and network parameters whenever the local network parameters may have changed; e.g., at system boot time
Prerequisites	Check section prerequisites
Test setup	Topology 3
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to release <DIface-0> 6. TESTER: Externally cause DUT to bring up <DIface-0> 7. TESTER: DHCP Server <SERVER-2> Cause DUT to transit its state to DHCPCLIENT_STATE_REQUESTING 8. DUT: Transit finite state to DHCPCLIENT_STATE_REQUESTING 9. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 50 (Requested IP Address Option) - Length field is not set to 4 - Value is not set to <SERVER1-IP-POOL-0-0>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 8. DUT: Transit finite state to DHCPCLIENT_STATE_REQUESTING 9. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 50 (Requested IP Address Option) - Length field is not set to 4 - Value is not set to <SERVER1-IP-POOL-0-0>
Test Iterations	
Notes	Derived from RFC 2131 Section 3.7 Page 22 'When clients should use DHCP' (SHOULD)

DHCPv4_CLIENT_USAGE_03: Revert from unicast to multicast on timeout

Synopsis	If the client receives no response to DHCP messages sent to the IP address of a known DHCP server, the DHCP client reverts to using the IP broadcast address (Note: In this test we check this feature for DHCPINFORM Message)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally assign IP Address <SERVER1-IP-POOL-0-0> with mask <SERVER1-IP-POOL-NETMASK> on DUT for interface <DIface-0> 3. TESTER: Externally cause DUT to send DHCPINFORM Message through <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS> 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPINFORM Message 9. TESTER: Verify that received DHCPINFORM Message contains: - Destination IP Address field is set to 0xffffffff
Pass Criteria	5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS> 8. DUT: Sends DHCPINFORM Message 9. TESTER: Verify that received DHCPINFORM Message contains: - Destination IP Address field is set to 0xffffffff
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.4 Page 40 'Use of broadcast and unicast' (MUST)

4.8.6.6 Constructing and sending DHCP messages

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_01: The last option must always be the 'end' option

Synopsis	The last option must always be the 'end' option
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - 255 (End Option) at position Last
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Message Option containing: - 255 (End Option) at position Last
Test Iterations	
Notes	Derived from RFC 2131 Section 4.1 Page 22-23 'Constructing and sending DHCP messages' (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_02: Use the IP address provided in the 'server identifier' option for any unicast requests

Synopsis	DHCP clients MUST use the IP address provided in the 'server identifier' option for any unicast requests to the DHCP server
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.1 Page 23 'Constructing and sending DHCP messages' (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_03: Source IP address field of DHCPDISCOVER Message is 0

Synopsis	DHCP messages broadcast by a client prior to that client obtaining its IP address must have the source address field in the IP header set to 0 (Note: This test verifies source IP address field of DHCPDISCOVER Message)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Source IP Address field is set to 0
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - Source IP Address field is set to 0
Test Iterations	
Notes	Derived from RFC 2131 Section 4.1 Page 23 'Constructing and sending DHCP messages' (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_04: Source IP address field of DHCPREQUEST Message is 0

Synopsis	DHCP messages broadcast by a client prior to that client obtaining its IP address must have the source address field in the IP header set to 0 (Note: This test verifies source IP address field of DHCPREQUEST Message)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_REQUESTING 4. DUT: Transit finite state to DHCPCLIENT_STATE_REQUESTING 5. TESTER: Verify that received DHCPREQUEST Message contains: - Source IP Address field is set to 0
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_REQUESTING 5. TESTER: Verify that received DHCPREQUEST Message contains: - Source IP Address field is set to 0
Test Iterations	
Notes	Derived from RFC 2131 Section 4.1 Page 23 'Constructing and sending DHCP messages' (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_05: Parse 'sname' field when Option Overload is present

Synopsis	If the options in a DHCP message extend into the 'sname' and 'file' fields, the 'option overload' option MUST appear in the 'options' field, with value 1, 2 or 3 (Note: Here we verify that DUT correctly parses 'sname' field when Option Overload is present and set to 'sname contains options')
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing:

	- Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 9. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends ICMP Echo Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ICMP Echo Message
Test Iterations	
Notes	Derived from RFC 2131 Section 4.1 Page 23 'Constructing and sending DHCP messages' (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_06: Parse 'file' field when Option Overload is present

Synopsis	If the options in a DHCP message extend into the 'sname' and 'file' fields, the 'option overload' option MUST appear in the 'options' field, with value 1, 2 or 3 (Note: Here we verify that DUT correctly parses 'file' field when Option Overload is present and set to 'file contains options')
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option) 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing:

	- Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option) 9. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends ICMP Echo Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ICMP Echo Message
Test Iterations	
Notes	Derived from RFC 2131 Section 4.1 Page 23 'Constructing and sending DHCP messages' (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_07: Do not parse 'file' field as Options when Option Overload is not present)

Synopsis	If the options in a DHCP message extend into the 'sname' and 'file' fields, the 'option overload' option MUST appear in the 'options' field, with value 1, 2 or 3 (Note: Here we verify that DUT correctly does not parse 'file' field as Options when Option Overload is not present)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing :

	- Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option) 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option) 9. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends ICMP Echo Message 12. TESTER: Externally reset all client configurations for all configured interfaces on the DUT 13. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 14. TESTER: Externally cause DUT to bring up <DIface-0> 15. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 16. DUT: Sends DHCPDISCOVER Message 17. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option)(*INVALID*) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing:

	- Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option) 18. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 19. DUT: Sends DHCPREQUEST Message 20. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option)(*INVALID*) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option) 21. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 22. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 23. DUT: Does not send ICMP Echo Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ICMP Echo Message 16. DUT: Sends DHCPDISCOVER Message 19. DUT: Sends DHCPREQUEST Message 23. DUT: Does not send ICMP Echo Message
Test Iterations	
Notes	Derived from (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_08: Check 'sname' field when 1st octet is not the start of the option

Synopsis	The options in the 'sname' and 'file' fields (if in use as indicated by the 'options overload' option) MUST begin with the first octet of the field (Note: In this test we verify that the DUT correctly does not imbibe the Option present in 'sname' field when 1st octet is not the start of the option)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4

	- Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 9. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends ICMP Echo Message 12. TESTER: Externally reset all client configurations for all configured interfaces on the DUT 13. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 14. TESTER: Externally cause DUT to bring up <DIface-0> 15. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 16. DUT: Sends DHCPDISCOVER Message 17. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 0 (Pad Option)(*INVALID*) - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 18. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 19. DUT: Sends DHCPREQUEST Message 20. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 0 (Pad Option)(*INVALID*) - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 21. TESTER: Externally cause DUT to send ICMP Echo Request

	through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 22. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 23. DUT: Does not send ICMP Echo Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ICMP Echo Message 16. DUT: Sends DHCPDISCOVER Message 19. DUT: Sends DHCPREQUEST Message 23. DUT: Does not send ICMP Echo Message
Test Iterations	
Notes	Derived from (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_09: Check 'sname' field in absence of 'end' option

Synopsis	The options in the 'sname' and 'file' fields (if in use as indicated ... MUST be terminated by an 'end' option (Note: In this test we verify that the DUT correctly does not imbibe the Option present in 'sname' field in absence of 'end' option)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4

	- Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 9. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends ICMP Echo Message 12. TESTER: Externally reset all client configurations for all configured interfaces on the DUT 13. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 14. TESTER: Externally cause DUT to bring up <DIface-0> 15. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 16. DUT: Sends DHCPDISCOVER Message 17. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option)(*INVALID*) 18. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 19. DUT: Sends DHCPREQUEST Message 20. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option)(*INVALID*) 21. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 22. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on

	<DIface-0> 23. DUT: Does not send ICMP Echo Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ICMP Echo Message 16. DUT: Sends DHCPDISCOVER Message 19. DUT: Sends DHCPREQUEST Message 23. DUT: Does not send ICMP Echo Message
Test Iterations	
Notes	Derived from (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_10: Check 'file' field in absence of 'end' option

Synopsis	The options in the 'sname' and 'file' fields (if in use as indicated ... MUST be terminated by an 'end' option (Note: In this test we verify that the DUT correctly does not imbibe the Option present in 'file' field in absence of 'end' option)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option) 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing: - Type field set to 3 (Router Option) - Length field set to 4

	- Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option) 9. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends ICMP Echo Message 12. TESTER: Externally reset all client configurations for all configured interfaces on the DUT 13. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 14. TESTER: Externally cause DUT to bring up <DIface-0> 15. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 16. DUT: Sends DHCPDISCOVER Message 17. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option)(*INVALID*) 18. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 19. DUT: Sends DHCPREQUEST Message 20. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Additional PAD Options to make packet length set to 576 - Message Option containing: - Type field set to 52 (Overload Option) - Length field set to 1 - Value set to the 'file' field is used to hold options - DHCP Message Option in 'file' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'file' field containing: - Type field set to 255 (End Option)(*INVALID*) 21. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 22. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on

	<DIface-0> 23. DUT: Does not send ICMP Echo Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ICMP Echo Message 16. DUT: Sends DHCPDISCOVER Message 19. DUT: Sends DHCPREQUEST Message 23. DUT: Does not send ICMP Echo Message
Test Iterations	
Notes	Derived from (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_11: The 'sname' field and 'pad' options

Synopsis	The options in the 'sname' and 'file' fields (if in use as indicated ...) and MUST be followed by 'pad' options to fill the remainder of the field (Note: This test is done using the 'sname' field only)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Message Option containing: - Type field set to 52 (Overload Option) - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Message Option containing: - Type field set to 52 (Overload Option) - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) 9. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS>

	10. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 11. DUT: Sends ICMP Echo Message 12. TESTER: Externally reset all client configurations for all configured interfaces on the DUT 13. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 14. TESTER: Externally cause DUT to bring up <DIface-0> 15. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 16. DUT: Sends DHCPDISCOVER Message 17. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Message Option containing: - Type field set to 52 (Overload Option) - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) - DHCP Message Option in 'sname' field containing: - Type field set to 0 (Pad Option)(*INVALID*) - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option)(*INVALID*) 18. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 19. DUT: Sends DHCPREQUEST Message 20. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Message Option containing: - Type field set to 52 (Overload Option) - Value set to the 'sname' field is used to hold options - DHCP Message Option in 'sname' field containing: - Type field set to 3 (Router Option) - Length field set to 4 - Value set to <SERVER1-IP-ADDRESS> - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option) - DHCP Message Option in 'sname' field containing: - Type field set to 0 (Pad Option)(*INVALID*) - DHCP Message Option in 'sname' field containing: - Type field set to 255 (End Option)(*INVALID*) 21. TESTER: Externally cause DUT to send ICMP Echo Request through <DIface-0> to IP address <IP-UNUSED-ADDRESS> 22. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0>

	23. DUT: Does not send ICMP Echo Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 11. DUT: Sends ICMP Echo Message 16. DUT: Sends DHCPDISCOVER Message 19. DUT: Sends DHCPREQUEST Message 23. DUT: Does not send ICMP Echo Message
Test Iterations	
Notes	Derived from (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_12: The retransmission delay should be doubled with subsequent retransmissions

Synopsis	The retransmission delay SHOULD be doubled with subsequent retransmissions up to a maximum of 64 seconds
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto (64 + 32 + 16 + 8 + 4 + 2 + 1 + <ParamToleranceTime>) second) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message(s) 5. TESTER: Verify that the time interval between reception of last two DHCPDISCOVER Messages is less than (64 + <ParamToleranceTime>) second
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message(s) 5. TESTER: Verify that the time interval between reception of last two DHCPDISCOVER Messages is less than (64 + <ParamToleranceTime>) second
Test Iterations	
Notes	Derived from RFC 2131 Section 4.1 Page 24 'Constructing and sending DHCP messages' (MUST)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_13: Retransmission strategy using a randomized exponential backoff algorithm

Synopsis	The client MUST adopt a retransmission strategy that incorporates a randomized exponential backoff algorithm to determine the delay between retransmissions
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Listen (for upto (3 * (<ParamFirstRetransmissionInterval> + <ParamToleranceTime>)) seconds) on <DIface-0> 4. DUT: Send multiple DHCPDISCOVER Messages 5. TESTER: Verify that the time interval between reception of 3rd last DHCPDISCOVER Message and second last DHCPDISCOVER Message is within the range of (<ParamFirstRetransmissionInterval> - 1) to (<ParamFirstRetransmissionInterval> + 1seconds 6. TESTER: Verify that the time interval between reception of last two DHCPDISCOVER Messages is within the range of ((2*<ParamFirstRetransmissionInterval>) - 1) to ((2*<ParamFirstRetransmissionInterval>) + 1seconds
Pass Criteria	4. DUT: Send multiple DHCPDISCOVER Messages 5. TESTER: Verify that the time interval between reception of 3rd last DHCPDISCOVER Message and second last DHCPDISCOVER Message is within the range of (<ParamFirstRetransmissionInterval> - 1) to (<ParamFirstRetransmissionInterval> + 1seconds 6. TESTER: Verify that the time interval between reception of last two DHCPDISCOVER Messages is within the range of ((2*<ParamFirstRetransmissionInterval>) - 1) to ((2*<ParamFirstRetransmissionInterval>) + 1seconds

Test Iterations	
Notes	Derived from RFC 2131 Section 4.1 Page 24 'Constructing and sending DHCP messages' (SHOULD)

DHCPv4_CLIENT_CONSTRUCTING_MESSAGES_14: A client may choose a different random initial 'xid' after reboot

Synopsis	For example, a client may choose a different, random initial 'xid' each time the client is rebooted,
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Extracts the content of 'xid' field to <extractedXID> 6. DUT CONFIGURE: Externally cause DUT to release <DIface-0> 7. TESTER: Externally cause DUT to bring up <DIface-0> 8. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 9. DUT: Sends DHCPDISCOVER Message 10. TESTER: Verify that received DHCPDISCOVER Message contains: - 'xid' field is not set to extractedXID
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 9. DUT: Sends DHCPDISCOVER Message 10. TESTER: Verify that received DHCPDISCOVER Message contains: - 'xid' field is not set to extractedXID
Notes	Derived from RFC 2131 Section 4.1 Page 24 'Constructing and sending DHCP messages' (MAY)

4.8.6.7 DHCPREQUEST message

DHCPv4_CLIENT_REQUEST_01: DHCPREQUEST message - the 'ciaddr' option

Synopsis	Client inserts ... , 'ciaddr' MUST be zero
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_REQUESTING 4. DUT: Transit finite state to DHCPCLIENT_STATE_REQUESTING 5. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to 0
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_REQUESTING 5. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to 0
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 31 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_02: DHCPREQUEST message - requested IP address

Synopsis	requested IP address' MUST be filled in with the yiaddr value from the chosen DHCPOFFER.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - 'yiaddr' field set to <SERVER1-IP-POOL-0-1> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 50 (Requested IP Address Option) - Length field set to 4 - Value set to <SERVER1-IP-POOL-0-1>
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 50 (Requested IP Address Option) - Length field set to 4 - Value set to <SERVER1-IP-POOL-0-1>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 31 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_03: DHCPREQUEST generated during INIT-REBOOT state: 'server identifier' option

Synopsis	o DHCPREQUEST generated during INIT-REBOOT state: 'server identifier' MUST NOT be filled in
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to reboot <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 54 (Server Identifier Option)
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 54 (Server Identifier Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 31 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_04: DHCPREQUEST generated during INIT-REBOOT state: 'requested IP address' option

Synopsis	o DHCPREQUEST generated during INIT-REBOOT state: 'requested IP address' option MUST be filled in with client's notion of its previously assigned address.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to reboot <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 50 (Requested IP Address Option) - Length field set to 4 - Value set to <SERVER1-IP-POOL-0-0>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field set to 50 (Requested IP Address Option) - Length field set to 4 - Value set to <SERVER1-IP-POOL-0-0>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 31 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_05: DHCPREQUEST generated during INIT-REBOOT state: 'ciaddr' option

Synopsis	o DHCPREQUEST generated during INIT-REBOOT state: 'ciaddr' MUST be zero
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. DUT CONFIGURE: Externally cause DUT to reboot <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to 0
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to 0
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 31 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_06: DHCPREQUEST generated during RENEWING state: 'server identifier' option

Synopsis	o DHCPREQUEST generated during RENEWING state: 'server identifier' MUST NOT be filled in
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 54 (Server Identifier Option)
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 54 (Server Identifier Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 32 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_07: DHCPREQUEST generated during RENEWING state: 'requested IP address' option

Synopsis	o DHCPREQUEST generated during RENEWING state: 'requested IP address' option MUST NOT be filled in
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 50 (Requested IP Address Option)
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 50 (Requested IP Address Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 32 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_08: DHCPREQUEST generated during RENEWING state: 'ciaddr' option

Synopsis	o DHCPREQUEST generated during RENEWING state: ciaddr' MUST be filled in with client's IP address
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 6. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to <SERVER1-IP-POOL-0-0>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to <SERVER1-IP-POOL-0-0>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 32 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_09: DHCPREQUEST generated during REBINDING state: 'server identifier' option

Synopsis	o DHCPREQUEST generated during REBINDING state: 'server identifier' MUST NOT be filled in
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_REBINDING 4. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 54 (Server Identifier Option)
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 54 (Server Identifier Option)
Test Iterations	
Notes	(MUST)

DHCPv4_CLIENT_REQUEST_10: DHCPREQUEST generated during REBINDING state: 'requested IP address' option

Synopsis	o DHCPREQUEST generated during REBINDING state: 'requested IP address' option MUST NOT be filled in
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_REBINDING 4. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 50 (Requested IP Address Option)
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Message Option containing: - Type field is not set to 50 (Requested IP Address Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 32 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_11: DHCPREQUEST generated during REBINDING state: 'ciaddr' option

Synopsis	o DHCPREQUEST generated during REBINDING state: 'ciaddr' MUST be filled in with client's IP address
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_REBINDING 6. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to <SERVER1-IP-POOL-0-0>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - 'ciaddr' field is set to <SERVER1-IP-POOL-0-0>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 32 'DHCPREQUEST message' (MUST)

DHCPv4_CLIENT_REQUEST_12: DHCPREQUEST generated during REBINDING state: use IP broadcast address

Synopsis	o DHCPREQUEST generated during REBINDING state: This message MUST be broadcast to the 0xffffffff IP broadcast address
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_REBINDING 6. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to 0xffffffff
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to 0xffffffff
Test Iterations	
Notes	Derived from RFC 2131 Section 4.3.2 Page 32 'DHCPREQUEST message' (MUST)

4.8.6.8 Initialization and allocation of network address

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_01: Random time between to desynchronize the use of DHCP at startup

Synopsis	The client SHOULD wait a random time between one and ten seconds to desynchronize the use of DHCP at startup
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPREQUEST Message 7. TESTER:DHCP Server <SERVER-1> Sends DHCPNAK Message to DUT through <DIface-0> 8. TESTER:DHCP Server <SERVER-1> Listens (upto (10 + <ParamToleranceTime>) second) on <DIface-0> 9. DUT: Sends DHCPDISCOVER Message 10. TESTER: Verify that the time interval between sending of last DHCPNAK Message and reception of last DHCPDISCOVER Message is greater than or equal to (1 - <ParamToleranceTime>) second 11. TESTER: Verify that the time interval between sending of last DHCPNAK Message and reception of last DHCPDISCOVER Message is less than or equal to (10 + <ParamToleranceTime>) second
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 6. DUT: Sends DHCPREQUEST Message 9. DUT: Sends DHCPDISCOVER Message 10. TESTER: Verify that the time interval between sending of last DHCPNAK Message and reception of last

	DHCPDISCOVER Message is greater than or equal to (1 - <ParamToleranceTime>) second 11. TESTER: Verify that the time interval between sending of last DHCPNAK Message and reception of last DHCPDISCOVER Message is less than or equal to (10 + <ParamToleranceTime>) second
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 36 (SHOULD)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_02: INIT state and DHCPDISCOVER messages

Synopsis	The client begins in INIT state and forms a DHCPDISCOVER message ... The client sets 'ciaddr' to 0x00000000
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - 'ciaddr' field is set to 0
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Verify that received DHCPDISCOVER Message contains: - 'ciaddr' field is set to 0
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 36 (MUST)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_03: INIT state and forms a DHCPDISCOVER message - the 'chaddr' field

Synopsis	The client begins in INIT state and forms a DHCPDISCOVER message ... The client MUST include its hardware address in the 'chaddr' field
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Extracts the content of Source Harware Address field to <extractedSrcHwAddr> 6. TESTER: Verify that received DHCPDISCOVER Message contains: - 'chaddr' field is set to extractedSrcHwAddr
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 6. TESTER: Verify that received DHCPDISCOVER Message contains: - 'chaddr' field is set to extractedSrcHwAddr
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 36 (MUST)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_04: Verify 'xid' of an arriving DHCPOFFER message

Synopsis	If the 'xid' of an arriving DHCPOFFER message does not match the 'xid' of the most recent DHCPDISCOVER message, the DHCPOFFER message must be silently discarded
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Extracts the content of 'xid' field to <extractedXID> 6. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - 'xid' field set to (extractedXID+1) 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Does not send DHCPREQUEST Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 8. DUT: Does not send DHCPREQUEST Message
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 36 (MUST)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_05: During Initialization discard arriving DHCPACK messages

Synopsis	Any arriving DHCPACK messages must be silently discarded
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Does not send DHCPREQUEST Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Does not send DHCPREQUEST Message
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 36 (MUST)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_06: The DHCPREQUEST message contains the same 'xid' as the DHCPOFFER message

Synopsis	The DHCPREQUEST message contains the same 'xid' as the DHCPOFFER message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER: Extracts the content of 'xid' field to <extractedXID> 6. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - 'xid' field set to extractedXID 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - 'xid' field is set to extractedXID
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 8. DUT: Sends DHCPREQUEST Message 9. TESTER: Verify that received DHCPREQUEST Message contains: - 'xid' field is set to extractedXID
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 38 (MUST)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_07: Lease expiration time value

Synopsis	The client records the lease expiration time as the sum of the time at which the original request was sent and the duration of the lease from the DHCPACK message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends DHCPDISCOVER Message 5. TESTER:DHCP Server <SERVER-1> Sends DHCPOFFER Message to DUT through <DIface-0> containing : - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to <ParamLeaseTime> 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Wait till (<ParamLeaseTime>/3) to keep DUT listening for DHCPACK from TESTER 9. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Message Option containing: - Type field set to 51 (IP Address Lease Time Option) - Length field set to 4 - Value set to <ParamLeaseTime> 10. TESTER: Wait till (<ParamLeaseTime>+(<ParamLeaseTime>/4)) for DUT to transit past RENEWING and REBINDING state but not past lease expiration 11. TESTER:DHCP Server <SERVER-1> Sends ICMP Echo Message to DUT through <DIface-0> containing : - Destination Harware Address field set to <DIface-0-MAC-ADDRESS> - Destination IP Address field set to <SERVER1-IP-POOL-0-0> 12. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 13. DUT: Sends ICMP Echo Reply Message 14. TESTER: Wait till ((<ParamLeaseTime>/4)+<ParamToleranceTime>) for DUT to transit past lease expiration

	15. TESTER:DHCP Server <SERVER-1> Sends ICMP Echo Message to DUT through <DIface-0> containing : - Destination Harware Address field set to <DIface-0-MAC-ADDRESS> - Destination IP Address field set to <SERVER1-IP-POOL-0-0> 16. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 17. DUT: Does not send ICMP Echo Reply Message
Pass Criteria	4. DUT: Sends DHCPDISCOVER Message 7. DUT: Sends DHCPREQUEST Message 13. DUT: Sends ICMP Echo Reply Message 17. DUT: Does not send ICMP Echo Reply Message
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 38 (MUST)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_08: Check the suggested address to ensure it is not in use

Synopsis	The client SHOULD perform a check on the suggested address to ensure that the address is not already in use... the client must fill in its own hardware address as the sender's hardware address, and 0 as the sender's IP address
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains all of the following: - ARP Sender MAC Address field is set to <DIface-0-MAC-ADDRESS> - ARP Sender IP Address field is set to 0 - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains all of the following: - ARP Sender MAC Address field is set to <DIface-0-MAC-ADDRESS> - ARP Sender IP Address field is set to 0 - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 38 (MUST)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_09: If address is in use send a DHCPDECLINE message to the server

Synopsis	If the network address appears to be in use, the client MUST send a DHCPDECLINE message to the server
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0> 8. TESTER:DHCP Server <SERVER-1> Sends ARP Response Message to DUT through <DIface-0> containing : - ARP Sender MAC Address field set to <MAC-UNUSED-ADDRESS> - ARP Sender IP Address field set to <SERVER1-IP-POOL-0-0> 9. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 10. DUT: Sends DHCPDECLINE Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Sends ARP Request Message 7. TESTER: Verify that received ARP Request Message contains: - ARP Target IP Address field is set to <SERVER1-IP-POOL-0-0> 10. DUT: Sends DHCPDECLINE Message
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 39 (MUST)

DHCPv4_CLIENT_INITIALIZATION_ALLOCATION_10: Broadcast an ARP reply to announce the client's new IP

Synopsis	The client SHOULD broadcast an ARP reply to announce the client's new IP address and clear any outdated ARP cache entries in hosts on the client's subnet
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends ARP Response Message 7. TESTER: Verify that received ARP Response Message contains all of the following: - ARP Sender MAC Address field is set to <DIface-0-MAC-ADDRESS> - ARP Sender IP Address field is set to <SERVER1-IP-POOL-0-0>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 6. DUT: Sends ARP Response Message 7. TESTER: Verify that received ARP Response Message contains all of the following: - ARP Sender MAC Address field is set to <DIface-0-MAC-ADDRESS> - ARP Sender IP Address field is set to <SERVER1-IP-POOL-0-0>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.1 Page 39 (SHOULD)

4.8.6.9 Initialization with an externally assigned network address

DHCPv4_CLIENT_INITIALIZATION_EXTERNAL_01: The client SHOULD NOT request lease time parameters

Synopsis	The client SHOULD NOT request lease time parameters
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally assign IP Address <SERVER1-IP-POOL-0-0> with mask <SERVER1-IP-POOL-NETMASK> on DUT for interface <DIface-0> 3. TESTER: Externally cause DUT to send DHCPINFORM Message through <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Message Option containing: - Type field is not set to 55 (Parameter Request List Option) - Length field is not set to 3 - Value is not set to 51 (IP Address Lease Time Option)
Pass Criteria	5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Message Option containing: - Type field is not set to 55 (Parameter Request List Option) - Length field is not set to 3 - Value is not set to 51 (IP Address Lease Time Option)
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.3 Page 39 (SHOULD)

DHCPv4_CLIENT_INITIALIZATION_EXTERNAL_02: Send DHCPINFORM to the DHCP server if it knows the server's address

Synopsis	The client then unicasts the DHCPINFORM to the DHCP server if it knows the server's address
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally assign IP Address <SERVER1-IP-POOL-0-0> with mask <SERVER1-IP-POOL-NETMASK> on DUT for interface <DIface-0> 3. TESTER: Externally cause DUT to send DHCPINFORM Message through <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS>
Pass Criteria	5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.3 Page 39 (SHOULD)

DHCPv4_CLIENT_INITIALIZATION_EXTERNAL_03: Broadcasts to the limited (all 1s) address

Synopsis	otherwise it broadcasts the message to the limited (all 1s) broadcast address
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally assign IP Address <SERVER1-IP-POOL-0-0> with mask <SERVER1-IP-POOL-NETMASK> on DUT for interface <DIface-0> 3. TESTER: Externally cause DUT to send DHCPINFORM Message through <DIface-0> 4. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Destination IP Address field is set to <IP-BROADCAST-ADDRESS>
Pass Criteria	5. DUT: Sends DHCPINFORM Message 6. TESTER: Verify that received DHCPINFORM Message contains: - Destination IP Address field is set to <IP-BROADCAST-ADDRESS>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.3 Page 39 (SHOULD)

4.8.6.10 *Reacquisition and expiration*

DHCPv4_CLIENT_REACQUISITION_01: RENEWING state - send unicast DHCPREQUEST message

Synopsis	At time T1 the client moves to RENEWING state and sends (via unicast) a DHCPREQUEST message to the server to extend its lease
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 5. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS>
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 6. DUT: Sends DHCPREQUEST Message 7. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to <SERVER1-IP-ADDRESS>
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.5 Page 40 'Reacquisition and expiration' (MUST)

DHCPv4_CLIENT_REACQUISITION_02: On DHCPACK timeout move to REBINDING state and send DHCPREQUEST broadcast

Synopsis	If no DHCPACK arrives before time T2, the client moves to REBINDING state and sends (via broadcast) a DHCPREQUEST message to extend its lease
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till (<REMOTE-CLIENT1-T2>-<ParamToleranceTime>) for DUT to go to a state just before T2 expires 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to 0xffffffff
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that received DHCPREQUEST Message contains: - Destination IP Address field is set to 0xffffffff
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.5 Page 41 'Reacquisition and expiration' (MUST)

DHCPv4_CLIENT_REACQUISITION_03: Reacquisition and expiration T1 defaults to (0.5 * duration_of_lease)

Synopsis	T1 defaults to (0.5 * duration_of_lease). T2 defaults to (0.875 * duration_of_lease). Times T1 and T2 SHOULD be chosen with some random \"fuzz\" around a fixed value, to avoid synchronization of client reacquisition (Note: This test verifies the value of T1)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till (<REMOTE-CLIENT1-T1>-<ParamToleranceTime>) for DUT to go to a state just before T1 expires 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that the time interval between sending of last DHCPACK Message and reception of last DHCPREQUEST Message is within the range of (<REMOTE-CLIENT1-T1> - <ParamToleranceTime>) to (<REMOTE-CLIENT1-T1> + <ParamToleranceTime>) second
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that the time interval between sending of last DHCPACK Message and reception of last DHCPREQUEST Message is within the range of (<REMOTE-CLIENT1-T1> - <ParamToleranceTime>) to (<REMOTE-CLIENT1-T1> + <ParamToleranceTime>) second
Notes	Derived from RFC 2131 Section 4.4.5 Page 41 'Reacquisition and expiration' (SHOULD)

DHCPv4_CLIENT_REACQUISITION_04: Reacquisition and expiration T2 defaults to (0.875 * duration_of_lease)

Synopsis	T1 defaults to (0.5 * duration_of_lease). T2 defaults to (0.875 * duration_of_lease). Times T1 and T2 SHOULD be chosen with some random \"fuzz\" around a fixed value, to avoid synchronization of client reacquisition (Note: This test verifies the value of T2)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till (<REMOTE-CLIENT1-T2>-<ParamToleranceTime>) for DUT to go to a state just before T2 expires 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that the time interval between sending of last DHCPACK Message and reception of last DHCPREQUEST Message is within the range of (<REMOTE-CLIENT1-T2> - <ParamToleranceTime>) to (<REMOTE-CLIENT1-T2> + <ParamToleranceTime>) second
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPREQUEST Message 8. TESTER: Verify that the time interval between sending of last DHCPACK Message and reception of last DHCPREQUEST Message is within the range of (<REMOTE-CLIENT1-T2> - <ParamToleranceTime>) to (<REMOTE-CLIENT1-T2> + <ParamToleranceTime>) second
Notes	Derived from RFC 2131 Section 4.4.5 Page 41 'Reacquisition and expiration' (SHOULD)

DHCPv4_CLIENT_REACQUISITION_05: Wait time for RENEWING state

Synopsis	In both RENEWING and REBINDING states, if the client receives no response to its DHCPREQUEST message, the client SHOULD wait one-half of the remaining time until T2 (in RENEWING state) and one-half of the remaining lease time (in REBINDING state), down to a minimum of 60 seconds, before retransmitting the DHCPREQUEST message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: Initialize server1 to offer IP Address Lease Time of <HIGH-LEASE-TIME> seconds to the DUT 4. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_RENEWING 5. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 6. TESTER:DHCP Server <SERVER-1> Listens (upto (((<REMOTE-CLIENT1-T2> - <REMOTE-CLIENT1-T1>) / 2) + (2 * <ParamToleranceTime>)) second) on <DIface-0> 7. DUT: Sends 2 DHCPREQUEST Messages 8. TESTER: Verify that the time interval between reception of last two DHCPREQUEST Messages is within the range of (((<REMOTE-CLIENT1-T2> - <REMOTE-CLIENT1-T1>) / 2) - <ParamToleranceTime>) to (((<REMOTE-CLIENT1-T2> - <REMOTE-CLIENT1-T1>) / 2) + <ParamToleranceTime>) second
Pass Criteria	5. DUT: Transit finite state to DHCPCLIENT_STATE_RENEWING 7. DUT: Sends 2 DHCPREQUEST Messages 8. TESTER: Verify that the time interval between reception of last two DHCPREQUEST Messages is within the range of (((<REMOTE-CLIENT1-T2> - <REMOTE-CLIENT1-T1>) / 2) - <ParamToleranceTime>) to (((<REMOTE-CLIENT1-T2> - <REMOTE-CLIENT1-T1>) / 2) + <ParamToleranceTime>) second

Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.5 Page 41 'Reacquisition and expiration' (SHOULD)

DHCPv4_CLIENT_REACQUISITION_06: Wait time for REBINDING state

Synopsis	In both RENEWING and REBINDING states, if the client receives no response to its DHCPREQUEST message, the client SHOULD wait one-half of the remaining time until T2 (in RENEWING state) and one-half of the remaining lease time (in REBINDING state), down to a minimum of 60 seconds, before retransmitting the DHCPREQUEST message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: Initialize server1 to offer IP Address Lease Time of <VERY-HIGH-LEASE-TIME> seconds to the DUT 4. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_REBINDING 5. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 6. TESTER:DHCP Server <SERVER-1> Listens (upto (((<REMOTE-CLIENT1-LEASE-TIME> - <REMOTE-CLIENT1-T2>) / 2) + <ParamToleranceTime>) second) on <DIface-0> 7. DUT: Sends 2 DHCPREQUEST Messages 8. TESTER: Verify that the time interval between reception of last two DHCPREQUEST Messages is within the range of (((<REMOTE-CLIENT1-LEASE-TIME> - <REMOTE-CLIENT1-T2>) / 2) - <ParamToleranceTime>) to (((<REMOTE-CLIENT1-LEASE-TIME> - <REMOTE-CLIENT1-T2>) / 2) + <ParamToleranceTime>) second
Pass Criteria	5. DUT: Transit finite state to DHCPCLIENT_STATE_REBINDING 7. DUT: Sends 2 DHCPREQUEST Messages 8. TESTER: Verify that the time interval between reception of last two DHCPREQUEST Messages

	is within the range of (((<REMOTE-CLIENT1-LEASE-TIME> - <REMOTE-CLIENT1-T2>) / 2) - <ParamToleranceTime>) to (((<REMOTE-CLIENT1-LEASE-TIME> - <REMOTE-CLIENT1-T2>) / 2) + <ParamToleranceTime>) second
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.5 Page 41 'Reacquisition and expiration' (SHOULD)

DHCPv4_CLIENT_REACQUISITION_07: Stop network processing after lease time expires

Synopsis	If the lease expires before the client receives a DHCPACK, the client moves to INIT state, MUST immediately stop any other network processing and requests network initialization parameters as if the client were uninitialized (Note: In this test we verify that the DUT stops network processing after lease time expires)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till (<ParamLeaseTime>+<ParamToleranceTime>) for the lease on the DUT to expire (*INVALID*) 6. TESTER:DHCP Server <SERVER-1> Sends ICMP Echo Message to DUT through <DIface-0> containing : - Destination Harware Address field set to <DIface-0-MAC-ADDRESS> - Destination IP Address field set to <SERVER1-IP-POOL-0-0> 7. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 8. DUT: Does not send ICMP Echo Reply Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 8. DUT: Does not send ICMP Echo Reply Message
Test Iterations	
Notes	Derived from (MUST)

DHCPv4_CLIENT_REACQUISITION_08: Request network initialization parameters after lease time expires

Synopsis	If the lease expires before the client receives a DHCPACK, the client moves to INIT state, MUST immediately stop any other network processing and requests network initialization parameters as if the client were uninitialized (Note: In this test we verify that the DUT requests network initialization parameters after lease time expires)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till (<ParamLeaseTime>+<ParamToleranceTime>) for the lease on the DUT to expire 6. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 7. DUT: Sends DHCPDISCOVER Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 7. DUT: Sends DHCPDISCOVER Message
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.5 Page 41 'Reacquisition and expiration' (MUST)

DHCPv4_CLIENT_REACQUISITION_09: Continue network processing on reallocating same address

Synopsis	If the client then receives a DHCPACK allocating that client its previous network address, the client SHOULD continue network processing
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till <ParamLeaseTime> for the lease on the DUT to expire 6. TESTER: DHCP Server <SERVER-1> sends DHCPACK Message to DUT through <DIface-0> 7. TESTER:DHCP Server <SERVER-1> Sends ICMP Echo Message to DUT through <DIface-0> containing : - Destination Harware Address field set to <DIface-0-MAC-ADDRESS> - Destination IP Address field set to <SERVER1-IP-POOL-0-0> 8. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 9. DUT: Sends ICMP Echo Reply Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 9. DUT: Sends ICMP Echo Reply Message
Test Iterations	
Notes	Derived from RFC 2131 Section 4.4.5 Page 41 'Reacquisition and expiration' (SHOULD)

DHCPv4_CLIENT_REACQUISITION_10: Do not use old address if new address is given

Synopsis	If the client is given a new network address, it MUST NOT continue using the previous network address
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Externally configure DHCP Client on <DIface-0> 2. TESTER: Externally cause DUT to bring up <DIface-0> 3. TESTER: DHCP Server <SERVER-1> Cause DUT to transit its state to DHCPCLIENT_STATE_BOUND 4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 5. TESTER: Wait till (<ParamLeaseTime>-<ParamToleranceTime>) for the lease on the DUT to almost expire 6. TESTER: DHCP Server <SERVER-1> Sends DHCPACK Message to DUT through <DIface-0> containing : - Destination IP Address field set to <SERVER1-IP-POOL-0-1>(*INVALID*) - 'yiaddr' field set to <SERVER1-IP-POOL-0-1> 7. TESTER:DHCP Server <SERVER-1> Sends ICMP Echo Message to DUT through <DIface-0> containing : - Destination Harware Address field set to <DIface-0-MAC-ADDRESS> - Destination IP Address field set to <SERVER1-IP-POOL-0-0> 8. TESTER:DHCP Server <SERVER-1> Listens (upto <ParamListenTime>) on <DIface-0> 9. DUT: Does not send ICMP Echo Reply Message
Pass Criteria	4. DUT: Transit finite state to DHCPCLIENT_STATE_BOUND 9. DUT: Does not send ICMP Echo Reply Message
Test Iterations	
Notes	Derived from (MUST)

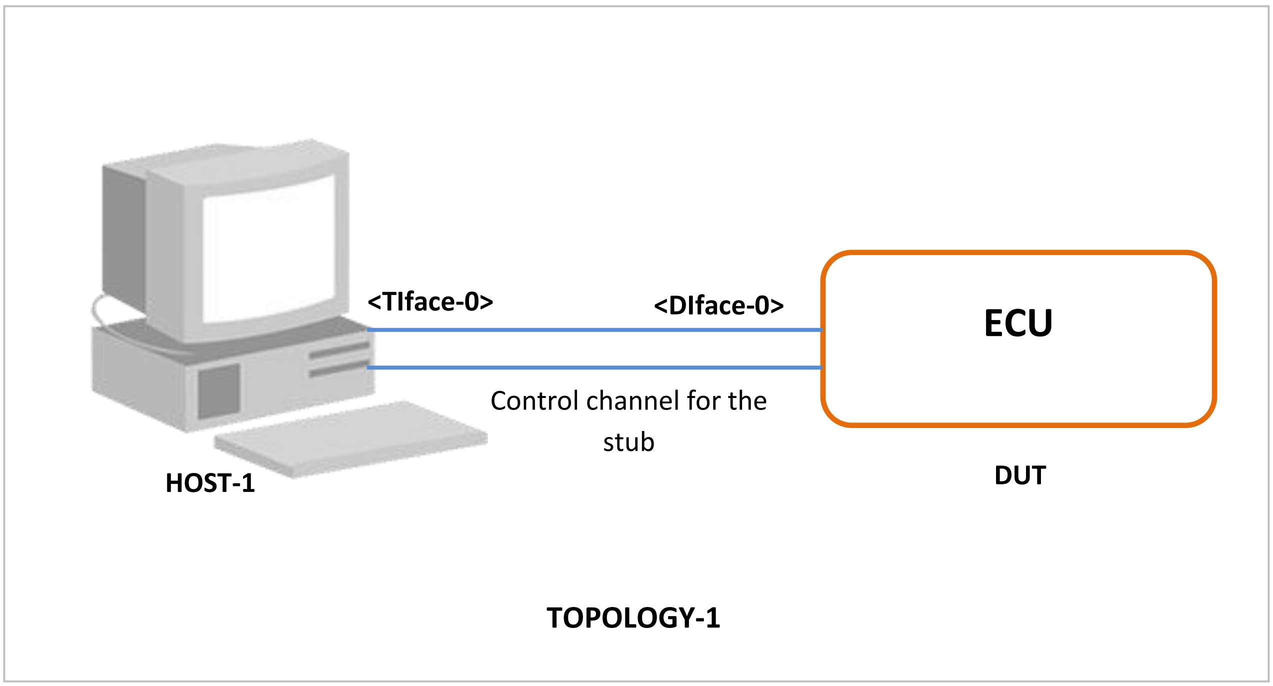
4.9 Transmisison Control Protocol (TCP)

4.9.1 General

The scope of this chapter is to specify test cases for the Transmisison Control Protocol (TCP) from the following standards:

- RFC 793 - "Transmission Control Protocol" of DARPA, Internet Program, Protocol Specification
- RFC 1122 - Requirement for Internet Hosts -- Communication Layers
- RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification

4.9.2 Simulated topologies



4.9.3 Required topology related configuration

- This test suite expects to be running against a TCP stack
- This test suite runs over Ethernet

4.9.4 Parameters used in the tests

Parameters	Description
nss	The number of data segments sent consecutively from tester side to DUT
ssz	A data segment size that is much smaller than the effective send MSS (<nsc> times <ssz> should be less than effective send MSS of DUT
tmx	The maximum time within which DUT must send a segment from its transmit buffer irrespective of its size or whether it contains PSH flag bit set
tfn	Time within which the DUT is expected to send a FIN in an error situation
wnp	A well-known port on the DUT where some standard application like TELNET server is assumed to waiting in LISTEN state
uopt	An unimplemented TCP option on DUT
msl	Maximum segment lifetime (MSL) used by DUT
Full window operation	The DUT's TCP has reached a state it is allowed to send data segments of size of tester's entire receive without getting any acknowledgement from the tester
Full-sized segment	Segment with size equal to the effective send MSS
OTW	Outside The Window (of the receiver)
PORT1	The port number the Uper Tester UDP communication is carried out through
<SEQ1>	A sequence number used to compare the Sequence or Acknowledgement Numbers in the received or sent packets.

4.9.5 Upper Tester Procedures

Several test cases within this test suite require another type of communication with the IUT that enables the tester to trigger some wished behaviors on the IUT; promting it to send certain types of messages, or to check its state and the received messages. This communication is carried out through a separate UDP port. Therefore, some procedures were conceived to fulfill the needs of such test cases:

<openTCPSocket(typeOfSocket)>: prompts the IUT to open a TCP socket depending on the type:

- Passive: opens a socket with a receiving call.
- Active: opens a socket with a sending call.

<openMultipleTCPSocket(typeOfSocket, numberOfSockets)>: prompts the IUT to open <numberOfSockets> TCP scokets of <typeOfSocket> type.

<closeTCPSocket>: Close the opened TCP socket(s).

<Confirmation>: a message sent by the IUT as a response to procedure call from the Tester through the Upper Tester channel. This message confirms, in case of the operation success, by sending a Boolean back with some additional information depending on the type of operation required from the IUT.

4.9.6 Tests

4.9.6.1 Connection Establishment and Basic Exercising of the State Machine

TCP_BASICS_01: [listen] SYN -> SYN/ACK [syn_recv]

Synopsis	TCP MUST send a SYN,ACK in response to a SYN in LISTEN state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state at a <wnp> 2. TESTER: Send a SYN to DUT at <wnp> 3. DUT: Send SYN,ACK
Pass Criteria	3. DUT: Send SYN,ACK
Test Iterations	
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_02: [syn_recv] ACK -> [established]

Synopsis	TCP MUST move on to ESTABLISHED state after receiving ACK in SYN-RCVD state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a SYN to DUT 2. DUT: Send a SYN,ACK(this will take DUT to the state SYN-RCVD) 3. TESTER: Send an ACK 4. TESTER: Verify that the DUT moves on to ESTABLISHED state
Pass Criteria	2. DUT: Send a SYN,ACK(this will take DUT to the state SYN-RCVD) 4. TESTER: Verify that the DUT moves on to ESTABLISHED state
Test Iterations	
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_03: [established] FIN -> ACK [close_wait]

Synopsis	TCP MUST send an ACK in response to a FIN received in ESTABLISHED state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state at a <wnp> 2. TESTER: Send a FIN ,ACK 3. DUT: Send ACK
Pass Criteria	3. DUT: Send ACK
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_04: [closed] data(no ack, no rst) -> RST(seq 0) [closed]

Synopsis	TCP, in CLOSED state, MUST send a RST segment with zero SEQ number in response to an incoming segment not containing RST and ACK flags
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a TCP segment with a flag set 2. DUT: Send a RST control message with zero SEQ number
Pass Criteria	2. DUT: Send a RST control message with zero SEQ number
Test Iterations	1. CASE: flag set = SYN 2. CASE: flag set = FIN 3. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p65 Event Processing (MUST)

TCP_BASICS_05: [closed] data(ack, no rst) -> RST(seq <- ack) [closed]

Synopsis	TCP, in CLOSED state, MUST send a RST in response to an incoming segment containing ACK and not containing RST and SEQ number is taken from SEG.ACK
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a segment with a flag set 2. DUT: Send a RST segment with SEQ number same as the ACK number of the incoming segment
Pass Criteria	2. DUT: Send a RST segment with SEQ number same as the ACK number of the incoming segment
Test Iterations	1. CASE: flag set = SYN,ACK 2. CASE: flag set = ACK
Notes	Derived from RFC 793 s3.9 p65 Event Processing (MUST)

TCP_BASICS_06: [closed] open -> syn

Synopsis	TCP, in CLOSED state, MUST send a SYN on an active OPEN call
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN
Pass Criteria	2. DUT: Send a SYN
Test Iterations	
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_07: [syn_sent] SYN/ACK -> ACK [established]

Synopsis	TCP MUST be capable of progressing to ESTABLISHED state after receiving SYN,ACK in SYN-SENT state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to SYN-SENT state 2. TESTER: Send a SYN,ACK 3. DUT: Send ACK 4. TESTER: Verify that the DUT moves on to ESTABLISHED state
Pass Criteria	3. DUT: Send ACK 4. TESTER: Verify that the DUT moves on to ESTABLISHED state
Test Iterations	
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_08: [established | syn_rcvd | close_wait] close -> FIN [...]

Synopsis	TCP MUST send a FIN on a CLOSE call in ESTABLISHED, SYN-RCVD or CLOSE-WAIT state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Cause the application on the DUT-side to issue a CLOSE call 3. DUT: Send a FIN
Pass Criteria	3. DUT: Send a FIN
Test Iterations	1. CASE: <wst> = ESTABLISHED 2. CASE: <wst> = SYN-RCVD 3. CASE: <wst> = CLOSE-WAIT
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_09: [last_ack] ACK of FIN -> [closed]

Synopsis	TCP MUST move on to CLOSED state after receiving an ACK of the sent FIN in LAST-ACK state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to LAST-ACK state 2. DUT: Send FIN when moving into LAST-ACK state 3. TESTER: Send ACK for the FIN just received from the DUT 4. TESTER: Send a segment without the RST flag set 5. DUT: Send a segment with the RST flag set (this will verify the the DUT has moved on to the CLOSED state)
Pass Criteria	2. DUT: Send FIN when moving into LAST-ACK state 5. DUT: Send a segment with the RST flag set (this will verify the the DUT has moved on to the CLOSED state)
Test Iterations	
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_10: [finwait-1 | finwait-2] FIN -> ACK

Synopsis	TCP MUST send an ACK after receiving a FIN in FINWAIT-1 or FINWAIT-2 state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a FIN 3. DUT: Send ACK
Pass Criteria	3. DUT: Send ACK
Test Iterations	1. CASE: <wst> = FINWAIT-1 2. CASE: <wst> = FINWAIT-2
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_11: [finwait-2 -> time_wait] delay(2*MSL) -> [closed]

Synopsis	TCP MUST move on to CLOSED state from TIME-WAIT state after a timeout of 2*MSL, where TIME-WAIT is reached through FINWAIT-2 state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to FINWAIT-2 state 2. TESTER: Send a FIN 3. DUT: Send ACK 4. TESTER: Send a FIN after 2*MSL + 20% 5. DUT: Send a RST segment(this will indicate DUT is in CLOSED state)
Pass Criteria	3. DUT: Send ACK 5. DUT: Send a RST segment(this will indicate DUT is in CLOSED state)
Test Iterations	
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_12: [closing -> time_wait] delay(2*MSL) -> [closed]

Synopsis	TCP MUST move on to CLOSED state from TIME-WAIT state after a timeout of 2*MSL, where TIME-WAIT is reached through CLOSING state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to CLOSING state 2. TESTER: Send ACK for the FIN just received from the DUT 3. TESTER: Send a FIN after 2*MSL + 20% 4. DUT: Send a RST segment(this will indicate DUT is in CLOSED state)
Pass Criteria	4. DUT: Send a RST segment(this will indicate DUT is in CLOSED state)
Test Iterations	
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_13: [finwait-2 -> time_wait] delay(<2*MSL) -> no change yet

Synopsis	TCP MUST NOT move on to CLOSED state from TIME-WAIT state before 2*MSL time expires, where TIME-WAIT state is reached through FINWAIT-2 state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to FINWAIT-2 state 2. TESTER: Send a FIN 3. DUT: Send ACK (DUT moves to TIME-WAIT state) 4. TESTER: Send a FIN within 2*MSL time 5. DUT: Send ACK
Pass Criteria	3. DUT: Send ACK (DUT moves to TIME-WAIT state) 5. DUT: Send ACK
Test Iterations	
Notes	Derived from NEGATIVE "RFC 793 s3.2 p23 Terminology" (MUST)

TCP_BASICS_14: [closing -> time_wait] delay(<2*MSL) -> no change yet

Synopsis	TCP MUST NOT move on to CLOSED state from TIME-WAIT state before 2*MSL time expires, where TIME-WAIT state is reached through CLOSING state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to CLOSING state 2. TESTER: Send ACK for the FIN just received from the DUT (DUT moves to TIME-WAIT state) 3. TESTER: Send a FIN within 2*MSL time 4. DUT: Send ACK
Pass Criteria	4. DUT: Send ACK
Test Iterations	
Notes	Derived from NEGATIVE "RFC 793 s3.2 p23 Terminology" (MUST)

TCP_BASICS_15: [finwait-1 | finwait-2] DATA -> ACK (MAY)

Synopsis	TCP MAY send an ACK after receiving a data segment in FINWAIT-1 or FINWAIT-2 state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a data segment 3. DUT: Send ACK
Pass Criteria	3. DUT: Send ACK
Test Iterations	1. CASE: <wst> = FINWAIT-1 2. CASE: <wst> = FINWAIT-2
Notes	Derived from RFC 793 s3.2 p23 Terminology (MAY)

TCP_BASICS_16: [syn-sent | close-wait | closing | last-ack | time-wait] DATA -> ignore

Synopsis	TCP MUST ignore a data segment in SYN-SENT, CLOSE-WAIT, CLOSING, LAST-ACK or TIME-WAIT state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a data segment 3. DUT: Do not send any response
Pass Criteria	3. DUT: Do not send any response
Test Iterations	1. CASE: <wst> = SYN-SENT 2. CASE: <wst> = CLOSE-WAIT 3. CASE: <wst> = CLOSING 4. CASE: <wst> = LAST-ACK 5. CASE: <wst> = TIME-WAIT
Notes	Derived from RFC 793 s3.2 p23 Terminology (MUST)

TCP_BASICS_17: Simultaneous Open Call

Synopsis	TCP MUST support simultaneous OPEN attempts
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN 3. TESTER: Send a SYN simultaneously 4. DUT: Send SYN,ACK in reply 5. TESTER: Send SYN,ACK in reply to the SYN received from the DUT 6. DUT: Send ACK 7. TESTER: Verify that the DUT moves on to ESTABLISHED state
Pass Criteria	2. DUT: Send a SYN 4. DUT: Send SYN,ACK in reply 6. DUT: Send ACK 7. TESTER: Verify that the DUT moves on to ESTABLISHED state
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.10 p87 Simultaneous Open Attempts (MUST)

4.9.6.2 Processing and Generating TCP Checksums

TCP_CHECKSUM_01: Receiver Check: checksum ok

Synopsis	Receiver TCP MUST check the checksum in any incoming segment, and MUST acknowledge in case of no error
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state at a <wnp> 2. TESTER: Send a data segment with correct checksum 3. DUT: Send ACK
Pass Criteria	3. DUT: Send ACK
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.7 p86 TCP Checksum "RFC-793 Section 3.1" (MUST)

TCP_CHECKSUM_02: Receiver Check: checksum not ok

Synopsis	Receiver TCP MUST check the checksum in any incoming segment, and MUST NOT acknowledge in case of an error
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state at a <wnp> 2. TESTER: Send a data segment with incorrect checksum 3. DUT: Do not send ACK
Pass Criteria	3. DUT: Do not send ACK
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.7 p86 TCP Checksum "RFC-793 Section 3.1" (MUST)

TCP_CHECKSUM_03: Sender compute checksum

Synopsis	Sender TCP MUST generate correct checksum
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND request for a data segment 3. DUT: Send the data segment 4. TESTER: Verify that correct checksum is present in the incoming segment
Pass Criteria	3. DUT: Send the data segment 4. TESTER: Verify that correct checksum is present in the incoming segment
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.7 p86 TCP Checksum "RFC-793 Section 3.1 p16 Header" (MUST)

TCP_CHECKSUM_04: Use clock-driven ISN selection

Synopsis	An ISN generator MUST be employed for selecting a 32 bit ISN that increments roughly every 4 microseconds
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT-side application to issue an active OPEN call 2. DUT: Send SYN 3. TESTER: Send RST,ACK to take DUT to CLOSED state 4. TESTER: Cause the DUT-side application to issue another active OPEN call 5. DUT: Send SYN 6. TESTER: Verify that sequence number of the recent SYN is more than the sequence number of the previous SYN (with the appropriate checking of wrap around of counter) at least by the (difference between the times of reception of two SYNs)x250000
Pass Criteria	2. DUT: Send SYN 5. DUT: Send SYN 6. TESTER: Verify that sequence number of the recent SYN is more than the sequence number of the previous SYN (with the appropriate checking of wrap around of counter) at least by the (difference between the times of reception of two SYNs)x250000
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.9 p87 ISN Selection "RFC-793 s3.3 p27 Sequence Numbers" (MUST)

4.9.6.3 Processing Unacceptable Acknowledgments and Out of Window Sequence Numbers

TCP_UNACCEPTABLE_01: [syn-recv] RST -> [listen] (passive open)

Synopsis	TCP MUST return to LISTEN state, on receiving an acceptable RST, in SYN-RCVD state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move to SYN-RCVD state at a <wnp>initiated by passive OPEN call 2. TESTER: Send RST segment (as if SYN,ACK just received was unexpected) 3. DUT: Do not send response 4. TESTER: Verify that the DUT moves on to LISTEN state
Pass Criteria	3. DUT: Do not send response 4. TESTER: Verify that the DUT moves on to LISTEN state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p33 Establishing a connection (MUST)

TCP_UNACCEPTABLE_02: [syn-recv] RST out-of-wdw -> [syn-recv]

Synopsis	TCP MUST NOT change state, on receiving an unacceptable RST, in SYN-RCVD state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to SYN-RCVD state at a <wnp> 2. TESTER: Send a RST segment (as if SYN,ACK just received was unexpected) with a SEQ number outside the receive window of the DUT 3. DUT: Ignore unacceptable RST 4. TESTER: Verify that the DUT remains in SYN-RCVD state
Pass Criteria	3. DUT: Ignore unacceptable RST 4. TESTER: Verify that the DUT remains in SYN-RCVD state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p33 Establishing a connection (MUST)

TCP_UNACCEPTABLE_03: [syn-recv] unacceptable ACK -> RST [syn-recv]

Synopsis	TCP MUST send a RST after receiving an unacceptable ACK in SYN-RCVD state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to SYN-RCVD state at a <wnp> 2. TESTER: Send a segment with an unacceptable ACK number 3. DUT: Send a RST segment
Pass Criteria	3. DUT: Send a RST segment
Test Iterations	
Notes	Derived from RFC 793 s3.4 p35 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_04: [established] out-of-wdw SEQ / unacceptable ACK -> empty msg w/ SEQ [established]

Synopsis	TCP, in ESTABLISHED state, MUST return ACK with proper SEQ and ACK numbers after recv a seg with OTW SEQ or unacc ACK number, and remain in same state If the connection is in a synchronized state, any unacceptable segment (out of window sequence number or unacceptable acknowledgment number) must elicit only an empty acknowledgment segment containing the current send-sequence number and an acknowledgment indicating the next sequence number expected to be received, and the connection remains in the same state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Send a data segment satisfying one of the following cases - CASE 1: Data segment with out of window SEQ number - CASE 2: Data segment with an unacceptable ACK number 3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in the ESTABLISHED state
Pass Criteria	3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in the ESTABLISHED state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p37 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_05: [listen] unacceptable ACK -> RST [listen]

Synopsis	TCP, in LISTEN state MUST send a RST after receiving a segment that is carrying an unacceptable ACK and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state at a <wnp> 2. TESTER: Send a segment with a flag set and with unacceptable ACK number 3. DUT: Send a RST segment 4. TESTER: Verify that the DUT remains in LISTEN state
Pass Criteria	3. DUT: Send a RST segment 4. TESTER: Verify that the DUT remains in LISTEN state
Test Iterations	1. CASE: flag set = SYN,ACK 2. CASE: flag set = ACK
Notes	Derived from RFC 793 s3.4 p36 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_06: [established] out-of-wdw SYN -> ACK (seq) [established]

Synopsis	TCP, in ESTABLISHED state, MUST send an ACK indicating the correct SEQ number it expects, after receiving a SYN with a SEQ number that is OTW
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Send a SYN with a sequence number outside the window 3. DUT: Send an ACK indicating next expected SEQ number
Pass Criteria	3. DUT: Send an ACK indicating next expected SEQ number
Test Iterations	
Notes	Derived from RFC 793 s3.4 p34 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_07: [listen] old SYN/ACK -> RST [listen]

Synopsis	TCP, in LISTEN state, MUST send a RST after receiving a spurious SYN,ACK that potentially corresponds to an old SYN
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state on a <wnp> 2. TESTER: Send a SYN,ACK 3. DUT: Send a RST
Pass Criteria	3. DUT: Send a RST
Test Iterations	
Notes	Derived from RFC 793 s3.4 p35 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_08: [syn-sent] unacceptable ACK -> RST(seq)

Synopsis	If the connection is in any non-synchronized state (LISTEN, SYN-SENT, SYN-RECEIVED), and the incoming segment acknowledges something not yet sent (the segment carries an unacceptable ACK), or if an incoming segment has a security level or compartment which does not exactly match the level and compartment requested for the connection, a reset is sent. (Note : This test checks for SYN-SENT state)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to SYN-SENT state 2. TESTER: Send a segment flag set and with unacceptable ACK number 3. DUT: Send a RST control message with SEQ number same as the ACK number of the incoming segment
Pass Criteria	3. DUT: Send a RST control message with SEQ number same as the ACK number of the incoming segment
Test Iterations	1. CASE: flag set = SYN,ACK 2. CASE: flag set = ACK
Notes	Derived from RFC 793 s3.4 p36 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_09: [finwait-1] out-of-wdw SEQ | unacceptable ACK -> ACK (seq, ack) [finwait-1]

Synopsis	TCP, in FINWAIT-1 state, MUST return an ACK with proper SEQ and ACK numbers after recv a seg with OTW SEQ or unacc ACK number, and remain in same state If the connection is in a synchronized state, any unacceptable segment (out of window sequence number or unacceptable acknowledgment number) must elicit only an empty acknowledgment segment containing the current send-sequence number and an acknowledgment indicating the next sequence number expected to be received, and the connection remains in the same state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to FINWAIT-1 state 2. TESTER: Send a data segment satisfying one of the following cases - CASE 1: Data segment with out of window SEQ number - CASE 2: Data segment with an unacceptable ACK number 3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in FINWAIT-1 state
Pass Criteria	3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in FINWAIT-1 state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p37 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_10: [finwait-2] out-of-wdw SEQ | unacceptable ACK -> ACK (seq, ack) [finwait-2]

Synopsis	TCP, in FINWAIT-2 state, MUST return an ACK with proper SEQ and ACK numbers after recv a seg with OTW SEQ or unacc ACK number, and remain in same state If the connection is in a synchronized state, any unacceptable segment (out of window sequence number or unacceptible acknowledgment number) must elicit only an empty acknowledgment segment containing the current send-sequence number and an acknowledgment indicating the next sequence number expected to be received, and the connection remains in the same state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to FINWAIT-2 state 2. TESTER: Send a data segment satisfying one of the following cases - CASE 1: Data segment with out of window SEQ number - CASE 2: Data segment with an unacceptable ACK number 3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in FINWAIT-2 state
Pass Criteria	3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in FINWAIT-2 state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p37 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_11: [closing] out-of-wdw SEQ | unacceptable ACK -> ACK (seq, ack) [closing]

Synopsis	TCP, in CLOSING state, MUST return an ACK with proper SEQ and ACK numbers after recv a seg with OTW SEQ or unacc ACK number, and remain in same state If the connection is in a synchronized state, any unacceptable segment (out of window sequence number or unacceptible acknowledgment number) must elicit only an empty acknowledgment segment containing the current send-sequence number and an acknowledgment indicating the next sequence number expected to be received, and the connection remains in the same state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to CLOSING state 2. TESTER: Send a data segment satisfying one of the following cases - CASE 1: Data segment with out of window SEQ number - CASE 2: Data segment with an unacceptable ACK number 3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in CLOSING state
Pass Criteria	3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in CLOSING state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p37 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_12: [last-ack] out-of-wdw SEQ | unacceptable ACK -> ACK (seq, ack) [last-ack]

Synopsis	TCP, in LAST-ACK state, MUST return an ACK with proper SEQ and ACK numbers after recv a seg with OTW SEQ or unacc ACK number, and remain in same state If the connection is in a synchronized state, any unacceptable segment (out of window sequence number or unacceptible acknowledgment number) must elicit only an empty acknowledgment segment containing the current send-sequence number and an acknowledgment indicating the next sequence number expected to be received, and the connection remains in the same state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LAST-ACK state 2. TESTER: Send a data segment satisfying one of the following cases - CASE 1: Data segment with out of window SEQ number - CASE 2: Data segment with an unacceptable ACK number 3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in LAST-ACK state
Pass Criteria	3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in LAST-ACK state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p37 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_13: [time-wait] out-of-wdw SEQ | unacceptable ACK -> ACK (seq, ack) [time-wait]

Synopsis	TCP, in TIME-WAIT state, MUST return an ACK with proper SEQ and ACK numbers after recv a seg with OTW SEQ or unacc ACK number, and remain in same state If the connection is in a synchronized state, any unacceptable segment (out of window sequence number or unacceptible acknowledgment number) must elicit only an empty acknowledgment segment containing the current send-sequence number and an acknowledgment indicating the next sequence number expected to be received, and the connection remains in the same state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to TIME-WAIT state 2. TESTER: Send a data segment satisfying one of the following cases - CASE 1: Data segment with out of window SEQ number - CASE 2: Data segment with an unacceptable ACK number 3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in TIME-WAIT state
Pass Criteria	3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Verify that the DUT remains in TIME-WAIT state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p37 Establishing a Connection (MUST)

TCP_UNACCEPTABLE_14: [close-wait] out-of-wdw SEQ | unacceptable ACK -> ACK (seq, ack) [close-wait]

Synopsis	TCP, in CLOSE-WAIT state, MUST return ACK with proper SEQ and ACK numbers after recv a seg with OTW SEQ or unacc ACK number, and remain in same state If the connection is in a synchronized state, any unacceptable segment (out of window sequence number or unacceptible acknowledgment number) must elicit only an empty acknowledgment segment containing the current send-sequence number and an acknowledgment indicating the next sequence number expected to be received, and the connection remains in the same state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to CLOSE-WAIT state 2. TESTER: Send a data segment satisfying one of the following cases - CASE 1: Data segment with out of window SEQ number - CASE 2: Data segment with an unacceptable ACK number 3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 4. TESTER: Sending a data packet to DUT 5. DUT: Does not send any response 6.TESTER: Verify that the DUT stays in CLOSE-WAIT state
Pass Criteria	3. DUT: Send an ACK with current send SEQ number and ACK number indicating next SEQ number expected 5. DUT: Does not send any response 6.TESTER: Verify that the DUT stays in CLOSE-WAIT state
Test Iterations	
Notes	Derived from RFC 793 s3.4 p37 Establishing a Connection (MUST)

4.9.6.4 Processing TCP OPEN Calls Received from the Application Layer

TCP_CALL_OPEN_01: Open: Active Open w/ unspec socket -> error

Synopsis	For an active OPEN call with unspecified foreign socket, TCP MUST return error: foreign socket unspecified
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call with unspecified foreign socket 2. TESTER: Cause the application on the DUT-side to return the error code 3. DUT: Returns \"error: foreign socket unspecified\"
Pass Criteria	3. DUT: Returns \"error: foreign socket unspecified\"
Test Iterations	
Notes	Derived from RFC 793 s3.9 p56 Event Processing (MUST)

TCP_CALL_OPEN_02: Open: Passive Open illegal for this process

Synopsis	TCP MUST refuse a passive OPEN call on a <wnp> indicating \"error: connection illegal for this process\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause an application on the DUT-side to issue a passive OPEN call on a <wnp> 2. DUT: Issue a passive open call 3. TESTER: Verify that the DUT refuses to create new connection indicating \"error: connection illegal for this process\" to the application
Pass Criteria	3. TESTER: Verify that the DUT refuses to create new connection indicating \"error: connection illegal for this process\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p54 OPEN Call (MUST)

TCP_CALL_OPEN_03: Open: Connection already exists [syn-sent] (active open)

Synopsis	If an OPEN call arrives on state SYN-SENT, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the SYN-SENT state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

TCP_CALL_OPEN_04: Open: Connection already exists [syn-rcvd] (passive open)

Synopsis	If an OPEN call arrives on state SYN-RCVD, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the SYN-RCVD state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

TCP_CALL_OPEN_05: Open: Connection already exists [established]

Synopsis	If an OPEN call arrives on state ESTABLISHED, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

TCP_CALL_OPEN_06: Open: Connection already exists [finwait-1]

Synopsis	If an OPEN call arrives on state FINWAIT-1, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the FINWAIT-1 state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

TCP_CALL_OPEN_07: Open: Connection already exists [finwait-2]

Synopsis	If an OPEN call arrives on state FINWAIT-2, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the FINWAIT-2 state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

TCP_CALL_OPEN_08: Open: Connection already exists [close-wait]

Synopsis	If an OPEN call arrives on state CLOSE-WAIT, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the CLOSE-WAIT state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

TCP_CALL_OPEN_09: Open: Connection already exists [closing]

Synopsis	If an OPEN call arrives on state CLOSING, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the CLOSING state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

TCP_CALL_OPEN_10: Open: Connection already exists [last-ack]

Synopsis	If an OPEN call arrives on state LAST-ACK, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the LAST-ACK state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

TCP_CALL_OPEN_11: Open: Connection already exists [time-wait]

Synopsis	If an OPEN call arrives on state TIME-WAIT, TCP MUST return \"error: connection already exists\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the TIME-WAIT state 2. TESTER: Cause the application on the DUT-side to issue an OPEN call 3. DUT: Issues an OPEN call 4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection already exists\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p55 Event Processing (MUST)

4.9.6.5 Processing TCP SEND Calls Received from the Application Layer

TCP_CALL_SEND_01: Send: Connection does not exists [closed]

Synopsis	If a SEND call arrives on CLOSED state, and user has got proper access to such conn, TCP MUST return \"error: connection does not exist\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue a SEND call 2. DUT: Issue a SEND call 3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
Pass Criteria	3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p56 Event Processing (MUST)

TCP_CALL_SEND_02: Send: Connection closing [finwait-1]

Synopsis	If SEND call arrives on state FINWAIT-1, TCP MUST return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to the FINWAIT-1 state 2. TESTER: Cause the application on the DUT-side to issue a SEND call 3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p57 Event Processing (MUST)

TCP_CALL_SEND_03: Send: Connection closing [finwait-2]

Synopsis	If SEND call arrives on state FINWAIT-2, TCP MUST return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to the FINWAIT-2 state 2. TESTER: Cause the application on the DUT-side to issue a SEND call 3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p57 Event Processing (MUST)

TCP_CALL_SEND_04: Send: Foreign socket unspecified [listen]

Synopsis	If SEND call arrives on LISTEN state and foreign socket was not specified, TCP MUST return \"error: foreign socket unspecified\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to LISTEN state 2. TESTER: Cause the application on the DUT-side to issue SEND call without specifying any foreign socket 3. DUT: Issues a SEND call 4. TESTER: Verify that the DUT returns \"error: foreign socket unspecified\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: foreign socket unspecified\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p56 Event Processing (MUST)

TCP_CALL_SEND_05: Send: Connection closing [closing]

Synopsis	If SEND call arrives on state CLOSING, TCP MUST return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to the CLOSING state 2. TESTER: Cause the application on the DUT-side to issue a SEND call 3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p57 Event Processing (MUST)

TCP_CALL_SEND_06: Send: Connection closing [last-ack]

Synopsis	If SEND call arrives on state LAST-ACK, TCP MUST return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to the LAST-ACK state 2. TESTER: Cause the application on the DUT-side to issue a SEND call 3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p57 Event Processing (MUST)

TCP_CALL_ SEND_07: Send: Queueing SEND [syn-sent] (MAY)

Synopsis	If a SEND call arrives on SYN-SENT state, TCP MAY queue the request
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT to move on to SYN-SENT state 2. TESTER: Cause the application on the DUT-side to issue a SEND call 3. TESTER: Send SYN,ACK 4. DUT: Send ACK 5. DUT: Send the data segment
Pass Criteria	4. DUT: Send ACK 5. DUT: Send the data segment
Test Iterations	
Notes	Derived from RFC 793 s3.9 p56 Event Processing (MAY)

TCP_CALL_ SEND_08: Send: Queueing SEND [syn-rcvd] (MAY)

Synopsis	If a SEND call arrives on SYN-RCVD state, TCP MAY queue the request
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT to move on to SYN-RCVD state 2. TESTER: Cause the application to issue a SEND call 3. TESTER: Send ACK 4. DUT: Send the data segment
Pass Criteria	4. DUT: Send the data segment
Notes	Derived from RFC 793 s3.9 p56 Event Processing (MAY)

TCP_CALL_SEND_09: Send: Connection closing [time-wait]

Synopsis	If SEND call arrives on state TIME-WAIT, TCP MUST return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to the TIME-WAIT state 2. TESTER: Cause the application on the DUT-side to issue a SEND call 3. DUT: Issue a SEND call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p57 Event Processing (MUST)

4.9.6.6 Processing TCP RECEIVE Calls Received from the Application Layer

TCP_CALL_RECEIVE_01: Receive: Connection closing [closing]

Synopsis	If RECEIVE call arrives on state CLOSING, TCP MUST return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the CLOSING state 2. TESTER: Cause the application to issue a RECEIVE call 3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p59 Event Processing (MUST)

TCP_CALL_RECEIVE_02: Receive: Connection does not exist [closed]

Synopsis	If RECEIVE call arrives on CLOSED state, and the user has got access to such conn, TCP MUST return \"error: connection does not exist\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue a RECEIVE call 2. DUT: Issues a RECEIVE call 3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
Pass Criteria	3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p58 Event Processing (MUST)

TCP_CALL_RECEIVE_03: Receive: Queueing Receive [syn-sent]

Synopsis	If RECEIVE call arrives on SYN-SENT state, TCP MUST queue the request for processing after entering ESTABLISHED state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to SYN-SENT state 2. TESTER: Cause the application on the DUT-side to issue a RECEIVE call 3. DUT: Queue the request 4. TESTER: Send SYN,ACK 5. DUT: Send ACK 6. TESTER: Send a data segment 7. TESTER: Verify that the DUT returns the data in response to the queued RECEIVE call
Pass Criteria	5. DUT: Send ACK 7. TESTER: Verify that the DUT returns the data in response to the queued RECEIVE call
Test Iterations	
Notes	Derived from RFC 793 s3.9 p58 Event Processing (MUST)

TCP_CALL_RECEIVE_04: Receive: Reassemble queues incoming segments [established | finwait-1 | finwait-2]

Synopsis	If RECEIVE call arrives in ESTABLISHED, FINWAIT-1 or FINWAIT-2 state, TCP MUST reassemble queued incoming segments and return to the application
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to the <wst> state 2. TESTER: Send <nss> number of small sized <ssz> data segments 3. TESTER: Cause the application on the DUT-side to issue a RECEIVE call having more than <nss>*<ssz> buffer size 4. DUT: Issues a RECEIVE call 5. TESTER: Verify that the DUT returns the reassembled data to the receiving application
Pass Criteria	5. TESTER: Verify that the DUT returns the reassembled data to the receiving application
Test Iterations	1. CASE: <wst> = ESTABLISHED 2. CASE: <wst> = FINWAIT-1 3. CASE: <wst> = FINWAIT-2
Notes	Derived from RFC 793 s3.9 p58 Event Processing (MUST)

TCP_CALL_RECEIVE_05: Receive: Queued data [close-wait]

Synopsis	If RECEIVE call arrives on CLOSE-WAIT state and there is data awaiting delivery, TCP MUST return the data to the application
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Send a data segment with FIN flag set 3. TESTER: Cause the application on the DUT-side to issue a RECEIVE call 4. DUT: Issues a RECEIVE call 5. TESTER: Verify that the DUT returns the data to the receiving application
Pass Criteria	5. TESTER: Verify that the DUT returns the data to the receiving application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p59 Event Processing (MUST)

TCP_CALL_RECEIVE_06: Receive: Connection closing on no queued data [close-wait]

Synopsis	If RECEIVE call arrives on CLOSE-WAIT state and there is no data awaiting delivery, TCP MUST respond \"error: connection closing\" to the application
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to CLOSE-WAIT state 2. TESTER: Cause the application on the DUT-side to issue a RECEIVE call 3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT responds \"error: connection closing\" to the receiving application
Pass Criteria	4. TESTER: Verify that the DUT responds \"error: connection closing\" to the receiving application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p59 Event Processing (MUST)

TCP_CALL_RECEIVE_07: Receive: Connection closing [last-ack]

Synopsis	If RECEIVE call arrives on state LAST-ACK, TCP MUST return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the LAST-ACK state 2. TESTER: Cause the application to issue a RECEIVE call 3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p59 Event Processing (MUST)

TCP_CALL_RECEIVE_08: Receive: Connection closing [time-wait]

Synopsis	If RECEIVE call arrives on state TIME-WAIT, TCP MUST return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to the TIME-WAIT state 2. TESTER: Cause the application to issue a RECEIVE call 3. DUT: Issues a RECEIVE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p59 Event Processing (MUST)

4.9.6.7 Processing CLOSE calls Received from the Application Layer

TCP_CALL_CLOSE_01: Close: Connection does not exist [closed]

Synopsis	For CLOSE call on CLOSED state, if the user has got access to such a connection, TCP MUST return \"error: connection does not exist\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue a CLOSE call 2. DUT: Issues a CLOSE call 3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
Pass Criteria	3. TESTER: Verify that the DUT returns \"error: connection does not exist\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p60 Event Processing (MUST)

TCP_CALL_CLOSE_02: Close: Connection closing [closing]

Synopsis	For CLOSE call on state CLOSING, TCP MUST return with \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to CLOSING state 2. TESTER: Cause the application on the DUT-side to issue a CLOSE call 3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p61 Event Processing (MUST)

TCP_CALL_CLOSE_03: Close: Connection closing [finwait-1]

Synopsis	For CLOSE call on state FINWAIT-1, TCP SHOULD return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to FINWAIT-1 state 2. TESTER: Cause the application on the DUT-side to issue a CLOSE call 3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p60 Event Processing (SHOULD)

TCP_CALL_CLOSE_04: Close: Connection closing [time-wait]

Synopsis	For CLOSE call on state TIME-WAIT, TCP MUST return with \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to TIME-WAIT state 2. TESTER: Cause the application on the DUT-side to issue a CLOSE call 3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p61 Event Processing (MUST)

TCP_CALL_CLOSE_05: Close: Connection closing [finwait-2]

Synopsis	For CLOSE call on state FINWAIT-2, TCP SHOULD return \"error: connection closing\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to FINWAIT-2 state 2. TESTER: Cause the application on the DUT-side to issue a CLOSE call 3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Pass Criteria	3. DUT: Issues a CLOSE call 4. TESTER: Verify that the DUT returns \"error: connection closing\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p60 Event Processing (SHOULD)

4.9.6.8 Processing TCP ABORT Calls Received from the Application Layer

TCP_CALL_ABORT_01: Abort: Connection does not exist [closed]

Synopsis	For ABORT call on CLOSED state, if user has got access to such connection, TCP MUST return with \"error: connection does not exist\"
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an ABORT call 2. DUT: Issues an ABORT call 3. TESTER: Verify that DUT MUST return \"error: connection does not exist\" to the application
Pass Criteria	3. TESTER: Verify that DUT MUST return \"error: connection does not exist\" to the application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p62 Event Processing (MUST)

TCP_CALL_ABORT_02: Abort: Closing connection [established] -> [closed]

Synopsis	For ABORT call in ESTABLISHED state TCP MUST enter CLOSED state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application to issue an ABORT call 3. DUT: Send a RST control message 4. TESTER: Verify that the DUT moves on to CLOSED state
Pass Criteria	3. DUT: Send a RST control message 4. TESTER: Verify that the DUT moves on to CLOSED state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p62 Event Processing (MUST)

TCP_CALL_ABORT_03: Abort: Closing connection [closing | last-ack | time-wait] -> [closed]

Synopsis	For ABORT call on CLOSING, LAST-ACK or TIME-WAIT state, TCP MUST respond with \"ok\" and enter CLOSED state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to the <wst> state 2. TESTER: Cause the application on the DUT-side to issue an ABORT call 3. DUT: Issues an ABORT call 4. TESTER: Verify that the DUT responds with \"ok\" to the application 5. TESTER: Verify that the DUT moves on to CLOSED state
Pass Criteria	4. TESTER: Verify that the DUT responds with \"ok\" to the application 5. TESTER: Verify that the DUT moves on to CLOSED state
Test Iterations	1. CASE: <wst> = CLOSING 2. CASE: <wst> = LAST-ACK 3. CASE: <wst> = TIME-WAIT
Notes	Derived from RFC 793 s3.9 p62 Event Processing (MUST)

4.9.6.9 TCP Packet Flag Generation in Response to Receiving Invalid Packets

TCP_FLAGS_INVALID_01: [listen] RST -> ignore

Synopsis	TCP MUST ignore an incoming segment with RST flag in LISTEN state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state at a <wnp> 2. TESTER: Send a segment with SYN and RST 3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in the LISTEN state
Pass Criteria	3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in the LISTEN state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p65 Event Processing (MUST)

TCP_FLAGS_INVALID_02: [listen] ACK-> RST(seq <- ack) [listen]

Synopsis	TCP in LISTEN state, TCP MUST send RST in response to incoming segment with ACK and remain in the same state, SEQ number of RST is taken from SEG.ACK
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state at a <wnp> 2. TESTER: Send a segment with SYN and ACK 3. DUT: Send a RST control message with SEQ number same as the ACK number of the incoming segment 4. TESTER: Verify that the DUT remains in the LISTEN state
Pass Criteria	3. DUT: Send a RST control message with SEQ number same as the ACK number of the incoming segment 4. TESTER: Verify that the DUT remains in the LISTEN state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p65 Event Processing (MUST)

TCP_FLAGS_INVALID_03: [syn-sent] ACK/RST-> ignore

Synopsis	TCP in SYN-SENT state MUST ignore a segment carrying an unacceptable ACK and RST
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to SYN-SENT state 2. TESTER: Send a segment having both ACK and RST flags with unacceptable ACK number 3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in SYN-SENT state
Pass Criteria	3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in SYN-SENT state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p66 Event Processing (MUST)

TCP_FLAGS_INVALID_04: [syn-sent] RST-> ignore

Synopsis	TCP, in SYN-SENT state MUST ignore a RST control message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to SYN-SENT state 2. TESTER: Send a RST control message 3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in SYN-SENT state
Pass Criteria	3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in SYN-SENT state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p67 Event Processing (MUST)

TCP_FLAGS_INVALID_05: [syn-sent] ACK/RST-> [CLOSED]

Synopsis	TCP, in SYN-SENT state MUST move on to CLOSED state after receiving a segment with ACK and RST and acceptable ACK number
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to SYN-SENT state 2. TESTER: Send a segment a flag set along with RST flag with acceptable ACK number 3. DUT: Goes to CLOSED state 4. TESTER: Verify that the DUT moves on to CLOSED state
Pass Criteria	3. DUT: Goes to CLOSED state 4. TESTER: Verify that the DUT moves on to CLOSED state
Test Iterations	1. CASE: flag set = SYN,ACK 2. CASE: flag set = ACK
Notes	Derived from RFC 793 s3.9 p67 Event Processing (MUST)

TCP_FLAGS_INVALID_06: [syn-sent] no syn/no rst-> do nothing

Synopsis	TCP, in SYN-SENT state MUST drop the packet and remain in the same state after receiving a segment with neither SYN nor RST flag set
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to SYN-SENT state 2. TESTER: Send a segment of type <stp> along with ACK flag with acceptable ACK number 3. DUT: DUT does not change state 4. TESTER: Verify that the DUT does not send a SYN, ACK and remains in SYN-SENT state
Pass Criteria	3. DUT: DUT does not change state 4. TESTER: Verify that the DUT does not send a SYN, ACK and remains in SYN-SENT state
Test Iterations	1. CASE: <stp> = no data 2. CASE: <stp> = data
Notes	Derived from RFC 793 s3.9 p68 Event Processing (MUST)

TCP_FLAGS_INVALID_07: [syn-rcvd] !RST(otw SEQ)-> ACK(SEQ)

Synopsis	TCP, in SYN-RCVD state, MUST send ACK with next expected SEQ num on receiving any segment (without RST) with OTW SEQ number and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to SYN-RCVD state 2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number 3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in SYN-RCVD state
Pass Criteria	3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in SYN-RCVD state
Test Iterations	1. CASE: flag set = SYN 2. CASE: flag set = SYN,ACK 3. CASE: flag set = ACK 4. CASE: flag set = FIN 5. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p69 Event Processing (MUST)

TCP_FLAGS_INVALID_08: [established] (otw SEQ)-> ACK(seq) [established]

Synopsis	TCP, in ESTABLISHED state, MUST send an ACK with next expected SEQ number after recv any segment with OTW SEQ number and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number 3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in ESTABLISHED state
Pass Criteria	3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in ESTABLISHED state
Test Iterations	1. CASE: flag set = SYN 2. CASE: flag set = SYN,ACK 3. CASE: flag set = ACK 4. CASE: flag set = FIN 5. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p69 Event Processing (MUST)

TCP_FLAGS_INVALID_09: [finwait-1] (otw SEQ)-> ACK(seq) [finwait-1]

Synopsis	TCP, in FINWAIT-1 state, MUST send ACK with next expected SEQ number after receiving any segment with OTW SEQ number and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to FINWAIT-1 state 2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number 3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in FINWAIT-1 state
Pass Criteria	3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in FINWAIT-1 state
Test Iterations	1. CASE: flag set = SYN 2. CASE: flag set = SYN,ACK 3. CASE: flag set = ACK 4. CASE: flag set = FIN 5. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p69 Event Processing Control Protocol (TCP) (MUST)

TCP_FLAGS_INVALID_10: [finwait-2] (otw SEQ)-> ACK(seq) [finwait-2]

Synopsis	TCP, in FINWAIT-2 state, MUST send ACK with next expected SEQ number after receiving any segment with OTW SEQ number and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to FINWAIT-2 state 2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number 3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in FINWAIT-2 state
Pass Criteria	3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in FINWAIT-2 state
Test Iterations	1. CASE: flag set = SYN 2. CASE: flag set = SYN,ACK 3. CASE: flag set = ACK 4. CASE: flag set = FIN 5. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p69 Event Processing (MUST)

TCP_FLAGS_INVALID_11: [close-wait] (otw SEQ)-> ACK(seq) [close-wait]

Synopsis	TCP, in CLOSE-WAIT state, MUST send an ACK with next expected SEQ number after recv any segment with OTW SEQ number and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to CLOSE-WAIT state 2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number 3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in CLOSE-WAIT state
Pass Criteria	3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in CLOSE-WAIT state
Test Iterations	1. CASE: flag set = SYN 2. CASE: flag set = SYN,ACK 3. CASE: flag set = ACK 4. CASE: flag set = FIN 5. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p69 Event Processing (MUST)

TCP_FLAGS_INVALID_12: [closing] (otw SEQ)-> ACK(seq) [closing]

Synopsis	TCP, in CLOSING state, MUST send an ACK with next expected SEQ number after receiving any segment with OTW SEQ number and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to CLOSING state 2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number 3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in CLOSING state
Pass Criteria	3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in CLOSING state
Test Iterations	1. CASE: flag set = SYN 2. CASE: flag set = SYN,ACK 3. CASE: flag set = ACK 4. CASE: flag set = FIN 5. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p69 Event Processing (MUST)

TCP_FLAGS_INVALID_13: [last-ack] (otw SEQ)-> ACK(seq) [last-ack]

Synopsis	TCP, in LAST-ACK state, MUST send an ACK with next expected SEQ number after receiving any segment with OTW SEQ number and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to LAST-ACK state 2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number 3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in LAST-ACK state
Pass Criteria	3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify that the DUT remains in LAST-ACK state
Test Iterations	1. CASE: flag set = SYN 2. CASE: flag set = SYN,ACK 3. CASE: flag set = ACK 4. CASE: flag set = FIN 5. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p69 Event Processing (MUST)

TCP_FLAGS_INVALID_14: [time-wait] (otw SEQ)-> ACK(seq) [time-wait]

Synopsis	TCP, in TIME-WAIT state, MUST send an ACK with next expected SEQ number after recv any segment with OTW SEQ number and remain in the same state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to TIME-WAIT state 2. TESTER: Send a segment with a flag set, RST=0 and with an unacceptable SEQ number 3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify DUT remains in TIME-WAIT state for other cases
Pass Criteria	3. DUT: Send an ACK with ACK number indicating the correct expected next SEQ number 4. TESTER: Verify DUT remains in TIME-WAIT state for other cases
Test Iterations	1. CASE: flag set = FIN 2. CASE: flag set = Data segment
Notes	Derived from RFC 793 s3.9 p69 Event Processing (MUST)

TCP_FLAGS_INVALID_15: [!closed & !syn-sent & !listen] RST(otw SEQ) -> ignore

Synopsis	TCP, in any state other than CLOSED, SYN-SENT and LISTEN states, MUST ignore a RST segment with OTW SEQ number
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a RST segment with an unacceptable SEQ number 3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in <wst> state
Pass Criteria	3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in <wst> state
Test Iterations	1. CASE: <wst> = SYN-RECEIVED 2. CASE: <wst> = ESTABLISHED 3. CASE: <wst> = FINWAIT-1 4. CASE: <wst> = FINWAIT-2 5. CASE: <wst> = CLOSE-WAIT 6. CASE: <wst> = CLOSING 7. CASE: <wst> = LAST-ACK 8. CASE: <wst> = TIME-WAIT
Notes	Derived from RFC 793 s3.9 p69 Event Processing (MUST)

4.9.6.10 *Processing TCP Flags*

TCP_FLAGS_PROCESSING_01: [syn-rcvd] RST -> connection-refused (active open)

Synopsis	TCP, in SYN-RCVD state, reached through active OPEN, MUST inform \"connection refused\" to application on recv a RST
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to SYN-RCVD state initiated by active OPEN call 2. TESTER: Send a RESET segment 3. TESTER: Cause DUT side app to issue a receive call 4. DUT: Returns error 5. TESTER: Verify that the DUT informs \"connection refused\" to the receiving application
Pass Criteria	4. DUT: Returns error 5. TESTER: Verify that the DUT informs \"connection refused\" to the receiving application
Test Iterations	
Notes	Derived from RFC 793 s3.9 p70 Event Processing (MUST)

TCP_FLAGS_PROCESSING_02: [established | FinWait-1 | FinWait-2 | Close-Wait] RST -> [closed]

Synopsis	TCP, in SYNC-RCVD, ESTABLISHED, FINWAIT-1, FINWAIT-2, CLOSE-WAIT states, MUST return to CLOSED state on RESET
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a RST segment 3. DUT: Moves to CLOSED state 4. TESTER: Verify that the DUT moves on to CLOSED state
Pass Criteria	3. TESTER: Verify that the DUT moves on to CLOSED state
Test Iterations	1. CASE: <wst> = SYN-RCVD 2. CASE: <wst> = ESTABLISHED 3. CASE: <wst> = FINWAIT-1 4. CASE: <wst> = FINWAIT-2 5. CASE: <wst> = CLOSE-WAIT
Notes	Derived from RFC 793 s3.9 p70 Event Processing (MUST)

TCP_FLAGS_PROCESSING_03: [closing | last-ack] RST -> [closed]

Synopsis	TCP, in CLOSING, LAST-ACK state, MUST return to CLOSED state on receiving a RST
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a RST segment 3. DUT: Goes to CLOSED state 4. TESTER: Verify that the DUT moves on to CLOSED state
Pass Criteria	4. TESTER: Verify that the DUT moves on to CLOSED state
Test Iterations	1. CASE: <wst> = CLOSING 2. CASE: <wst> = LAST-ACK
Notes	Derived from RFC 793 s3.9 p70 Event Processing

TCP_FLAGS_PROCESSING_04: [time-wait] RST -> [closed]

Synopsis	TCP, in TIME-WAIT state, MUST return to CLOSED state on receiving a RST (Note: Test case will be executes only if DUT does not support RFC 1337 based Time Wait Assassination Avoidance)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to TIME-WAIT state 2. TESTER: Send a RST segment 3. DUT: Goes to CLOSED state 4. TESTER: Verify that the DUT moves on to CLOSED state
Pass Criteria	4. TESTER: Verify that the DUT moves on to CLOSED state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p70 Event Processing

TCP_FLAGS_PROCESSING_05: [syn-rcvd] SYN -> [closed]

Synopsis	TCP, in SYN-RCVD state, MUST send a reset and go to CLOSED state, on recv a seg with SYN in window
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to SYN-RCVD state 2. TESTER: Send a segment of type Stp 3. DUT: Send a RST segment 4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site and the DUT moves on to CLOSED state
Pass Criteria	3. DUT: Send a RST segment 4.TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site and the DUT moves on to CLOSED state
Test Iterations	1. CASE: <stp> = SYN in window 2. CASE: <stp> = SYN,ACK in window
Notes	Derived from RFC 793 s3.9 p71 Event Processing (MUST)

TCP_FLAGS_PROCESSING_06: [time-wait] FIN -> FIN/ACK

Synopsis	TCP, in TIME-WAIT state, MUST acknowledge a retransmitted FIN and restart the 2*MSL time-out
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to TIME-WAIT state 2. TESTER: Send the last FIN once more 3. DUT: Send ACK for the retransmitted FIN 4. TESTER: Wait for 1.5*MSL time 5. TESTER: Verify that the DUT remains in TIME-WAIT state
Pass Criteria	3. DUT: Send ACK for the retransmitted FIN 5. TESTER: Verify that the DUT remains in TIME-WAIT state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p73 Event Processing (MUST)

TCP_FLAGS_PROCESSING_07: [close-wait | closing | last-ack | time-wait] URG -> ignore

Synopsis	TCP, in CLOSE-WAIT, CLOSING, LAST-ACK or TIME-WAIT state, MUST ignore any segment with only URG flag set
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a data segment with only URG flag set 3. DUT: Do not send any response
Pass Criteria	3. DUT: Do not send any response
Test Iterations	1. CASE: <wst> = CLOSE-WAIT 2. CASE: <wst> = CLOSING 3. CASE: <wst> = LAST-ACK 4. CASE: <wst> = TIME-WAIT
Notes	Derived from RFC 793 s3.9 p74 Event Processing (MUST)

TCP_FLAGS_PROCESSING_08: [closed| listen | syn-sent] FIN -> ignore

Synopsis	TCP, in CLOSED, LISTEN or SYN-SENT state, MUST not process a FIN
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a FIN 3. DUT: in SYN-SENT or LISTEN state : Do not send any response or retransmit SYN In CLOSED state : send a RST segment 4. TESTER: Verify that the DUT remains in <wst> state
Pass Criteria	1. DUT: in SYN-SENT or LISTEN state : Do not send any response or retransmit SYN In CLOSED state : send a RST segment 4. TESTER: Verify that the DUT remains in <wst> state
Test Iterations	1. CASE: <wst> = CLOSED 2. CASE: <wst> = LISTEN 3. CASE: <wst> = SYN-SENT
Notes	Derived from RFC 793 s3.9 p75 Event Processing (MUST)

TCP_FLAGS_PROCESSING_09: [close-wait| closing | last-ack] FIN -> ignore

Synopsis	TCP, in CLOSE-WAIT, CLOSING or LAST-ACK state, MUST not change state after receiving a FIN
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a FIN 3. DUT: Do not change state 4. TESTER: Verify that the DUT remains in <wst> state
Pass Criteria	4. TESTER: Verify that the DUT remains in <wst> state
Test Iterations	1. CASE: <wst> = CLOSE-WAIT 2. CASE: <wst> = CLOSING 3. CASE: <wst> = LAST-ACK
Notes	Derived from RFC 793 s3.9 p75 Event Processing (MUST)

TCP_FLAGS_PROCESSING_10: [established] piggybacking

Synopsis	TCP, in ESTABLISHED state, SHOULD piggyback acknowledgement with a segment being transmitted (whenever possible) without incurring undue delay
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move to ESTABLISHED state 2. TESTER: Send the last ACK once more 3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in ESTABLISHED state
Pass Criteria	3. DUT: Send the data segment 5. DUT: Do not send this data segment 7. DUT: Send the pending data segment piggybacking the acknowledgement within 0.5 sec
Test Iterations	
Notes	Derived from RFC 793 s3.9 p74 Event Processing (SHOULD)

TCP_FLAGS_PROCESSING_11: [established] duplicate ACK -> ignore

Synopsis	TCP, in ESTABLISHED state, MUST ignore a duplicate ACK
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move to ESTABLISHED state 2. TESTER: Send the last ACK once more 3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in ESTABLISHED state
Pass Criteria	3. DUT: Do not send any response 4. TESTER: Verify that the DUT remains in ESTABLISHED state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p72 Event Processing (MUST)

4.9.6.11 *Closing a TCP Connection*

TCP_CLOSING_01: [syn-rvcd] RST -> [closed] (active open)

Synopsis	TCP, if starts as an active connection and reaches SYN-RCVD state MUST go to CLOSED state on RESET
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT: Move on to SYN-RCVD state starting with an active OPEN call 2. TESTER: Send a RST 3. DUT: Go to CLOSED state 4. TESTER: Verify that DUT goes to CLOSED state
Pass Criteria	4. TESTER: Verify that DUT goes to CLOSED state
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.13 p88 Closing a connection "RFC-793 Section 3.5" (MUST)

TCP_CLOSING_02: [established] RST -> [closed] inform upper layer

Synopsis	TCP, in ESTABLISHED state MUST inform the application in case of aborting from remote site
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT: Move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a RECEIVE call 3. TESTER: Send a RST 4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
Pass Criteria	4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.13 p87 Closing a connection "RFC-793 Section 3.5" (MUST)

TCP_CLOSING_03: RST with DATA

Synopsis	TCP SHOULD allow a received RST segment to include data
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Send a RST segment containing some data 3. DUT: Do not send any response 4. TESTER: Verify that the DUT is in CLOSED state
Pass Criteria	3. DUT: Do not send any response 4. TESTER: Verify that the DUT is in CLOSED state
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.12 p87 RST Segment "RFC-793 Section 3.4" (MUST)

TCP_CLOSING_04: [finwait-1 | finwait-2] DATA half duplex closing

Synopsis	TCP MAY implement a half-duplex TCP close sequence, so that an application after closing cannot continue to read data from the connection
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to <wst> state 2. TESTER: Send a data segment 3. DUT: Send a RST control message 4. TESTER: Verify that the receiving application does not receive the data
Pass Criteria	3. DUT: Send a RST control message 4. TESTER: Verify that the receiving application does not receive the data
Test Iterations	1. CASE: <wst> = FINWAIT-1 2. CASE: <wst> = FINWAIT-2
Notes	Derived from RFC 1122 s4.2.2.13 p88 Closing a connection "RFC-793 Section 3.5" (MAY)

TCP_CLOSING_05: [time-wait] SYN -> SYN/ACK

Synopsis	TCP MAY accept a new SYN from the remote TCP in TIME-WAIT state
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to TIME-WAIT state through <wst> state 2. TESTER: Send a SYN 3. DUT: Send SYN,ACK 4. TESTER: Send ACK and verify that the DUT moves on to ESTABLISHED state
Pass Criteria	3. DUT: Send SYN,ACK 4. TESTER: Verify that the DUT moves on to ESTABLISHED state
Test Iterations	1. CASE: <wst> = FINWAIT-2 2. CASE: <wst> = CLOSING
Notes	Derived from RFC 1122 s4.2.2.13 p88 Closing a connection "RFC-793 Section 3.5" (MAY)

TCP_CLOSING_06: [established] CLOSE -> FIN

Synopsis	Local user initiates the close, a FIN segment can be constructed and placed on the outgoing segment queue
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state 2. TESTER: Cause DUT side application to issue a CLOSE call 3. DUT: Send a FIN
Pass Criteria	3. DUT: Send a FIN
Test Iterations	
Notes	Derived from RFC 793 s3.5 p38 Closing a Connection (MUST)

TCP_CLOSING_07: [established] CLOSE -> FIN [finwait-1] RECEIVE + DATA -> ACK [finwait-1]

Synopsis	Local user initiates the close, TCP enters the FIN-WAIT-1 state. RECEIVEs are allowed in this state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state 2. TESTER: Cause DUT side application to issue a CLOSE call 3. DUT: DUT sends a FIN and Reaches FIN-WAIT-1 State 4. TESTER: Cause DUT side application to issue a RECEIVE call 5. DUT: Issue a RECEIVE call 6. TESTER: Send some data to DUT 7. DUT: Send ACK of the received data 8. TESTER: Check that DUT receives proper data 9. TESTER: Check that DUT remains in FIN-WAIT-1 state
Pass Criteria	3. DUT: DUT sends a FIN 7. DUT: Send ACK of the received data 9. TESTER: Check that DUT remains in FIN-WAIT-1 state
Test Iterations	
Notes	Derived from RFC 793 s3.5 p38 Closing a Connection (MUST)

TCP_CLOSING_08: [finwait-2] RECEIVE + DATA -> ACK [finwait-2]

Synopsis	TCP enters the FIN-WAIT-2 state. RECEIVEs are allowed in this state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to FIN-WAIT-2 state 2. TESTER: Cause DUT side application to issue a RECEIVE call 3. DUT: Issue a RECEIVE call 4. TESTER: Send some data to DUT 5. DUT: Send ACK of the received data 6. TESTER: Check that DUT receives proper data 7. TESTER: Check that DUT remains in FIN-WAIT-2 state
Pass Criteria	3. DUT: Issue a RECEIVE call 5. DUT: Send ACK of the received data 6. TESTER: Check that DUT receives proper data 7. TESTER: Check that DUT remains in FIN-WAIT-2 state
Test Iterations	
Notes	Derived from RFC 793 s3.5 p38 Closing a Connection (MUST)

TCP_CLOSING_09: [established] FIN -> [close_wait]

Synopsis	If an unsolicited FIN arrives from the network TCP enters the CLOSE_WAIT state. TCP can send any remaining data.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state 2. TESTER: Send a FIN 3. DUT: Send ACK for the received FIN 4. TESTER: Cause the DUT side application to send some data 5. DUT: Send data 6. TESTER: Check that DUT sent proper data 7. TESTER: Check that DUT remains in CLOSE_WAIT state
Pass Criteria	3. DUT: Send ACK for the received FIN 6. TESTER: Check that DUT sent proper data 7. TESTER: Check that DUT remains in CLOSE_WAIT state
Test Iterations	
Notes	Derived from RFC 793 s3.5 p38 Closing a Connection (MUST)

TCP_CLOSING_10: [finwait-1] RECEIVE + RST -> connection reset

Synopsis	TCP, FINWAIT-1 state MUST inform the application in case of aborting from remote site
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT: Move on to FINWAIT-1 state 2. TESTER: Cause the application on the DUT-side to issue a RECEIVE call 3. TESTER: Send a RST 4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
Pass Criteria	4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.13 p87 Closing a connection "RFC-793 Section 3.5" (MUST)

TCP_CLOSING_11: [finwait-2] RECEIVE + RST -> connection reset

Synopsis	TCP, FINWAIT-2 state MUST inform the application in case of aborting from remote site
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT: Move on to FINWAIT-2 state 2. TESTER: Cause the application on the DUT-side to issue a RECEIVE call 3. TESTER: Send a RST 4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
Pass Criteria	4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.13 p87 Closing a connection "RFC-793 Section 3.5" (MUST)

TCP_CLOSING_12: [close-wait] RECEIVE + RST -> connection reset

Synopsis	TCP, in CLOSE-WAIT state MUST inform the application in case of aborting from remote site
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT: Move on to CLOSE-WAIT state 2. TESTER: Cause the application on the DUT-side to issue a RECEIVE call 3. TESTER: Send a RST 4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
Pass Criteria	4. TESTER: Verify that the receiving application receives the signal that \"connection reset\" has occurred from the remote site
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.13 p87 Closing a connection "RFC-793 Section 3.5" (MUST)

TCP_CLOSING_13: [closed] RST -> [closed]

Synopsis	TCP in a CLOSED state, MUST ignore a RST control message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a RST control message 2. DUT: Do not send any response 3. TESTER: Verify that the DUT remains in CLOSED state
Pass Criteria	2. DUT: Do not send any response 3. TESTER: Verify that the DUT remains in CLOSED state
Test Iterations	
Notes	Derived from RFC 793 s3.9 p65 Event Processing (MUST)

4.9.6.12 Processing of TCP MSS, End of Option List, and No-Operation Options

TCP_MSS_OPTIONS_01: Illegal option length for MSS in a SYN segment

Synopsis	TCP MUST be prepared to handle an illegal option length for MSS, in a SYN segment, without crashing
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state at a <wnp> 2. TESTER: Send a SYN with MSS option having illegal length, ilen 3. DUT: Send a TCP Packet (indicating DUT has NOT crashed) 4. TESTER: Verify that the DUT remains in the LISTEN state
Pass Criteria	3. DUT: Send a TCP Packet (indicating DUT has NOT crashed) 4. TESTER: Verify that the DUT remains in the LISTEN state
Test Iterations	1. CASE: ilen = 0 (less than actual) 2. CASE: ilen = 5 (more than actual)
Notes	Derived from RFC 1122 s4.2.2.5 p85 TCP Options "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_02: No Operation and End of Options List options in SYN segment

Synopsis	TCP MUST be able to receive No Operation and End of Options List options in SYN segment
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state at a <wnp> 2. TESTER: Send a SYN with three No Operation options followed by an End of Options list option 3. DUT: Send SYN,ACK 4. TESTER: Send ACK 5. TESTER: Verify that the DUT is in ESTABLISHED state
Pass Criteria	3. DUT: Send SYN,ACK 5. TESTER: Verify that the DUT is in ESTABLISHED state
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.5 p85 TCP Options "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_03: Unimplemented TCP Option

Synopsis	TCP MUST ignore without error any TCP option it does not implement
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to LISTEN state on a <wnp> 2. TESTER: Send a SYN with an unimplemented TCP option <uopt> 3. DUT: Send SYN,ACK 4. TESTER: Send ACK 5. TESTER: Verify that the DUT is in ESTABLISHED state
Pass Criteria	3. DUT: Send SYN,ACK 5. TESTER: Verify that the DUT is in ESTABLISHED state
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.5 p85 TCP Options "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_04: Illegal option length for MSS in a SYN segment

Synopsis	TCP MUST be prepared to handle an illegal option length for MSS, in a SYN segment; a suggested procedure is to reset the connection
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state at a <wnp> 2. TESTER: Send a SYN with MSS option having illegal length, ilen 3. DUT: Send a RST 4. TESTER: Verify that the DUT remains in the LISTEN state
Pass Criteria	3. DUT: Send a RST 4. TESTER: Verify that the DUT remains in the LISTEN state
Test Iterations	1. CASE: ilen = 0 (less than actual) 2. CASE: ilen = 5 (more than actual)
Notes	Derived from RFC 1122 s4.2.2.5 p85 TCP Options "RFC-793 Section 3.1" (MAY)

TCP_MSS_OPTIONS_05: Illegal option length for MSS in a SYN-ACK segment

Synopsis	TCP MUST be prepared to handle an illegal option length for MSS, in a SYN-ACK segment, without crashing
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN 3. TESTER: Send SYN,ACK with MSS option having illegal length, ilen 4. DUT: Send a TCP Packet (indicating DUT has NOT crashed)
Pass Criteria	2. DUT: Send a SYN 4. DUT: Send a TCP Packet (indicating DUT has NOT crashed)
Test Iterations	1. CASE: ilen = 0 (less than actual) 2. CASE: ilen = 5 (more than actual)
Notes	Derived from RFC 1122 s4.2.2.5 p85 TCP Options "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_06: MSS option in SYN segment

Synopsis	TCP MUST be able to receive MSS option in SYN segment and calculate the effective send segment size appropriately
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to LISTEN state 2. TESTER: Send a SYN with MSS option having a value Mv 3. DUT: Send SYN,ACK 4. TESTER: Send ACK 5. TESTER: Cause the application to issue a SEND request for data with size at least max (MSS) 6. DUT: Send data segment(s) 7. TESTER: Verify that the first received segment has size that is min(MSS)
Pass Criteria	3. DUT: Send SYN,ACK 6. DUT: Send data segment(s) 7. TESTER: Verify that the first received segment has size that is min(MSS)
Test Iterations	1. CASE: Mv is smaller than MSS of DUT 2. CASE: Mv is larger than MSS of DUT
Notes	Derived from RFC 1122 s4.2.2.6 p85 Maximum Segment Size Option "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_07: Default send MSS of 536

Synopsis	TCP MUST assume a default send MSS of 536 if MSS option is not received at connection setup
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send SYN,ACK 4. TESTER: Send ACK with MSS option having a value smaller than min(transport_layer_max_send_msg_size-20, 536) 5. TESTER: Cause the application to issue a SEND request for data with size just higher than min(transport_layer_max_send_msg_size-20, 536) 6. DUT: Send a data segment 7. TESTER: Verify that the received segment has size equal to min(transport_layer_max_send_msg_size-20, 536)
Pass Criteria	3. DUT: Send SYN,ACK 6. DUT: Send a data segment 7. TESTER: Verify that the received segment has size equal to min(transport_layer_max_send_msg_size-20, 536)
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.6 p85 Maximum Segment Size Option "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_08: MSS value default and changing

Synopsis	SendMSS is the MSS value received from the remote host, or the default 536 if no MSS option is received.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move to ESTABLISHED state 2. TESTER: Send a data segment with MSS option having a value smaller than the min(transport_layer_max_send_msg_size-20, 536) 3. TESTER: Cause the application to issue a SEND request for data with size min(transport_layer_max_send_msg_size-20, 536) 4. DUT: Send a data segment 5. TESTER: Verify that the received segment has size equal to min(transport_layer_max_send_msg_size-20, 536)
Pass Criteria	4. DUT: Send a data segment 5. TESTER: Verify that the received segment has size equal to min(transport_layer_max_send_msg_size-20, 536)
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.6 p85 Maximum Segment Size Option "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_09: MSS option in SYN ACK segment

Synopsis	TCP MUST be able to receive MSS option in SYN,ACK segment and calculate the effective send segment size appropriately
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an OPEN call 2. DUT: Send a SYN 3. TESTER: Send SYN,ACK with MSS option having a value Mv 4. DUT: Send ACK 5. TESTER: Cause the application to issue a send request for data with size at least max(transport_layer_max_send_msg_size-20, MSS) 6. DUT: Send data segment(s) 7. TESTER: Verify that the first received segment has size that is min(transport_layer_max_send_msg_size-20, MSS)
Pass Criteria	2. DUT: Send a SYN 4. DUT: Send ACK 6. DUT: Send data segment(s) 7. TESTER: Verify that the first received segment has size that is min(transport_layer_max_send_msg_size-20, MSS)
Test Iterations	1. CASE: Mv is smaller than (transport_layer_max_send_msg_size-20) of DUT 2. CASE: Mv is larger than (transport_layer_max_send_msg_size-20) of DUT
Notes	Derived from RFC 1122 s4.2.2.5 p85 TCP Options "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_10: MSS option is not received

Synopsis	If an MSS option is not received at connection setup, TCP MUST assume a default send MSS of 536
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a SYN without MSS option 2. DUT: Send SYN,ACK 3. TESTER: Send ACK 4. TESTER: Cause the application on the DUT-side to issue a send request for data of size more than 536 bytes 5. DUT: Send data segment(s) 6. TESTER: Verify that the first received data segment has 536 bytes Note: It may happen that the send mss of DUT is also 536 bytes. In order to verify that DUT is actually dividing the data into segments based on the default mss value of TESTER, Transport Layer Maximum Send Segment Size of DUT must be greater than 556. Therefore Transport Layer Maximum Send Segment Size of DUT is compared with a value 600(greater than 556) in this test case.The value 600 includes max ipoptions size(if any)
Pass Criteria	2. DUT: Send SYN,ACK 5. DUT: Send data segment(s) 6. TESTER: Verify that the first received data segment has 536 bytes
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.6 p85 Maximum Segment Size Option "RFC-793 Section 3.1" (MUST)

TCP_MSS_OPTIONS_11: Sending the MSS option

Synopsis	TCP MUST implement sending the MSS option
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause an application on the DUT-side to issue an OPEN call 2. DUT: Send a SYN 3. TESTER: Verify that received SYN contains MSS option
Pass Criteria	2. DUT: Send a SYN 3. TESTER: Verify that received SYN contains MSS option
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.6 p85 Maximum Segment Size Option "RFC-793 Section 3.1" Control Protocol (TCP) (MUST)

TCP_MSS_OPTIONS_12: MSS option in every SYN segment differs default

Synopsis	TCP SHOULD send MSS option in every SYN segment when its receive MSS differs from the default 536
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Receive MSS of the DUT is different from the default 2. TESTER: Cause the application on the DUT-side to issue an active OPEN call 3. DUT: Send a SYN with an MSS different from the default 536
Pass Criteria	3. DUT: Send a SYN with an MSS different from the default 536
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.6 p85 Maximum Segment Size Option "RFC-793 Section 3.1" (SHOULD)

TCP_MSS_OPTIONS_13: MSS option in every SYN segment same default

Synopsis	TCP MAY send MSS option in every SYN segment even when its receive MSS value is same as the default 536
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN which may contain MSS option having a value of 536
Pass Criteria	2. DUT: Send a SYN which may contain MSS option having a value of 536
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.6 p85 Maximum Segment Size Option "RFC-793 Section 3.1" (MAY)

TCP_MSS_OPTIONS_14: Handle an illegal option length for MSS in a SYN ACK segment

Synopsis	TCP MUST be prepared to handle an illegal option length for MSS, in a SYN,ACK segment; a suggested procedure is to reset the connection
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN 3. TESTER: Send SYN,ACK with MSS option having illegal length, ilen 4. DUT: Send a RST
Pass Criteria	2. DUT: Send a SYN 4. DUT: Send a RST
Test Iterations	1. CASE: ilen = 0 (less than actual) 2. CASE: ilen = 5 (more than actual)
Notes	Derived from RFC 1122 s4.2.2.5 p85 TCP Options "RFC-793 Section 3.1" (MAY)

4.9.6.13 Processing Out of Order Segments and Delayed ACKs

TCP_OUT_OF_ORDER_01: Timing full-sized segment

Synopsis	A full-sized segment MUST be acknowledged within a time of 0.5 sec
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state at a <wnp> 2. TESTER: Send a full-sized segment 3. DUT: Send an ACK after some time 4. TESTER: Verify that the delay in receiving the ACK (subtracting the average round-trip-time) is less than 0.5 sec
Pass Criteria	3. DUT: Send an ACK after some time 4. TESTER: Verify that the delay in receiving the ACK (subtracting the average round-trip-time) is less than 0.5 sec
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.2 p96 When to Send an ACK Segment (MUST)

TCP_OUT_OF_ORDER_02: Timing delayed ACK

Synopsis	TCP SHOULD implement a delayed ACK, but the delay MUST be less than 0.5 sec
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to full-window operation in ESTABLISHED state at a <wnp> 2. TESTER: Send two data segments consecutively without any delay 3. DUT: Send a single ACK for the data segments after a delay 4. TESTER: Verify that the delay (subtracting the average round trip time) is less than 0.5 sec
Pass Criteria	3. DUT: Send a single ACK for the data segments after a delay 4. TESTER: Verify that the delay (subtracting the average round trip time) is less than 0.5 sec
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.2 p96 When to Send an ACK Segment (MUST)

TCP_OUT_OF_ORDER_03: Queuing out-of-order segments

Synopsis	TCP SHOULD be capable of queuing out-of-order segments
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state at a <wnp> 2. TESTER: Send a data segment 3. DUT: Send the ACK for the data segment 4. TESTER: Send <nss> number of data segments leaving a gap in SEQ numbers (as if a segment is skipped) at the beginning 5. DUT: Do not send ACKs for these segments 6. TESTER: Send a segment having data for the missing SEQ numbers 7. DUT: Send ACK for all queued data seg(s) including the latest one
Pass Criteria	3. DUT: Send the ACK for the data segment 5. DUT: Do not send ACKs for these segments 7. DUT: Send ACK for all queued data seg(s) including the latest one
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.20 p93 Event Processing "RFC-793 Section 3.9" (SHOULD)

TCP_OUT_OF_ORDER_04: Acknowledging RCV.NXT for valid out-of-order data segments

Synopsis	TCP MAY send an ACK segment acknowledging RCV.NXT for valid out-of-order data segments
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state at a <wnp> 2. TESTER: Send a data segment 3. DUT: Send the ACK for the segment 4. TESTER: Send a data segment that is in the receive window of DUT but not at left window edge 5. DUT: Send an ACK acknowledging RCV.NXT
Pass Criteria	3. DUT: Send the ACK for the segment 5. DUT: Send an ACK acknowledging RCV.NXT
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.21 p94 Acknowledging Queued Segments "RFC-793"Section 3.9 (MAY)

TCP_OUT_OF_ORDER_05: Stream of full-sized segments

Synopsis	In a stream of full-sized segments there SHOULD be an ACK for at least every second segment
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to full-window operation in ESTABLISHED state 2. TESTER: Send some number of full-sized segments to fill the DUTs receive window buffer 3. DUT: Send ACKs for all the data segments 4. TESTER: Verify that for every even number of data segments sent the number of ACKs received is at least half of that number
Pass Criteria	3. DUT: Send ACKs for all the data segments 4. TESTER: Verify that for every even number of data segments sent the number of ACKs received is at least half of that number
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.2 p96 When to Send an ACK Segment (SHOULD)

TCP_OUT_OF_ORDER_08: Queuing out-of-order segments

Synopsis	TCP SHOULD be capable of queuing out-of-order segments
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Send a data segment 3. DUT: Send the ACK for the data segment 4. TESTER: Send <nss> number of data segments leaving a gap in SEQ numbers (as if a segment is skipped) at the beginning 5. DUT: Do not send ACKs for these segments 6. TESTER: Send a segment having data for the missing SEQ numbers 7. DUT: Send ACK for all queued data seg(s) including the latest one
Pass Criteria	3. DUT: Send the ACK for the data segment 5. DUT: Do not send ACKs for these segments 7. DUT: Send ACK for all queued data seg(s) including the latest one
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.20 p93 Event Processing "RFC-793 Section 3.9" (SHOULD)

4.9.6.14 SWS Avoidance Algorithm

TCP_AVOIDANCE_1: SWS Avoidance Algorithm less MSS

Synopsis	TCP MUST include an SWS avoidance algorithm in the receiver (Note: read some data of size less than MSS)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Continue sending data until DUT sends ACK with window 0 3. TESTER: Cause DUT to read some data of size less than MSS 4. TESTER: Send 0 window probe 5. DUT: Send ACK containing zero window
Pass Criteria	5. DUT: Send ACK containing zero window
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.3 p97 When to Send a Window Update (MUST)

TCP_AVOIDANCE_2: SWS Avoidance Algorithm greater MSS

Synopsis	TCP MUST include an SWS avoidance algorithm in the receiver (Note:read some data of size greater or equal to MSS. TESTER follow the third step in a sequence. TESTER cause DUT to receive data until DUT read data of size greater or equal to MSS.)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Continue sending data until DUT sends ACK with window 0 3. TESTER: Cause DUT to read some data of size greater or equal to MSS 4. TESTER: Send 0 window probe 5. DUT: Send ACK containing non-zero window 6. TESTER: Verify that non zero window size is greater or equal to MSS
Pass Criteria	5. DUT: Send ACK containing non-zero window 6. TESTER: Verify that non zero window size is greater or equal to MSS
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.3 p97 When to Send a Window Update (MUST)

4.9.6.15 Retransmission Timeout

TCP_RETRANSMISSION_TO_01: Retransmitted packet (differs from the original packet)

Synopsis	If a retransmitted packet differs from the original packet in the window value, then the same IP identification field MUST NOT be used
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause an application on the DUT-side to issue a data SEND call 3. DUT: Send the data segment 4. TESTER: Send data packet of size MSS to DUT 5. TESTER: Do not send ACK for the received data and cause the retransmission timer on the DUT side to expire 6. DUT: Retransmit the data segment with changed window value 7. TESTER: Verify that the IP identification fields in the two received packets are different
Pass Criteria	3. DUT: Send the data segment 6. DUT: Retransmit the data segment with changed window value 7. TESTER: Verify that the IP identification fields in the two received packets are different
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.15 p90 Retransmission Timeout "RFC-793 Section 3.7 page 41" (MUST)

TCP_RETRANSMISSION_TO_02: Retransmitted packet (differs from the original packet in ack field)

Synopsis	If a retransmitted packet differs from the original packet in the acknowledgement field, the same IP identification field MUST NOT be used
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause an application on the DUT-side to issue a data SEND request 3. DUT: Send the data segment 4. TESTER: Do not send the ACK 5. TESTER: Send a data segment so that it reaches on the DUT-side just before the retransmission timer expires on the DUT-side 6. DUT: Retransmit the data segment with ACK for the data segment received from the tester 7. TESTER: Verify that the IP identification fields in the two received packets are different
Pass Criteria	3. DUT: Send the data segment 6. DUT: Retransmit the data segment with ACK for the data segment received from the tester 7. TESTER: Verify that the IP identification fields in the two received packets are different
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.15 p90 Retransmission Timeout "RFC-793 Section 3.7 page 41" (MUST)

TCP_RETRANSMISSION_TO_03: Karn's algorithm

Synopsis	TCP MUST follow the Karn's algorithm
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause an application on the DUT-side to issue a data SEND request 3. DUT: Send the data segment 4. TESTER: Do not send the ACK and cause the retransmission timer on the DUT side to expire 5. DUT: Send the data segment once more 6. TESTER: Note the retransmission time (RTO) 7. TESTER: Send an ACK just before the RTO occurs on DUT 8. TESTER: Cause the application to issue SEND request for one more data segment 9. DUT: Send the data segment 10. TESTER: Do not send ACK 11. DUT: Retransmit the data segment 12. TESTER: Verify that the retransmission timer value is twice the value of RTO noted earlier(exponential backoff only)
Pass Criteria	3. DUT: Send the data segment 5. DUT: Send the data segment once more 9. DUT: Send the data segment 11. DUT: Retransmit the data segment 12. TESTER: Verify that the retransmission timer value is twice the value of RTO noted earlier(exponential backoff only)
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.1 p95 Retransmission Timeout Calculation (MUST)

TCP_RETRANSMISSION_TO_04: Exponential backoff RTO Data

Synopsis	TCP MUST include \"exponential backoff\" (check that it increases) for successive RTO values for sending data segments
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause an application on the DUT-side to issue a data SEND request 3. DUT: Send the data segment 4. TESTER: Do not send ACK 5. DUT: Retransmit the data segment after a timeout 6. TESTER: Do not send ACK and verify that RTO retransmission interval is increasing at a fast (at least more than linear) rate
Pass Criteria	2. DUT: Send a SYN 4. DUT: Retransmit the SYN segment after a timeout 5. TESTER: Verify that RTO retransmission interval is increasing at a fast (at least more than linear) rate
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.1 p95 Retransmission Timeout Calculation (MUST)

TCP_RETRANSMISSION_TO_05: Exponential backoff RTO SYN

Synopsis	TCP MUST include \"exponential backoff\" (check that it increases) for successive RTO values for sending SYN segments
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN 3. TESTER: Do not send SYN,ACK 4. DUT: Retransmit the SYN segment after a timeout 5. TESTER: Do not send SYN,ACK and verify that RTO retransmission interval is increasing at a fast (at least more than linear) rate
Pass Criteria	2. DUT: Send a SYN 4. DUT: Retransmit the SYN segment after a timeout 5. TESTER: Verify that RTO retransmission interval is increasing at a fast (at least more than linear) rate
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.1 p95 Retransmission Timeout Calculation (MUST)

TCP_RETRANSMISSION_TO_06: Initial RTO 3

Synopsis	TCP SHOULD use RTO = 3 sec initially
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN 3. TESTER: Do not send SYN,ACK 4. DUT: Retransmit the SYN segment after a timeout 5. TESTER: Verify that the DUT used a timeout of 3 sec
Pass Criteria	2. DUT: Send a SYN 4. DUT: Retransmit the SYN segment after a timeout 5. TESTER: Verify that the DUT used a timeout of 3 sec
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.1 p96 Retransmission Timeout Calculation (SHOULD)

TCP_RETRANSMISSION_TO_07: Initial RTT 0

Synopsis	TCP SHOULD use RTT = zero sec initially
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND call for some data 3. DUT: Send the data segment 4. TESTER: Do not send ACK 5. DUT: Retransmit the data segment 6. TESTER: Verify that the RTO used has considered RTT = zero sec with some pre-defined alpha, beta, LBOUND and UBOUND values
Pass Criteria	3. DUT: Send the data segment 5. DUT: Retransmit the data segment 6. TESTER: Verify that the RTO used has considered RTT = zero sec with some pre-defined alpha, beta, LBOUND and UBOUND values
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.1 p96 Retransmission Timeout Calculation (SHOULD)

TCP_RETRANSMISSION_TO_08: 2*MSL of RTO for data

Synopsis	TCP SHOULD use an upper bound of 2*MSL of RTO for data segments
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND call call some data 3. DUT : Send the data segment 4. TESTER: Do not send ACK 5. DUT: Retransmit the data segment 6. TESTER: Do not send any ACK and verify that the DUT repeatedly retransmits with increasing delays till the retransmit timeout reaches 2*MSL after which the RTO gets fixed at that value
Pass Criteria	3. DUT : Send the data segment 5. DUT: Retransmit the data segment 6. TESTER: Verify that the DUT repeatedly retransmits with increasing delays till the retransmit timeout reaches 2*MSL after which the RTO gets fixed at that value
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.1 p96 Retransmission Timeout Calculation (SHOULD)

TCP_RETRANSMISSION_TO_09: 2*MSL of RTO for SYN

Synopsis	TCP SHOULD use an upper bound of 2*MSL of RTO for SYN segment
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN 3. TESTER: Do not send SYN,ACK 4. DUT: Retransmit the SYN segment 5. TESTER: Do not send any SYN,ACK and verify that the DUT repeatedly retransmits with increasing delays till the retransmit timeout reaches 2*MSL after which the RTO gets fixed at that value
Pass Criteria	4. DUT: Send only <cwnd-ini> segments
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.1 p96 Retransmission Timeout Calculation (SHOULD)

TCP_RETRANSMISSION_TO_10: Same IP identification field

Synopsis	If a retransmitted packet is identical to the original packet, then the same IP identification field MAY be used
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause an application on the DUT-side to issue a SEND call 3. DUT: Send the data segment 4. TESTER: Do not send ACK and allow the retransmit timer on the DUT-side to expire 5. DUT: Send the data segment once more 6. TESTER: Verify that the IP identification fields in the two received packets are the same
Pass Criteria	3. DUT: Send the data segment 5. DUT: Send the data segment once more 6. TESTER: Verify that the IP identification fields in the two received packets are the same
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.15 p90 Retransmission Timeout "RFC-793 Section 3.7 page 41" (MAY)

4.9.6.16 Generation of Zero Window Probes

TCP_PROBING_WINDOWS_01: Indefinitely closed window

Synopsis	A TCP MAY keep its offered receive window closed indefinitely
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Send enough data segments to fill the receiver window of DUT where the receiving application is not asked to extract any data 3. DUT: Send ACKs with the last window update shown to be zero 4. TESTER: Verify that even after a long time (2*MSL) the DUT remains in the ESTABLISHED state
Pass Criteria	3. DUT: Send ACKs with the last window update shown to be zero 4. TESTER: Verify that even after a long time (2*MSL) the DUT remains in the ESTABLISHED state
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.17 p92 Probing Zero Windows "RFC-793 Section 3.7, page 42" (MAY)

TCP_PROBING_WINDOWS_02: windows size unsigned number

Synopsis	The windows size MUST be treated as an unsigned number
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to SYN-RCVD state 2. TESTER: Send ACK segment with a window size having the MSB set 3. TESTER: Cause the application on the DUT-side to issue a SEND request for a data segment 4. DUT: Send the data segment
Pass Criteria	4. DUT: Send the data segment
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.3 p83 Window Size "RFC-793 Section 3.1" (MUST)

TCP_PROBING_WINDOWS_03: Window shrinking

Synopsis	A sending TCP MUST be robust against window shrinking, which may cause the \"useable window\" to become negative
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue one data SEND request 3. DUT: Send the data segment 4. TESTER: Send the ACK 5. TESTER: Cause the application on the DUT-side to issue two more SEND calls for data segments 6. DUT: Send the data segments 7. TESTER: Send ACK for the first segment with an updated window value of zero 8. TESTER: Cause the application on the DUT-side to issue a SEND request for a data segment 9. DUT: Do not send the segment as the \"useable window\" is negative
Pass Criteria	3. DUT: Send the data segment 6. DUT: Send the data segments 9. DUT: Do not send the segment as the \"useable window\" is negative
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.16 p91 Managing the Window "RFC-793 Section 3.7 page 41" (MUST)

TCP_PROBING_WINDOWS_04: Open connection probes ACK

Synopsis	Even if the receive window is closed indefinitely, the sending TCP MUST allow the connection to stay open as long as probes are acknowledged
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND request for data segment 3. DUT: Send the data segment 4. TESTER: Send ACK declaring a zero receive window 5. TESTER: Cause the application on the DUT-side to issue another SEND request for data segment 6. DUT: Send a zero window probe 7. TESTER: Acknowledge the probe maintaining zero window 8. DUT: Keep on sending the zero window probes, staying in the ESTABLISHED state, as long as the tester acknowledges each of them at every reception
Pass Criteria	3. DUT: Send the data segment 6. DUT: Send a zero window probe 8. DUT: Keep on sending the zero window probes, staying in the ESTABLISHED state, as long as the tester acknowledges each of them at every reception
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.17 p92 Probing Zero Windows "RFC-793 Section 3.7, page 42" Control Protocol (TCP) (MUST)

TCP_PROBING_WINDOWS_05: First zero window probe

Synopsis	TCP SHOULD send the first zero window probe when the receiver window size remains zero for the retransmission timeout period
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND request for data segments 3. DUT: Send the data segment 4. TESTER: Send ACK declaring a zero window 5. TESTER: Cause the application on the DUT-side to issue another SEND request for data segment 6. DUT: Send a zero window probe after some delay 7. TESTER: Verify that the probe has been sent by the DUT after waiting for at least the retransmission timeout
Pass Criteria	3. DUT: Send the data segment 6. DUT: Send a zero window probe after some delay 7. TESTER: Verify that the probe has been sent by the DUT after waiting for at least the retransmission timeout
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.17 p92 Probing Zero Windows "RFC-793 Section 3.7 page 42" (SHOULD)

TCP_PROBING_WINDOWS_06: Increase interval zero window probes

Synopsis	TCP SHOULD increase exponentially the interval between successive zero window probes
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND request for data segments 3. DUT: Send the data segment 4. TESTER: Send ACK declaring a zero window 5. TESTER: Cause the application on the DUT-side to issue another SEND request for data segment 6. DUT: Keep sending zero window probes after some delays 7. TESTER: Verify that the delays between successive probes increases continually
Pass Criteria	3. DUT: Send the data segment 6. DUT: Keep sending zero window probes after some delays 7. TESTER: Verify that the delays between successive probes increases continually
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.17 p92 Probing Zero Windows "RFC-793 Section 3.7 page 42" (SHOULD)

4.9.6.17 Nagle Algorithm

TCP_NAGLE_01: Configurable TTL value

Synopsis	The TTL value used to send TCP segments MUST be configurable
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Change the TTL of DUT side application 3. TESTER: Cause the application on the DUT-side to issue a SEND request for data segment 4. DUT: Send the data segment 5. TESTER: Verify that the IP datagram received contains the specified TTL value
Pass Criteria	4. DUT: Send the data segment 5. TESTER: Verify that the IP datagram received contains the specified TTL value
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.19 p93 Time to Live "RFC-793 Section 3.9, page 52" (MUST)

TCP_NAGLE_02: Buffer all the user data until ACK

Synopsis	TCP SHOULD implement the Nagle Algorithm, i.e., buffer all the user data, regardless of PSH, until the outstanding data has been acknowledged
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND request for a segment 3. DUT: Send the data segment 4. TESTER: Do not acknowledge the data 5. TESTER: Cause the application to issue another SEND request with the PSH bit (if possible to specify) 6. DUT: Do not send the data segment 7. TESTER: Acknowledge the previous data 8. DUT: Send the latter data segment
Pass Criteria	3. DUT: Send the data segment 6. DUT: Do not send the data segment 8. DUT: Send the latter data segment
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.4 p98 When to Send Data (SHOULD)

TCP_NAGLE_03: Buffer all the user data until full-sized segment

Synopsis	TCP SHOULD implement the Nagle Algorithm, i.e., it buffers all user data, regardless of PSH bit, until the TCP can send a full-sized segment
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND call for a data segment 3. DUT: Send the data segment 4. TESTER: Do not acknowledge the data 5. TESTER: Cause the application to issue another send request of small size <ssz> with the PSH bit (if possible to specify) 6. DUT: Do not send the data segment 7. TESTER: Cause the application to issue one more SEND request with the aggregate data size equal to effective send MSS 8. DUT: Send the aggregate data segment
Pass Criteria	3. DUT: Send the data segment 6. DUT: Do not send the data segment 8. DUT: Send the aggregate data segment
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.4 p98 When to Send Data (SHOULD)

TCP_NAGLE_04: Disabling on application

Synopsis	There MUST be a way for an application to disable the Nagle Algorithm on an individual connection
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a disabling request of usage of Nagle Algorithm 3. TESTER: Cause the application to issue a SEND request for a data segment 4. DUT: Send the data segment 5. TESTER: Do not acknowledge the data 6. TESTER: Cause the application to issue another SEND request 7. DUT: Send the data segment
Pass Criteria	4. DUT: Send the data segment 7. DUT: Send the data segment
Test Iterations	
Notes	Derived from RFC 1122 s4.2.3.4 p98 When to Send Data (MUST)

4.9.6.18 Use of the TCP PUSH Flag

TCP_FLAGS_PUSH_01: No Push flag implemented

Synopsis	If PUSH flags are not implemented, then the sending TCP MUST NOT buffer data indefinitely
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to SEND a small segment of size <ssz> 3. DUT: Send data within <tmx> time
Pass Criteria	3. DUT: Send data within <tmx> time
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.2 p83 Use of Push "RFC-793 Section 2.8" (MUST)

TCP_FLAGS_PUSH_02: Aggregate data

Synopsis	TCP MAY aggregate data requested by an application for sending until accumulated data exceeds effective send MSS
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue 2 SEND calls 3. DUT: Do not send data 4. TESTER: Cause the application to issue a SEND call so that the accumulated data just exceeds the effective send MSS 5. DUT: Send a data segment with effective send MSS size
Pass Criteria	3. DUT: Do not send data 5. DUT: Send a data segment with effective send MSS size
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.2 p82 Use of Push "RFC-793 Section 2.8" (MAY)

4.9.6.19 Use of the Urgent Pointer

TCP_URGENT_PTR_01: Pointing to the sequence number

Synopsis	In the time of transmitting, TCP MUST keep the urgent pointer pointing to the sequence number of the LAST octet in a sequence of urgent data
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND request for a data segment setting both the PUSH and URGENT flags 3. DUT: Send the data segment with URG flag bit set and the urgent pointer value providing the offset of the last data byte
Pass Criteria	3. DUT: Send the data segment with URG flag bit set and the urgent pointer value providing the offset of the last data byte
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.4 p84 Urgent Pointer "RFC-793 Section 3.1" (MUST)

TCP_URGENT_PTR_02: Sequence of urgent data of any length

Synopsis	TCP MUST support a sequence of urgent data of any length for transmission
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to full-window operation in ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue SEND request for data of size much more than MSS (say 10 times) with urgent flag set 3. DUT: Send data segments each of size MSS and containing the URG flag bit set with urgent pointer pointing to the sequence number of last byte of URG data, indicated as an offset from current SEQ
Pass Criteria	3. DUT: Send data segments each of size MSS and containing the URG flag bit set with urgent pointer pointing to the sequence number of last byte of URG data, indicated as an offset from current SEQ
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.4 p84 Urgent Pointer "RFC-793 Section 3.1" (MUST)

TCP_URGENT_PTR_03: Sequence number of the octet following the urgent data

Synopsis	The urgent pointer points to the sequence number of the octet following the urgent data.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue a SEND request for a data segment setting both the PUSH and URGENT flags 3. DUT: Send the data segment with URG flag bit set and the urgent pointer value providing the offset of the (last+1) data byte
Pass Criteria	3. DUT: Send the data segment with URG flag bit set and the urgent pointer value providing the offset of the (last+1) data byte
Test Iterations	
Notes	Derived from RFC-793 Section 3.1 p17 (MUST)

TCP_URGENT_PTR_04: Data following the urgent pointer not same buffer

Synopsis	The data following the urgent pointer (non-urgent data) MUST NOT be delivered to the user in the same buffer with preceding urgent data
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Send a data segment with URG flag set and urgent pointer containing the offset for the SEQ number of a byte that is not the last byte of this data segment 3. DUT: Receive the data segment and issue a signal to the application that urgent data has arrived 4. TESTER: Cause the application on the DUT-side to issue a RECEIVE call with a data buffer having size equal to the size of the incoming data segment 5. DUT: Return the RECEIVE call putting only the urgent data
Pass Criteria	3. DUT: Receive the data segment and issue a signal to the application that urgent data has arrived 5. DUT: Return the RECEIVE call putting only the urgent data
Test Iterations	
Notes	Derived from RFC 793 s3.7 p48 Interfaces (MUST)

TCP_URGENT_PTR_05: Sequence of urgent data of any length RFC 793

Synopsis	TCP MUST support a sequence of urgent data of any length for transmission (Note : This test supports Urgent Pointer Mechanism of RFC 793)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to full-window operation in ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to issue SEND request for data of size much more than MSS (say 10 times) with urgent flag set 3. DUT: Send data segments each of size MSS and containing the URG flag bit set with urgent pointer pointing to the sequence number of the octet following the URG data, indicated as an offset from current SEQ
Pass Criteria	3. DUT: Send data segments each of size MSS and containing the URG flag bit set with urgent pointer pointing to the sequence number of the octet following the URG data, indicated as an offset from current SEQ
Test Iterations	
Notes	Derived from RFC 1122 s4.2.2.4 p84 Urgent Pointer "RFC-793 Section 3.1" (MUST)

4.9.6.20 Slow Start and Congestion Avoidance

TCP_SLOWSTART_CONGESTION_01: Slow Start - initial value of cwnd

Synopsis	IW, the initial value of cwnd, MUST be less than or equal to 2*SMSS bytes and MUST NOT be more than 2 segments.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS set to SMALL_MAX_SEG_SIZE and window set to LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segments and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments
Pass Criteria	4. DUT: Send only <cwnd-ini> segments
Test Iterations	
Notes	Derived from RFC 2581 s 3.1 p 4 Slow Start and Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_02: Slow Start - increments cwnd

Synopsis	During slow start, a TCP increments cwnd by at most SMSS bytes for each ACK received that acknowledges new data.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Do not send ACK 8. DUT: Retransmit data packet
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Retransmit data packet
Test Iterations	
Notes	Derived from RFC 2581 s 3.1 p 4 Slow Start and Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_03: Congestion Avoidance - cwnd less than the receiver's advertised window

Synopsis	The TCP output routine never sends more than the minimum of cwnd and the receiver's advertised window. (Note: cwnd is less than the receiver's advertised window)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segments and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Send ACK for all the received segments 8. DUT: Set cwnd to 4*<cwnd-ini>, send 4*<cwnd-ini> segments and wait for their ACK 9. TESTER: Send ACK for all the received segments with window as LARGEST_WINDOW_SIZE 10. DUT: Set cwnd to 8*<cwnd-ini>, send 8*<cwnd-ini> segments and wait for their ACK 11. TESTER: Do not send ACK 12. DUT: Retransmit data packet
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: send 8*<cwnd-ini> segments and wait for their ACK 12. DUT: Retransmit data packet
Notes	Derived from RFC 2001 s2 p3 Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_04: Congestion Avoidance - never sends more than receiver's advertised window

Synopsis	The TCP output routine never sends more than the minimum of cwnd and the receiver's advertised window. (Note: receiver's advertised window is less than the cwnd)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segment 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Send ACK for all the received segments 8. DUT: Set cwnd to 4*<cwnd-ini>, send 4*<cwnd-ini> segments and wait for their ACK 9. TESTER: Send ACK for all the received segments with window as 4 * SMALL_MAX_SEG_SIZE 10. DUT: Set cwnd to 8*<cwnd-ini>, send 4 segments and wait for their ACK 11. TESTER: Do not send ACK 12. DUT: Retransmit data packet
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: send 4*<cwnd-ini> segments and wait for their ACK 10. DUT: send 4 segments and wait for their ACK 12. DUT: Retransmit data packet
Notes	Derived from RFC 2001 s2 p3 Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_05: Congestion Avoidance - one-half of the current window size is saved in ssthresh

Synopsis	When congestion occurs, one-half of the current window size is saved in ssthresh.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segment 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Send ACK for all the received segments 8. DUT: Set cwnd to 4*<cwnd-ini> segment. Send 4*<cwnd-ini> segments and wait for their ACK 9. TESTER: Do not send any ACK for the received segments so that RTO occurs on DUT. This will indicate congestion in the network 10. DUT: Set ssthresh to (cwnd/2)= (2*<cwnd-ini>) and cwnd to <LW> 11. DUT: Retransmit <cwnd> segment 12. TESTER: Send ACK for the retransmitted segment 13. DUT: Send only <cwnd-ini> segments 14. TESTER: Send ACK for the all the received segments 15. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 16. TESTER: Send ACK for all the received segments 17. DUT: Start congestion avoidance. Send (2*<cwnd-ini> + 1) segments because the DUT is performing Congestion Avoidance, and not (4*<cwnd-ini>) segments as would be the case with Slow Start.
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Send 4*<cwnd-ini> segments and wait for their ACK 11. DUT: Retransmit <cwnd> segment 13. DUT: Send only <cwnd-ini> segments 15. DUT: send 2*<cwnd-ini> segments and wait for their ACK

	17. DUT: Start congestion avoidance. Send (2*<cwnd-ini> + 1) segments because the DUT is performing Congestion Avoidance, and not (4*<cwnd-ini>) segments as would be the case with Slow Start.
Test Iterations	
Notes	Derived from RFC 2001 s2 p3 Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_06: Congestion Avoidance - one-half of the current window size but at least 2 segments is saved in ssthresh

Synopsis	When congestion occurs, one-half of the current window size, but at least 2 segments is saved in ssthresh.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segment 5. TESTER: Send ACK for the received segment 6. TESTER: Repeat last two steps till DUT send 2 segments 7. TESTER: Do not send any ACK for the received segments so that RTO occurs on DUT. This will indicate congestion in the network 8. DUT: Set ssthresh to 2 and cwnd to <LW> 9. DUT: Send only <cwnd> segment 10. TESTER: Send ACK for the received segment 11. TESTER: Repeat last two steps till DUT send 2 segments and wait for their ACK 12. TESTER: Send ACK for all the received segments 13. DUT: Start congestion avoidance. Send 3 segments because the DUT is performing Congestion Avoidance, and not 4 segments as would be the case with Slow Start.
Pass Criteria	4. DUT: Send only <cwnd-ini> segment 9. DUT: Send only <cwnd> segment 13. DUT: Start congestion avoidance. Send 3 segments because the DUT is performing Congestion Avoidance, and not 4 segments as would be the case with Slow Start.
Notes	Derived from RFC 2001 s2 p3 Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_07: Congestion Avoidance - cwnd settings upon a timeout

Synopsis	Upon a timeout cwnd MUST be set to no more than the loss window, LW, which equals 1 full-sized segment (regardless of the value of IW).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segment 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Send ACK for all the received segments 8. DUT: Set cwnd to 4*<cwnd-ini> segment. Send 4*<cwnd-ini> segments and wait for their ACK 9. TESTER: Do not send any ACK for the received segments so that RTO occurs on DUT. This will indicate congestion in the network 10. DUT: Set ssthresh to 2 and cwnd to <LW> 11. DUT: Send only <cwnd> segment
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Send 4*<cwnd-ini> segments and wait for their ACK 11. DUT: Send only <cwnd> segment
Test Iterations	
Notes	Derived from RFC 2581 s 3.1 p 5 Slow Start and Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_08: Congestion Avoidance - slow start continues until TCP is halfway

Synopsis	Slow start continues until TCP is halfway to where it was when congestion occurred
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segment 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Send ACK for all the received segments 8. DUT: Set cwnd to 4*<cwnd-ini> segment. Send 4*<cwnd-ini> segments and wait for their ACK 9. TESTER: Do not send any ACK for the received segments so that RTO occurs on DUT. This will indicate congestion in the network 10. DUT: Set ssthresh to (cwnd/2)=2*<cwnd-ini> and cwnd to <LW> 11. DUT: Send only <cwnd> segments 12. TESTER: Send ACK for the received segment 13. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 14. TESTER: Send ACK for all the received segments 15. DUT: Start congestion avoidance by sending 2*<cwnd-ini>+1 segments but not 4*<cwnd-ini>
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 8. DUT: Send 4*<cwnd-ini> segments and wait for their ACK 11. DUT: Send only <cwnd> segments 13. DUT: send 2*<cwnd-ini> segments and wait for their ACK 15. DUT: Start congestion avoidance by sending 2*<cwnd-ini>+1 segments but not 4*<cwnd-ini>
Test Iterations	

Notes	Derived from RFC 2001 s2 p3 Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_09: Congestion Avoidance - Formula

Synopsis	Formula commonly used to update cwnd during congestion avoidance is cwnd += SMSS*SMSS/cwnd executed on every incoming non-duplicate ACK.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segment 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Send ACK for all the received segments 8. TESTER: Repeat the last two steps till DUT sends 4 segments 9. DUT: Set cwnd to 4 segment. Send 4 segments and wait for their ACK 10. TESTER: Do not send any ACK for the received segments so that RTO occurs on DUT. This will indicate congestion in the network 11. DUT: Set ssthresh = (cwnd/2)=2 and cwnd to <LW> 12. DUT: Send only <cwnd> segment 13. TESTER: Send ACK for the received segment 14. TESTER: Repeat last step till DUT sends 2 segments 15. DUT: Set cwnd to 2 (2*SMALL_MAX_SEG_SIZE in bytes) send 2 segments and wait for their ACK 16. TESTER: Send ACK for all the received segments 17. DUT: Start congestion avoidance. Calculate cwnd (in bytes) as cwnd + { (segsize * segsize) / cwnd) } * no_of_ACK_received i.e. (2*SMALL_MAX_SEG_SIZE) + {(SMALL_MAX_SEG_SIZE* SMALL_MAX_SEG_SIZE)/(2*SMALL_MAX_SEG_SIZE)}*2 = (2*SMALL_MAX_SEG_SIZE)+SMALL_MAX_SEG_SIZE (bytes)= 3 segments 18. DUT: Send 3 segments
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK

	9. DUT: Send 4 segments and wait for their ACK 12. DUT: Send only <cwnd> segment 15. DUT: send 2 segments and wait for their ACK 18. DUT: Send 3 segments
Test Iterations	
Notes	Derived from RFC 2581 s 3.1 p 4 Slow Start and Congestion Avoidance (MUST)

TCP_SLOWSTART_CONGESTION_10: Congestion Avoidance - increase in cwnd

Synopsis	The increase in cwnd should be at most one segment each round-trip time (regardless how many ACKs are received in that RTT)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT side application send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segment 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Send ACK for all the received segments 8. TESTER: Repeat the last two steps till DUT sends 4 segments 9. DUT: Set cwnd to 4 segment. Send 4 segments and wait for their ACK 10. TESTER: Do not send any ACK for the received segments so that RTO occurs on DUT. This will indicate congestion in the network 11. DUT: Set ssthresh = (cwnd/2)=2 and cwnd to <LW> 12. DUT: Send only <cwnd> segment 13. TESTER: Send ACK for the received segment 14. TESTER: Repeat last step till DUT sends 2 segments 15. TESTER: Send ACK for all the received segments 16. DUT: Start congestion avoidance, send 3 segments 17. TESTER: Send ACK for all the received segments 18. DUT: Send 4 segments 19. TESTER: Send a single ACK for all the received segments 20. DUT: Send 5 segments
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments and wait for their ACK 9. DUT: Send 4 segments and wait for their ACK 12. DUT: Send only <cwnd> segment 16. DUT: Start congestion avoidance, send 3 segments 18. DUT: Send 4 segments

	20. DUT: Send 5 segments
Test Iterations	
Notes	Derived from RFC 2001 s2 p3 Congestion Avoidance (SHOULD)

4.9.6.21 Fast Retransmit/Fast Recovery

TCP_FAST_RETRANSMIT_01: After receiving 3 duplicate ACKs

Synopsis	After receiving 3 duplicate ACKs, TCP performs a retransmission of missing segment, without waiting for retransmission timer to expire.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE with delay between sending packets, so that the calculated RTO on DUT is large 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT side application send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for all the received segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACK 7. TESTER: Send ACK for all the received segments 8. DUT: Set cwnd to 4*<cwnd-ini>, send 4*<cwnd-ini> segments and wait for their ACK 9. TESTER: On receiving every packet send the last sent ACK in response 10. DUT: On receiving the 3rd duplicate ACK, retransmit the ACKd segment 11. TESTER: Check that DUT retransmits the correct packet without waiting for retransmission timer to expire
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments 10. DUT: Retransmit the segment indicated by the last received ACK 12. DUT: Send 4 segments
Test Iterations	
Notes	Derived from RFC 2581 s3.2 p6 Fast Retransmit/Fast Recovery (MUST)

TCP_FAST_RETRANSMIT_02: When next ACK arrives

Synopsis	When next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This is termed `deflating' the window.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS SMALL_MAX_SEG_SIZE and window LARGE_WINDOW_SIZE with delay between sending packets, so that the calculated RTO on DUT is large 2. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 3. TESTER: Cause the DUT side application send DATA_PACKET_LARGE 4. DUT: Send only <cwnd-ini> segments 5. TESTER: Send ACK for the received segments 6. DUT: Set cwnd to 2*<cwnd-ini> and send 2*<cwnd-ini> segments 7. TESTER: Send ACK for the received segments 8. TESTER: Repeat the last two steps till DUT sends 8 segments 9. TESTER: Send the last sent ACK 3 times 10. DUT: On receivng the 3 duplicate ACKs, set ssthresh to one-half the current cwnd. Retransmit the segment indicated by the last received ACK and set cwnd to ssthresh plus 3 (in segments) 11. TESTER: On receiving the retransmitted segment, send an ACK for all the segments received so far, causing the DUT to set cwnd to the value of ssthresh (4). 12. DUT: Send 4 segments 13. TESTER: Send ACK for the received segments 14. DUT: Send 5 segments
Pass Criteria	4. DUT: Send only <cwnd-ini> segments 6. DUT: send 2*<cwnd-ini> segments 10. DUT: Retransmit the segment indicated by the last received ACK 12. DUT: Send 4 segments 14. DUT: Send 5 segments
Test Iterations	
Notes	Derived from RFC 2581 s3.2 p7 Fast Retransmit/Fast Recovery (MUST)

4.9.6.22 Host Specification and MSS

TCP_HOST_SPEC_01: Host receiving Datagram Too Big message

Synopsis	Host receiving Datagram Too Big message MUST reduce its estimate of MTU for relevant path, based on value of Next-Hop MTU field in the message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to set DF bit 6. TESTER: Cause the application on the DUT-side to send large data 7. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 9. DUT: Send packet containing data 10. TESTER: Check data length does not exceed (SMALL_PATH_MTU - 40) if: - DF bit in the IP header is set (IPv4) OR if: - Fragmentation Header is present in the IP header (IPv6)
Pass Criteria	3. DUT: Send a SYN,ACK 7. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 9. DUT: Send packet containing data

	10. TESTER: Check data length does not exceed (SMALL_PATH_MTU - 40) if: - DF bit in the IP header is set (IPv4) OR if: - Fragmentation Header is present in the IP header (IPv6)
Test Iterations	
Notes	Derived from RFC 1191 s3 p4 Host specification (MUST)

TCP_HOST_SPEC_02: After receiving Datagram Too Big message

Synopsis	After receiving Datagram Too Big message, host may reduce the size of datagrams it is sending along the path or cease setting DF bit.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to set DF bit 6. TESTER: Cause the application on the DUT-side to send large data 7. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, `DATAGRAM_TOO_BIG` - Next-Hop MTU field set to SMALL_PATH_MTU 9. DUT: Send packet containing data 10. TESTER: Check whether: A) DF bit is not set (IPv4)

	OR Fragmentation Header is not present (IPv6) OR B) length does not exceed (SMALL_PATH_MTU - 40) if: DF bit is set (IPv4) OR: Fragmentation Header is present (IPv6)
Pass Criteria	3. DUT: Send a SYN,ACK 7. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 9. DUT: Send packet containing data 10. TESTER: Check whether: A) DF bit is not set (IPv4) OR Fragmentation Header is not present (IPv6) OR B) length does not exceed (SMALL_PATH_MTU - 40) if: DF bit is set (IPv4) OR: Fragmentation Header is present (IPv6)
Test Iterations	
Notes	Derived from RFC 1191 s3 p4 Host specification (MUST)

TCP_HOST_SPEC_03: Attempt to detect increase - after a Datagram Too Big message

Synopsis	Attempt to detect increase MUST NOT be done less than 5 minutes after a Datagram Too Big message has been received for the given destination
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to send large data 3. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 4. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 5. DUT: Send data packet of size less than (SMALL_PATH_MTU - 40) 6. TESTER: Send ACK for the received data 7. DUT: Continue sending data of size (SMALL_PATH_MTU - 40) 8. TESTER: Send ACK for the received data, and observe for 5 minutes that DUT never sends data more than (SMALL_PATH_MTU - 40) bytes
Pass Criteria	3. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 5. DUT: Send data packet of size less than (SMALL_PATH_MTU - 40) 7. DUT: Continue sending data of size (SMALL_PATH_MTU - 40)
Test Iterations	
Notes	Derived from RFC 1191 s3 p4 Host specification (MUST)

TCP_HOST_SPEC_04: Attempt to detect increase - after a previous, successful attempted increase

Synopsis	Attempt to detect increase MUST NOT be done less than 1 minute after a previous, successful attempted increase.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 8. DUT: Send packet containing data of size (SMALL_PATH_MTU - 40) 9. TESTER: Send ACK for the received data 10. DUT: Continue sending data of size (SMALL_PATH_MTU - 40) 11. TESTER: Continue sending ack for the given data 12. DUT: Send data larger than (SMALL_PATH_MTU - 40) 13. TESTER: Send ACK for the received data 14. DUT: Increase its PMTU estimate and continue sending data of size same as the size of last data packet sent 15. TESTER: Continue sending ACK for received data and check that within one minute, DUT does not send data larger than the size of the last data packet sent
Pass Criteria	3. DUT: Send a SYN,ACK

	6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. DUT: Send packet containing data of size (SMALL_PATH_MTU - 40) 10. DUT: Continue sending data of size (SMALL_PATH_MTU - 40) 12. DUT: Send data larger than (SMALL_PATH_MTU - 40) 14. DUT: Increase its PMTU estimate and continue sending data of size same as the size of last data packet sent
Test Iterations	
Notes	Derived from RFC 1191 s3 p4 Host specification (MUST)

TCP_HOST_SPEC_05: Dealing with Datagram Too Big messages

Synopsis	Hosts MUST be able to deal with Datagram Too Big messages that do not include the next-hop MTU
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to set DF bit 6. TESTER: Cause the application on the DUT-side to send large data 7. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6)

	- type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 9. DUT: Send packet containing data 10. TESTER: Check if data is smaller than the data last received, else check that DF bit is not set (IPv4) OR Fragmentation Header is not present (IPv6)
Pass Criteria	3. DUT: Send a SYN,ACK 7. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 9. DUT: Send packet containing data 10. TESTER: Check if data is smaller than the data last received, else check that DF bit is not set (IPv4) OR Fragmentation Header is not present (IPv6)
Test Iterations	
Notes	Derived from RFC 1191 s3 p4 Host specification (MUST)

TCP_HOST_SPEC_06: Never reduce Path MTU below 68 octets

Synopsis	Host MUST never reduce its estimate of the Path MTU below 68 octets.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to set DF bit 6. TESTER: Cause the application on the DUT-side to send large data 7. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to below 68 OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to below 68 9. DUT: Send packet containing data 10. TESTER: Check data length is not smaller than (68 - 40)
Pass Criteria	3. DUT: Send a SYN,ACK 7. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 9. DUT: Send packet containing data 10. TESTER: Check data length is not smaller than (68 - 40)
Test Iterations	
Notes	Derived from RFC 1191 s3 p5 Host specification (MUST)

TCP_HOST_SPEC_07: No increase of estimate of the Path MTU

Synopsis	Host MUST not increase its estimate of the Path MTU in response to the contents of a Datagram Too Big message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data with DF bit set 7. DUT: Send data with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to a value larger than TESTER MSS OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, `DATAGRAM_TOO_BIG` - Next-Hop MTU field set to a value larger than TESTER MSS 9. DUT: Send packet containing data 10. TESTER: Check data length does not exceed the size of data received before this packet.
Pass Criteria	3. DUT: Send a SYN,ACK 6. DUT: Send data with DF bit set 7. DUT: Send data with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 9. DUT: Send packet containing data 10. TESTER: Check data length does not exceed the size of data received

	before this packet.
Test Iterations	
Notes	Derived from RFC 1191 s3 p5 Host specification (MUST)

TCP_HOST_SPEC_08: Datagrams larger than 40 octets plus MSS

Synopsis	Host must not send datagrams larger than 40 octets plus the Maximum Segment Size (MSS) sent by its peer.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the application on the DUT-side to issue an active OPEN call 2. DUT: Send a SYN 3. TESTER: Send a SYN,ACK with MSS smaller than DUT MSS 4. DUT: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Check data length is not larger than TESTER MSS
Pass Criteria	2. DUT: Send a SYN 4. DUT: Send an ACK 6. DUT: Send data with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Check data length is not larger than TESTER MSS
Test Iterations	
Notes	Derived from RFC 1191 s3.1 p5 TCP MSS Option (MUST)

TCP_HOST_SPEC_09: Host receiving a Datagram Too Big message from an unmodified router - PMTU

Synopsis	Host receiving a Datagram Too Big message from an unmodified router assumes that the PMTU is minimum of its currently-assumed PMTU and 576
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to 0 OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to 0 8. DUT: Send packet containing data 9. TESTER: Check data length is less than or equal to (576-40)
Pass Criteria	3. DUT: Send a SYN,ACK 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. DUT: Send packet containing data 9. TESTER: Check data length is less than or equal to (576-40)
Test Iterations	

Notes	Derived from RFC 1191 s5 p7 Host processing of old-style messages (MUST)

TCP_HOST_SPEC_10: Host receiving a Datagram Too Big message from an unmodified router - stop DF bit

Synopsis	Host receiving a Datagram Too Big message from an unmodified router stops setting the DF bit in datagrams sent on that path.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to 0 OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to 0 8. DUT: Send packet containing data 9. TESTER: Check that DF bit is not set (IPv4) OR that Fragmentation Header is not present (IPv6)
Pass Criteria	3. DUT: Send a SYN,ACK 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. DUT: Send packet containing data 9. TESTER: Check that DF bit is not set (IPv4) OR that Fragmentation Header is not present (IPv6)

Test Iterations	
Notes	Derived from RFC 1191 s5 p7 Host processing of old-style messages (MAY)

4.9.6.23 TCP Layer Actions

TCP_LAYER_ACTIONS_01: Increase in the PMTU estimate

Synopsis	An upper layer MUST not retransmit datagrams in response to an increase in the PMTU estimate
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU8. DUT: Send packet containing data of size (SMALL_PATH_MTU - 40) 9. TESTER: Send ACK for the received data 10. DUT: Continue sending data of size (SMALL_PATH_MTU - 40) 11. TESTER: Continue sending ACK for the given data 12. DUT: Send data larger than (SMALL_PATH_MTU - 40) 13. TESTER: Send ACK for the received data 14. DUT: Increase its PMTU estimate and continue sending data of size same as the size of last data packet sent 15. TESTER: Check the sequence number of last two packet received are not same
Pass Criteria	3. DUT: Send a SYN,ACK 6. DUT: Send data of size DUT MSS with:

	- DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. DUT: Send packet containing data of size (SMALL_PATH_MTU - 40) 10. DUT: Continue sending data of size (SMALL_PATH_MTU - 40) 12. DUT: Send data larger than (SMALL_PATH_MTU - 40) 14. DUT: Increase its PMTU estimate and continue sending data of size same as the size of last data packet sent 15. TESTER: Check the sequence number of last two packet received are not same
Test Iterations	
Notes	Derived from RFC 1191 s6.3 p12 Purging stale PMTU information (MUST)

TCP_LAYER_ACTIONS_02: Datagram Too Big message - treat it as dropped segment

Synopsis	On receiving Datagram Too Big message treat it as dropped segment, and wait till retransmission timer expires and retransmit segment
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to ESTABLISHED state 2. TESTER: Cause the application on the DUT-side to send large data 3. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 4. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 5. DUT: Wait till retransmission timer expires and retransmit packet containing data 6. TESTER: Check the sequence number of last two received packet are same
Pass Criteria	3. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 5. DUT: Wait till retransmission timer expires and retransmit packet containing data 6. TESTER: Check the sequence number of last two received packet are same
Test Iterations	

Notes	Derived from RFC 1191 s6.4 p13 TCP layer actions (MUST)

TCP_LAYER_ACTIONS_03: Datagram size used in the retransmission

Synopsis	The Datagram size used in the retransmission should, of course, be no larger than the new PMTU.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 8. DUT: Retransmit packet containing data 9. TESTER: Check retransmitted data length does not exceed (SMALL_PATH_MTU - 40) OR - DF bit is unset (IPv4) OR without: - Fragmentation Header present (IPv6)
Pass Criteria	3. DUT: Send a SYN,ACK 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. DUT: Retransmit packet containing data 9. TESTER: Check retransmitted data length does not exceed

	(SMALL_PATH_MTU - 40) OR - DF bit is unset (IPv4) OR without: - Fragmentation Header present (IPv6)
Test Iterations	
Notes	Derived from RFC 1191 s6.4 p13 TCP layer actions (MUST)

TCP_LAYER_ACTIONS_04: Retransmission caused by a Datagram Too Big message - slow-start mechanism

Synopsis	Retransmission caused by a Datagram Too Big message triggers the slow-start mechanism(i.e., only one segment should be retransmitted)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with SMALL_MAX_SEG_SIZE and LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send large data 4. DUT: Send <cwnd-ini> segment and wait for its ACK 5. TESTER: Send ACK for the received segment 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for its ACKs 7. TESTER: Send an ICMP packet with code field set to DATAGRAM_TOO_BIG and Next-Hop MTU field set to SMALL_PATH_MTU. 8. DUT: Retransmit packet containing data and wait for its ACK 9. TESTER: Check whether DUT send only one packet
Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> and ssthresh to 65535 bytes 4. DUT: Send <cwnd-ini> segment and wait for its ACK 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for its ACKs 8. DUT: Retransmit packet containing data and wait for its ACK 9. TESTER: Check whether DUT send only one packet

Test Iterations	
Notes	Derived from RFC 1191 s6.4 p14 TCP layer actions (MUST)

TCP_LAYER_ACTIONS_05: Retransmission caused by a Datagram Too Big message - no change of congestion window

Synopsis	Retransmission caused by a Datagram Too Big message should not change the congestion window. It should, however, trigger the slow-start mechanism
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with SMALL_MAX_SEG_SIZE and LARGE_WINDOW_SIZE 2. DUT: Set cwnd to <cwnd-ini> segments and ssthresh to 65535 bytes 3. TESTER: Cause the DUT-side application to send large data 4. DUT: Send <cwnd-ini> segments and wait for their ACK 5. TESTER: Send ACK for the received segments 6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACKs 7. TESTER: Send ACKs for all the received segments 8. TESTER: Repeat the last two steps till DUT sets cwnd to 4 segments 9. DUT: Set cwnd to 4 segments. Send 4 segments 10. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 11. DUT: Retransmit data segment and trigger slow-start mechanism 12. TESTER: Send ACKs for all the received segments 13. TESTER: Verify that DUT follows Path MTU Discovery, by not changing the congestion window during slow-start
Pass Criteria	2. DUT: Set cwnd to <cwnd-ini> segments and ssthresh to 65535 bytes 4. DUT: Send <cwnd-ini> segments and wait for their ACK

	6. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACKs 9. DUT: Set cwnd to 4 segments. Send 4 segments 11. DUT: Retransmit data segment and trigger slow-start mechanism 13. TESTER: Verify that DUT follows Path MTU Discovery, by not changing the congestion window during slow-start
Test Iterations	
Notes	Derived from RFC 1191 s6.4 p14 TCP layer actions (SHOULD)

TCP_LAYER_ACTIONS_06: Maximum window size

Synopsis	Maximum window size should be set to greatest multiple of segment size (PMTU - 40) that is less than or equal to sender's buffer space size.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Set TCP buffer size of the DUT to 4K 2. TESTER: Take DUT to ESTABLISHED state with SMALL_MAX_SEG_SIZE and LARGE_WINDOW_SIZE 3. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 4. TESTER: Retrieve DUT buffer size and make the DUT-side application send large data 5. DUT: Send <cwnd-ini> segment and wait for its ACK 6. TESTER: Send ACK for the received segment 7. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACKs 8. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 9. DUT: Send packet containing data and wait for its ACK 10. TESTER: Retrieve window size that should be greatest multiple of (PMTU - 40) and less than or equal to DUT buffer size.
Pass Criteria	3. DUT: Set cwnd to <cwnd-ini> segment and ssthresh to 65535 bytes 5. DUT: Send <cwnd-ini> segment and wait for its ACK 7. DUT: Set cwnd to 2*<cwnd-ini>, send 2*<cwnd-ini> segments and wait for their ACKs 9. DUT: Send packet containing data and wait for its ACK 10. TESTER: Retrieve window size that should be greatest multiple of (PMTU - 40) and less than or equal to DUT buffer size.
Test Iterations	
Notes	Derived from RFC 1191 s6.4 p14 TCP layer actions (SHOULD)

TCP_LAYER_ACTIONS_07: PMTU Discovery

Synopsis	PMTU Discovery does not affect the value sent in the TCP MSS option
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause DUT to move on to LISTEN state 2. TESTER: Send a SYN 3. DUT: Send a SYN,ACK 4. TESTER: Send an ACK 5. TESTER: Cause the application on the DUT-side to send large data 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 7. TESTER: Send an ICMP packet with (IPv4) - type set to 3, Destination Unreachable message - code field set to 4, Fragmentation needed and DF bit set - Next-Hop MTU field set to SMALL_PATH_MTU OR TESTER: Send an ICMPv6 packet with (IPv6) - type set to 2, Packet too big message - code field set to 0, DATAGRAM_TOO_BIG - Next-Hop MTU field set to SMALL_PATH_MTU 8. DUT: Send packet containing data 9. TESTER: Check Options field of TCP header, if it contains MSS, then value should be equal to MSS value sent with the initial SYN.
Pass Criteria	3. DUT: Send a SYN,ACK 6. DUT: Send data of size DUT MSS with: - DF bit set (IPv4) OR with: - Fragmentation Header present (IPv6) 8. DUT: Send packet containing data 9. TESTER: Check Options field of TCP header, if it contains MSS, then value should be equal to MSS value sent with the initial SYN.
Test Iterations	

Notes	Derived from RFC 1191 s6.4 p14 TCP layer actions (MUST)

4.9.6.24 Improved Window Algorithms – RFC 813 & CONGESTION CONTROL IN IP/TCP INTERNETWORKS- RFC 896

TCP_IMPROVED_WINDOWING_01: Postponing ack

Synopsis	So, if a segment arrives, postpone sending an acknowledgement if both of the following conditions hold. First, the push bit is not set in the segment, since it is a reasonable assumption that there is more data coming in a subsequent segment. Second, there is no revised window information to be sent back. The text suggested that when setting a timer to postpone the sending of an acknowledgement, a fixed interval of 200 to 300 milliseconds would work properly in practice. Note: DUT is fine if it changes its window size evenif ack does not come within the timing range
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Cause the DUT to move on to ESTABLISHED state at a <wnp> 2. TESTER: Send a data segment with PUSH bit unset 3. DUT: Send ACK 4. TESTER: Verify that the Ack has been received within 200 to 300 milliseconds
Pass Criteria	3. DUT: Send ACK 4. TESTER: Verify that the Ack has been received within 200 to 300 milliseconds
Test Iterations	
Notes	Derived from RFC 813 s5 p11 Improved Acknowledgement Algorithms (MUST)

TCP_IMPROVED_WINDOWING_02: Avoiding SWS – less 25%

Synopsis	Simple experiments suggest that the exact value of the ratio is not very important, but that a value of about 25 percent is sufficient to avoid SWS and achieve reasonable throughput, even for machines with a small offered window..
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS <PARAM-MSS> and window 2 * <PARAM-MSS> 2. TESTER: Cause the DUT-side application to send 4 * <PARAM-MSS> 3. DUT: Send data segments 4. TESTER: Send ACK for the 1st received segment with window size <PARAM-MSS> + <PARAM-MSS> / 3 5. DUT: Usable window is smaller than 25%, DUT does not send a new data segment
Pass Criteria	3. DUT: Send data segments 5. DUT: Usable window is smaller than 25%, DUT does not send a new data segment
Test Iterations	
Notes	Derived from RFC 813 s 4 p 8 Improved Window Algorithms (SHOULD)

TCP_IMPROVED_WINDOWING_03: Avoiding SWS – more 25%

Synopsis	Simple experiments suggest that the exact value of the ratio is not very important, but that a value of about 25 percent is sufficient to avoid SWS and achieve reasonable throughput, even for machines with a small offered window.. Note: Value more than 25 percent will release new data from dut
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS <PARAM-MSS> and window 2 * <PARAM-MSS> 2. TESTER: Cause the DUT-side application to send 4 * <PARAM-MSS> 3. DUT: Send data segments 4. TESTER: Send ACK for the 1st received segment with window size 1.5 * <PARAM-MSS> 5. DUT: Calculate usable window and it'll be more than 25% of offered window It'll send 1 new data segment
Pass Criteria	3. DUT: Send data segments 5. DUT: Calculate usable window and it'll be more than 25% of offered window It'll send 1 new data segment
Test Iterations	
Notes	Derived from RFC 813 s 4 p 8 Improved Window Algorithms (SHOULD)

TCP_IMPROVED_WINDOWING_04: Attempt to hold off sending – equal to MSS

Synopsis	An additional enhancement which might help throughput would be to attempt to hold off sending until one can send a maximum size segment.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS <PARAM-MSS> and window 2 * <PARAM-MSS> 2. TESTER: Cause the DUT-side application to send 3 * <PARAM-MSS> 3. DUT: Send data segments 4. TESTER: Send ACK for the 1st received segment with window size of 2 * <PARAM-MSS> 5. DUT: Send a new data segment as the usable window is bigger than the Maximum Segment Size
Pass Criteria	3. DUT: Send data segments 5. DUT: Send a new data segment as the usable window is bigger than the Maximum Segment Size
Test Iterations	
Notes	Derived from RFC 813 s 4 p 8 Improved Window Algorithms (MAY)

TCP_IMPROVED_WINDOWING_05: Attempt to hold off sending – less than MSS

Synopsis	An additional enhancement which might help throughput would be to attempt to hold off sending until one can send a maximum size segment.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS <PARAM-MSS> and window 2 * <PARAM-MSS> 2. TESTER: Cause the DUT-side application to send 3 * <PARAM-MSS> bytes of data 3. DUT: Send data segments 4. TESTER: Send ACK for the 1st received segment with window size 2 * <PARAM-MSS> - 2 5. DUT: Does not send a new data segment as the usable window is smaller than the Maximum Segment Size It'll not send a new data segment
Pass Criteria	3. DUT: Send data segments 5. DUT: Does not send a new data segment as the usable window is smaller than the Maximum Segment Size It'll not send a new data segment
Test Iterations	
Notes	Derived from RFC 813 s 4 p 8 Improved Window Algorithms (MAY)

TCP_IMPROVED_WINDOWING_08: Refraining from sending – more than MSS

Synopsis	If it refrains from sending now, it will typically send the data later when an incoming packet arrives and changes the state of the system. The state changes in one of two ways; the incoming packet acknowledges old data the distant host has received, or announces the availability of buffer space in the distant host for new data. Note: This case is for announcement of availability of buffer space
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS <PARAM-MSS> and window 2 * <PARAM-MSS> 2. TESTER: Cause the DUT-side application to send 2 * <PARAM-MSS> + 15 3. DUT: Send data segments 4. TESTER: Send ACK for the all received segment with window size 0 5. DUT: It'll not send any new data segments
Pass Criteria	3. DUT: Send data segments 5. DUT: It'll not send any new data segments
Test Iterations	
Notes	Derived from RFC 896 Section "The solution to the small-packet problem" (MUST)

TCP_IMPROVED_WINDOWING_09: Refraining from sending – less than MSS

Synopsis	If it refrains from sending now, it will typically send the data later when an incoming packet arrives and changes the state of the system. The state changes in one of two ways; the incoming packet acknowledges old data the distant host has received, or announces the availability of buffer space in the distant host for new data.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Take DUT to ESTABLISHED state with MSS <PARAM-MSS> and window <PARAM-MSS> 2. TESTER: Cause the DUT-side application to send <PARAM-MSS> - 10 3. DUT: Send 1 data segments with push 4. TESTER: Cause the DUT-side application to send <PARAM-MSS> - 10 5. DUT: It'll not send any new data segment 6. TESTER: Send ACK for the received segment 7. DUT: It'll send a new data segment
Pass Criteria	3. DUT: Send 1 data segments with push 5. DUT: It'll not send any new data segment 7. DUT: It'll send a new data segment
Test Iterations	
Notes	Derived from RFC 896 Section "The solution to the small-packet problem" (SHOULD)

4.9.6.25 Connection Establishment

TCP_CONNECTION_ESTAB_01: To verify that the DUT accepts connections from several remote sockets with one passive socket

Synopsis	Ensure that when DUIT is requested to open a TCP passive socket and receive multiple Tcp Segments from 3 different ports containing SYN flag indicating a value of 1 then DUT opens a TCP passive socket and sends a TCP Segment to each sending port containing SYN and ACK flags indicating a value of 1.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an UpperTester <OpenTCPSocket (passive)> on a <PORT1> 2. TESTER: Send 3 TCP Segments with SYN set to 1 from 3 different ports 3. DUT: Opens a TCP passive socket and sends 3 TCP segments on each sending port with SYN and ACK set to 1
Pass Criteria	DUT opens a TCP passive socket and sends a TCP Segment to each sending port containing SYN and ACK flags indicating a value of 1.
Test Iterations	
Notes	Derived from RFC 793, chapter 1.5., page 5.

TCP_CONNECTION_ESTAB_02: To verify that the DUT opens multiple Passive sockets and connects them to remote socket

Synopsis	Ensure that when DUT is requested to open multiple TCP passive sockets and DUT receives multiple Tcp Segments through these sockets containing SYN Flag indicating a value of 1 Then DUT opens a TCP passive sockets and DUT sends multiple TCP segments to the sending port containing SYN and ACK Flags indicating a value of 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an UpperTester <OpenMultipleTCPSocket (passive,3)> request. 2. TESTER: Send 3 TCP segments with SYN set to 1 to the 3 opened sockets. 3. DUT: Opens 3 TCP passive sockets and send 3 TCP segements with SYN and ACK set to 1 corresponding to the received SYNs.
Pass Criteria	DUT opens a TCP passive sockets and DUT sends multiple TCP segments to the sending port containing SYN and ACK Flags indicating a value of 1
Test Iterations	
Notes	Derived from RFC 793, chapter 1.5., page 5.

TCP_CONNECTION_ESTAB_03: To verify that the DUT opens multiple Active sockets and connects them to remote socket

Synopsis	Ensure that when DUT is requested to open multiple TCP active sockets and DUT receives Tcp Segments with SYN,ACK Flags set to 1 (After that DUT sends SYN) Then DUT opens multiple TCP active sockets' and DUT sends multiple Tcp Segments from the open active sockets containing SYN flag indicating a value of 1 and DUT receives SYN ACK as a resonpse to its SYN and DUT sends multiple TCP segments from the open active sockets containing ACK flag indicating a value of 1.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send an UpperTester <OpenMultipleTCPSocket (active,3)> request 2. DUT: Opens 3 TCP active sockets and send 3 SYNs from each socket. 3. TESTER: Send 3 TCP Segments with SYN, ACK set to 1 for each socket. 4. DUT: Send 3 TCP Segments with ACK set to 1 from each socket.
Pass Criteria	DUT opens multiple TCP active sockets' and DUT sends multiple Tcp Segments from the open active sockets containing SYN flag indicating a value of 1 and DUT sends multiple TCP segments from the open active sockets containing ACK flag indicating a value of 1.
Test Iterations	
Notes	Derived from RFC 793, chapter 1.5., page 5.

TCP_CONNECTION_ESTAB_07: To verify that DUT accepts remote closing of a connection.

Synopsis	Ensure that when DUT receives a FIN packet then DUT sends an ACK packet and DUT sends a 'FIN s in response to user's 'Close' request and DUT moves to 'CLOSED' state.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to state ‘ESTABLISHED’. 2. TESTER: Send a FIN packet 3. DUT: Sends an ACK packet. 4. DUT: Sends a FIN packet and moves to ‘CLOSED’ state. 5. TESTER: externally check the state of the DUT.
Pass Criteria	DUT sends an ACK packet and DUT sends a 'FIN s in response to user's 'Close' request and DUT moves to 'CLOSED' state.
Test Iterations	
Notes	Derived from RFC 793, chapter 3.5. Closing a Connection.

TCP_CONNECTION_ESTAB_12: To verify that DUT responds to TCP keep-alive segments

Synopsis	Ensure that when DUT receives an ACK of X with no payload then DUT sends an ACK packet indicating DUT 'still-alive'
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send a Keep-Alive ACK packet with SEG.SEQ = SND.NXT - 1 (Note: on a quiet connection SND.NXT = RCV.NXT) so that the Sequence-Number is out of the receive window 3. DUT: Sends an ACK packet.
Pass Criteria	DUT sends an ACK packet indicating DUT 'still-alive'
Test Iterations	
Notes	Derived from RFC 1122 chapter 4.2.3.6 TCP Keep-Alives

4.9.6.26 Header

TCP_HEADER_01: To verify that DUT generates a TCP packet containing valid header field values

Synopsis	Ensure that when DUT generates a 'TCP packet' then DUT sends a 'TCP packet' containing a well-formed 'TCP header'
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state. 2. TESTER: Send <generateTCPSegment> request. 3. DUT: Sends a data segment containing valid header values.
Pass Criteria	DUT sends a 'TCP packet' containing a well-formed 'TCP header'
Test Iterations	
Notes	Derived from RFC 793.

TCP_HEADER_02: To verify that DUT accepts the TCP packet containing valid header field values

Synopsis	Ensure that when DUT receives a valid 'TCP packet' then DUT sends and ACK
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state. 2. TESTER: Send a data packet 3. DUT: Sends an ACK packet with the expected Ack Number.
Pass Criteria	DUT sends and ACK
Test Iterations	
Notes	RFC 793.

TCP_HEADER_04: To verify that a DUT discards the packet in case TCP header contains invalid source port

Synopsis	Ensure that when DUT receives a TCP packet containing a Source Port indicating a value different from PORT1 containing a sequence number indicating SEQ1 then DUT discards the TCP packet and optionally DUT sends a RST packet
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state on <PORT1>. 2. TESTER: Send a TCP packet on a <PORT2> with Sequence Number set to <SEQ1> 3. DUT: DUT discards the TCP packet and optionally DUT sends a RST packet
Pass Criteria	DUT discards the 'TCP packet' and optionally DUT sends a 'RST packet'
Test Iterations	
Notes	RFC 793.

TCP_HEADER_05: To verify that a DUT accepts the packet in case TCP header Reserved field having a zero value

Synopsis	Ensure that when DUT receives a TCP packet containing a Reserved field indicating a zero value then DUT accepts the TCP packet
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send a TCP packet with Reserved field set to zero 3. DUT: Send an ACK with the expected Ack Number.
Pass Criteria	DUT accepts the 'TCP packet'
Test Iterations	
Notes	RFC 793, chapter 3.1, RFC 4413, chapter 4.2.3., Reserved

TCP_HEADER_06: To verify that a DUT accepts the packet in case TCP header Reserved field having non-zero value.

Synopsis	Ensure that when DUT receives a 'TCP packet ' containing 'Reserved field ' indicating a non-zero value then DUT ignores the 'Reserved' field and DUT accepts the 'TCP packet'
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send a TCP packet with Reserved field different from zero 3. DUT: Send an ACK with the expected Ack Number.
Pass Criteria	DUT ignores the 'Reserved' field and DUT accepts the 'TCP packet'
Test Iterations	
Notes	RFC 4413, chapter 4.2.3, Reserved.

TCP_HEADER_07: To verify that DUT discards TCP packets in case TCP header data offset field having an invalid non zero value

Synopsis	Ensure that when DUT receives a TCP packet containing Data Offset indicating a value less than 5 and non-zero then DUT discards TCP packet
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send a TCP packet with Data Offset value less than 5 3. DUT: Discards the TCP packet.
Pass Criteria	DUT discards the TCP packet
Test Iterations	
Notes	RFC 793 chapter 3.1,

TCP_HEADER_08: To verify that DUT discards TCP packets in case TCP header data offset field having value greater than the actual value

Synopsis	Ensure that when DUT receives a TCP packet containing Data Offset indicating a value greater than the actual value then DUT discards TCP packet
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send a TCP packet with Data Offset value greater than 5 3. DUT: Discards the TCP packet.
Pass Criteria	DUT discards 'TCP packet'
Test Iterations	
Notes	RFC 793 chapter 3.1,

TCP_HEADER_09: To verify that DUT discards the TCP packet in case TCP header checksum value is zero

Synopsis	Ensure that when DUT receives a TCP packet containing a Checksum indicating a value of 0 then DUT discards the TCP packet and sends no ACK back
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state. 2. TESTER: Send a TCP packet with a Checksum = 0 and Sequence Number = <SEQ> 3. DUT: Discards the TCP packet and sends no ACK back.
Pass Criteria	DUT discards the TCP packet and does not send an ACK back
Test Iterations	
Notes	RFC 793.

TCP_HEADER_10: To verify that DUT discards TCP packets in case header zero option length

Synopsis	Ensure that when DUT receives a TCP packet containing a Header Option indicating a value of 'NOP' and containing 'EOP' indicating length := 0 then DUT discards the TCP packet and sends no ACK back
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state. 2. TESTER: Send a TCP packet with a Header Option = ‘NOP’ and ‘EOP’ Lentgh = 0 3. DUT: Discards the TCP packet and sends no ACK back.
Pass Criteria	DUT discards the TCP packet and sends no ACK back
Test Iterations	
Notes	RFC 793.

TCP_HEADER_11: To verify that DUT discards TCP packets with SYN flag set and a Multicast IP Destination Address

Synopsis	Ensure that when DUT receives a TCP packet' containing SYN = 1 and containing IP Destination Address indicating a value of <multicastAddress> then Discards the TCP packet and sends no ACK back.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state. 2. TESTER: Send a SYN with IP Destination Address = <multicastAddress> 3. DUT: Discards the TCP packet and sends no ACK back.
Pass Criteria	Discards the TCP packet and sends no ACK back.
Test Iterations	
Notes	RFC 793

4.9.6.27 Sequence Number

TCP_SEQUENCE_01: To verify that DUT synchronizes on initial sequence number in state 'LISTEN'

Synopsis	Ensure that when DUT receives a SYN packet containing 'Sequence Number' indicating <ISN> then DUT sends an ACK containing an Acknowledgment Number indicating <ISN>+1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to state ‘LISTEN’. 2. TESTER: Send a SYN with the Sequence Number equal to <ISN> 3. DUT: Sends an SYN,ACK with an ACK Number equal to <ISN>+1
Pass Criteria	DUT sends an SYN,ACK containing an Acknowledgment Number indicating <ISN>+1'
Test Iterations	
Notes	RFC 793 chapter 3.1.

TCP_SEQUENCE_02: To verify that DUT synchronizes on initial sequence number in state 'SYN SENT'

Synopsis	Ensure that when DUT receives a SYN,ACK packet containing Sequence Number indicating <ISN> then DUT sends an ACK packet containing Acknowledgment Number indicating <ISN>+1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to state ‘SYN-SENT’. 2. TESTER: Send a SYN,ACK with the Sequence Number equal to <ISN> 3. DUT: Sends an ACK with an ACK Number equal to <ISN>+1
Pass Criteria	DUT sends an ACK packet containing Acknowledgment Number indicating <ISN>+1'
Test Iterations	
Notes	RFC 793.

TCP_SEQUENCE_03: To verify that DUT accepts the TCP packet in case initial sequence number has zero value

Synopsis	Ensure that when DUT receives a SYN packet containing Sequence Number indicating a zero value then DUT accepts the TCP packet
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to state 'LISTEN'. 2. TESTER: Send a SYN with the Sequence Number equal to 0 3. DUT: Sends an SYN,ACK with an ACK Number equal to 1
Pass Criteria	DUT accepts the TCP packet
Test Iterations	
Notes	RFC 793.

TCP_SEQUENCE_04: To verify that DUT accepts the TCP packet in case initial sequence number has maximum value

Synopsis	Ensure that when DUT receives a TCP packet containing Sequence Number indicating <SeqMaxVal> value then DUT accepts the TCP packet and DUT sends an SYN,ACK packet containing Acknowledgment Number indicating 0
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Send a SYN to DUT at <wnp> with Initial Sequence Number equal to <SeqMaxVal>. 2. DUT: Send SYN,ACK with Ack Number equal to 0.
Pass Criteria	DUT accepts the TCP packet and DUT sends an SYN,ACK packet containing Acknowledgment Number indicating 0
Test Iterations	
Notes	RFC 793.

TCP_SEQUENCE_05: To verify that DUT accepts the TCP packets in case sequence numbers received are in the right order

Synopsis	Ensure that when DUT receives a severalTCP packets containing SEQ indicating values incremented correctly. then DUT sends an ACK for every received TCP Segments
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send several TCP packets with Sequence Number correcitly incremented 3. DUT: Sends an ACK for every received packet with the expected Ack Numbers.
Pass Criteria	DUT sends an ACK for every received TCP Segments indicating TCP Segments
Test Iterations	
Notes	RFC 793.

4.9.6.28 Acknowledgment

TCP_ACKNOWLEDGEMENT_02: To verify that DUT accpets the ACK piggybacked with next transmit packet

Synopsis	Ensure that when DUT receives a TCP packet containing ACK flag indicating a value of 1 and a payload then Sends an ACK containing Ack Number indicating the expected value
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state. 2. TESTER: Send <sendTCPPacket> on the <udpPort> to request a TCP packet from the DUT. 3. DUT: Sends a TCP packet with the expected Sequence Number 4. TESTER: Send an ACK with a <payload>. 5. DUT: Accepts the TCP packet and sends an ACK with the expected Akc Number.
Pass Criteria	DUT accepts the TCP packet and sends an ACK with the expected Ack Number
Test Iterations	
Notes	RFC 793.

TCP_ACKNOWLEDGEMENT_03: To verify that DUT sends only ACK in case no packet left to send

Synopsis	Ensure that when DUT has no data left to send and DUT receives a TCP packet then DUT sends an ACK containing an Ack Number indicating the expected value.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state using a passive socket call. 2. TESTER: Send TCP packet 3. DUT: Sends an ACK with the expected Ack Number.
Pass Criteria	DUT sends an ACK with the expected Ack Number.
Test Iterations	
Notes	RFC 793.page 74

TCP_ACKNOWLEDGEMENT_04: To verify that DUT receives ACKs alone (no piggybagging)

Synopsis	Ensure that when DUT receives an ACK containing Length indicating a value of 0 (an empty ACK, no piggybagging) then DUT does not send an RST and DUT ends connection with the expected ACK number
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send <sendTCPPacket> to request a TCP packet from the DUT. 3. DUT: Sends a TCP packet. 4. TESTER: Send an ACK with the expected Ack Number, with Length equal to 0. 5. DUT: Sends no RST and ends the connection correctly when requested to.
Pass Criteria	DUT does not send an RST and DUT ends connection with the expected ACK number
Test Iterations	
Notes	RFC 793.

4.9.6.29 Control Flags

TCP_CONTROL_FLAGS_01: To verify that DUT generates TCP data packet with PSH flag not set

Synopsis	Ensure that when DUT is requested to generate a TCP packet then DUT generates a TCP packet containing a PSH flag indicating a value of 0
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to ‘ESTABLISHED’ state. 2. TESTER: Send <sendTCPPacket> to request a TCP packet from the DUT. 3. DUT: Sends a TCP packet with PSH flag not set.
Pass Criteria	DUT generates a TCP packet containing a PSH flag indicating a value of 0
Test Iterations	
Notes	RFC 4413 chapter 4.2.4.

TCP_CONTROL_FLAGS_05: To verify that DUT receives TCP data packet in case URG flag is set

Synopsis	Ensure that when DUT receives a TCP packet containing URG flag indicating a value of 1 then DUT sends an ACK containing an Ack Number indicating the expected value.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send a TCP packet with URG being set to DUT. 3. DUT: Sends an ACK with expected Ack Number.
Pass Criteria	DUT sends an ACK containing an Ack Number indicating the expected value.
Test Iterations	
Notes	RFC 4413 chapter 4.2.4.

TCP_CONTROL_FLAGS_07: To verify that DUT accepts TCP data packet with both PSH and URG Flags set to 1

Synopsis	Ensure that when DUT receives a 'TCP packet' containing 'URG flag' and 'PSH flag' indicating a value of 1 buffer then DUT sends an ACK containing an Ack Number indicating the expected value.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'ESTABLISHED' state. 2. TESTER: Send a TCP packet with URG and PSH flags being set, to DUT. 3. DUT: Sends an ACK with expected Ack Number.
Pass Criteria	DUT sends an ACK containing an Ack Number indicating the expected value.
Test Iterations	
Notes	RFC 4413 chapter 4.2.4.

TCP_CONTROL_FLAGS_08: To verify that DUT recovers the duplicated SYN by sending RST

Synopsis	Ensure that when DUT receives a SYN packet containing RST flag indicating a value of 1 and containg a Sequence Number indicating a value of a late duplicated SYN and DUT receives a SYN packet containing a Sequence Number indicating <SEQ> then DUT sends a SYN,ACK containing an Ack Number indicating a value of <SEQ> + 1 and DUT moves from state 'LISTEN' to 'SYN-RECEIVED'
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: Bring DUT to 'LISTEN' state. 2. TESTER: Send SYN with Sequence Number equal to <SEQ1> 3. DUT: Sends a SYN,ACK with Ack Number equal to <SEQ1>+1 and moves to 'SYN-RECEIVED' state. 4. TESTER: Send SYN with RST flag set with Sequence Number equal to <SEQ1> 5. TESTER: Send SYN with Sequence Number equal to <SEQ2> 6 DUT: Sends a SYN,ACK with Ack Number equal to <SEQ2>+1
Pass Criteria	DUT sends a SYN,ACK containing an Ack Number indicating a value of <SEQ> + 1 and DUT moves from state 'LISTEN' to 'SYN-RECEIVED'
Notes	RFC 793.

5 Test Scope Automotive Protocols

5.1 Scalable service-Oriented MiddlewarE over IP Protocol (SOME/IP)

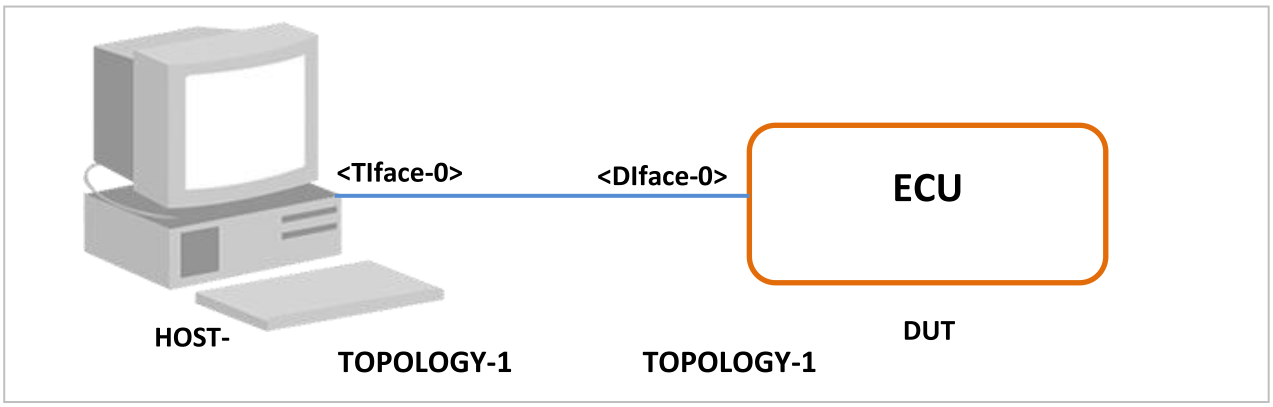
5.1.1 General

5.1.1.1 Referenced specification

The scope of this chapter is to specify test cases for Scalable service-Oriented MiddlewarE over IP Protocol from the following standards:

- Specification of Service Discovery V1.2.0 R4.1 Rev 3
- Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3

5.1.1.2 Simulated topologies



5.1.1.3 Required topology related configuration

Tester configuration required for the tests in the following sections pertaining to SOME/IP tests:

- Correct DUT IP Address for DUT interface connected to TESTER interface
- All the test cases in this suite require DUT to be configured with only one IP interface

5.1.1.4 Coverage

Specification Document	Section Number	Test Category	Test Number(s)
Specification of Service Discovery V1.2.0 R4.1 Rev 3	7.3	Message format	SOMEIPSRV_FORMAT_01 to SOMEIPSRV_FORMAT_27
Specification of Service Discovery V1.2.0 R4.1 Rev 3	7.3.9	Options Array	SOMEIPSRV_OPTIONS_01 to SOMEIPSRV_OPTIONS_15
Specification of Service Discovery V1.2.0 R4.1 Rev	7.4.5	StopOfferService entry	SOMEIPSRV_SD_BEHAVIOR_05

3			
Specification of Service Discovery V1.2.0 R4.1 Rev 3	7.4.6.2	StopSubscribeEventgroup entry	SOMEIPSRV_SD_BEHAVIOR_06
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.4.1.1	Find Service Entry	SOMEIPSRV_SD_MESSAGE_01 to SOMEIPSRV_SD_MESSAGE_06
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.4.1.2	Offer Service Entry	SOMEIPSRV_SD_MESSAGE_07 to SOMEIPSRV_SD_MESSAGE_09
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.4.1.3	Stop Offer Service Entry	SOMEIPSRV_SD_MESSAGE_10
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.4.2.1	Subscribe Eventgroup Entry	SOMEIPSRV_SD_MESSAGE_11
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.4.2.2	Stop Subscribe Eventgroup Entry	SOMEIPSRV_SD_MESSAGE_12
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.4.2.3	Subscribe Eventgroup Acknowledgement Entry	SOMEIPSRV_SD_MESSAGE_13
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.4.2.4	Subscribe Eventgroup Negative Acknowledgement Entry	SOMEIPSRV_SD_MESSAGE_14 to SOMEIPSRV_SD_MESSAGE_19
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.5.1	Startup Behavior	SOMEIPSRV_SD_BEHAVIOR_01 to SOMEIPSRV_SD_BEHAVIOR_02
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.7.5.2	Server Answer Behavior	SOMEIPSRV_SD_BEHAVIOR_03 to SOMEIPSRV_SD_BEHAVIOR_04
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.1	Definition of Identifiers	SOMEIPSRV_BASIC_01 to SOMEIPSRV_BASIC_03
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.2.3.1.1	Mapping of IP Addresses and Ports Response and Error Messages	SOMEIPSRV_ONWIRE_01 to SOMEIPSRV_ONWIRE_12

Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.2.3.2.1	Structure of the Message ID	SOMEIPSRV_ONWIRE_02
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.2.3.4	Request ID	SOMEIPSRV_ONWIRE_03 to SOMEIPSRV_ONWIRE_04
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.2.3.5	Protocol Version	SOMEIPSRV_ONWIRE_05
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.2.3.6	Interface Version	SOMEIPSRV_ONWIRE_056
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.2.3.7	Message Type	SOMEIPSRV_ONWIRE_07 to SOMEIPSRV_ONWIRE_10
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.2.3.8	Return Code	SOMEIPSRV_ONWIRE_11 to SOMEIPSRV_ONWIRE_12
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.3.1.2	TCP Binding	SOMEIPSRV_RPC_01, SOMEIPSRV_RPC_02, SOMEIPSRV_RPC_17
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.3.5	Fields	SOMEIPSRV_RPC_03, SOMEIPSRV_RPC_11, SOMEIPSRV_RPC_12
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.3.3	Fire&Forget Communication	SOMEIPSRV_RPC_04, SOMEIPSRV_RPC_05
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.3.6.2	Return Code	SOMEIPSRV_RPC_06 to SOMEIPSRV_RPC_10
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.3.1.3	Multiple Service-Instances	SOMEIPSRV_RPC_13, SOMEIPSRV_RPC_14
Example for a Serialization Protocol(SOME/IP) V1.1.0 R4.1 Rev 3	6.3.4.1	Strategy for sending notifications	SOMEIPSRV_RPC_15, SOMEIPSRV_RPC_16

5.1.2 Parameters used in the tests

5.1.2.1 User defined configuration parameters for IUT

Parameter used in test	Description
Listen Time Setting	This is the maximum time interval for which anvl waits for a packet for cases when a certain event has been triggered on the DUT either by some protocol timer or using some external mechanism (script). Default: 10
Tolerance Time Setting	Tolerance time associated with an event. When waiting or listening then this number will be added with the actual wait-time or listen-time. Default: 1
Process Time Setting	This is the time DUT appoximately takes to process packet ANVL uses this process time added with listen time for some time critical situation while listening for DUT Responses. Default: 2
Millisec Tolerance Time Setting	Tolerance time associated with some event in Millisecond. When waiting or listening then this number will be added with the actual wait-time or listen-time. This is currently used in case of DUTs responses are in millisecond interval Default: 100
DUT Supports Error Messages Setting	This is used to indicate whether DUT supports SOME/IP Error messages. Default: false
DUT Supports SD Reception via Unicast Setting	This is Used to indicate whether DUT supports SD message reception via Unicast. Default: true
DUT Supports Multiple Service Instances Setting	This is Used to indicate whether DUT supports Multiple instances of SAME service. Default: false
Server Unknown Method ID Setting	This is a Method ID value that is not implemented in DUT and not listed in xml specification as well. Default: 15

Server Unknown Interface Version Setting	This is a Service Interface version value that is not implemented in DUT and not listed in xml specification as well. Default: 0xFF
Server Unknown Service ID Setting	This is a Service ID value that is not implemented in DUT and not listed in xml specification as well. Default: 0xFF
Server Unknown EventGroup ID Setting	This is an Eventgroup ID value that is not implemented in DUT and not listed in xml specification as well. Default: 50
Server Interface Specification XMLSetting	This is path for SOMEIP Service Interface Specification XML file. Default: “someipserverspec.xml"”

5.1.2.2 User defined configuration parameters for TESTER

Parameter used in test	Description
<DIface-n>	n-th Interface of DUT
<SERVICE-ID-1>	Service ID of Service 1
<SERVICE-ID-2>	Service ID of Service 2
<SERVICE-ID-1-INSTANCE-ID>	Instance ID of Service 1
<SERVICE-ID-1-MAJ-VER>	Major version supported by Service 1
<SERVICE-ID-1-INTF-VER-MAJ>	Interface Version of Service 1.
<SERVICE-ID-2-INTF-VER-MAJ>	Interface Version of Service 2.
<SERVICE-ID-1-TTL>	TTL value of Service 1.
<SERVICE-ID-1-MINOR-VER>	Minor version supported by Service 1
<SERVICE-ID-1-INST-1>	First Instance ID of Service 1.
<EVENT-GROUP-ID-2>	Eventgroup ID 2.
<EVENT-GROUP-ID-1-SI-1>	Eventgroup ID 1 of Service 1.
<SERVICE-ID-1-UDP-PORT>	UDP port number where Service 1 runs.
<SOME_IP_MULTICAST_IP_ADDR>	Multicast Address used by SOME/IP-SD.
<CLIENT1-CURR-REQUEST-ID>	Current Request ID of Client 1.
<EVENT-ID-1-EG-ID-1>	Event ID 1 of Eventgroup ID 1.
<EVENT-ID-2-EG-ID-1>	Event ID 2 of Eventgroup ID 1.
<EVENT-ID-2-EG-ID-2>	Event ID 2 of Eventgroup ID 12
<EVENT-ID-2-EG-ID-1-CYCLIC-TIME>	Cycle Time of Event ID-2 of Eventgroup 1.(Here Event ID 1 is assumed to be a cyclic event)
<EVENT-GROUP-ID-1-SI-1-MulticastAddr>	Muticast Address of Eventgroup 1 of Service 1 where the event and notifications will be multicast.

<UNKNOWN-SERVICE-ID>	Invalid Service ID.
<UNKNOWN-EVENT-GROUP-ID>	Invalid Eventgroup ID.
<UNKNOWN-METHOD-ID-SI-1>	Invalid Method ID of Service ID 1.
<UNKNOWN-INTF-VER-MAJ-SI-1>	Invalid Interface Version of Service ID 1.
<SERVICE-ID-1-REP-BASE-INTV>	Repetition Base Interval of Service 1.
<SERVICE-ID-1-TOTAL-REP-INTV>	Total Repetition Period of Service 1. Calculated using Repetition max value as - (Repetition_base_intv) * ((2 ^ repetiton_max) - 1)
<SERVICE-ID-1-INITIAL-WAIT-TIME>	Initial Wait time for Service 1.
<SERVICE-ID-1-CYCLE-INTV>	Offer Cycle interval of Service 1.
<METHOD-ID-1-SI-1>	First InOut Type Method Listed in XML file for Service 1.
<METHOD-ID-1-SI-2>	First InOut Type Method Listed in XML file for Service 2.
<SERVICE-ID-2-TRANSPORT-PORT>	Transport port of Service ID 2.(UDP/TCP)
<SERVICE-ID-2-INITIAL-WAIT-TIME>	Time taken by the DUT before Sending Offer Service message.
<SERVER1-IP-ADDR>	IP Address of SOME/IP-SD Server 1.
<CLIENT1-IP-ADDR>	ANVL client1 IP address.
<CLIENT1-UDP-PORT>	ANVL client1 UDP port.
<FOREGROUND>	Boolean value set to false to indicate script running in foreground.
<BACKGROUND>	Boolean value set to true to indicate script running in background.
<ParamToleranceTime>	This is tolerance time taken from parameter file.
<CLIENT1-LISTEN-TIME>	Listen time used by ANVL taken from parameter file.
<BUSY-WAIT>	It is to indicate ANVL will be listening to incoming packets while waiting.
<LAZY-WAIT>	It is to indicate ANVL will not be listening to incoming packets while waiting.
ListenTime	This is the maximum time interval for which anvl waits for a packet for cases when a certain event has been triggered on the DUT either by some protocol timer or using some external mechanism (script). [Default: 10 second]
ToleranceTime	Tolerance time associated with an event. When waiting or listening then this number will be added with the actual wait-time or listen-time. [Default: 1 second]

5.1.3 Terminology used in Test Procedure

Name	Description
TESTER	Entity which is responsible for validating the Device under Test (DUT)
DUT	Device under Test

5.1.4 Specification of the SOMEIP TestStub Enhanced Testability Service (ETS)

5.1.4.1 Introduction

5.1.4.1.1 Overview

Defined interfaces are needed to test SOME/IP [1]. In order to allow for generic testing tools and test cases, this interface needs to be defined and possibly standardized. This interface and the functionality behind it are called *Enhanced Testability Service (ETS)*.

The protocol parts currently being addressed by the Enhanced Testability Service include:

- SOME/IP Stack - Service Discovery
- SOME/IP Stack - Serialization
- SOME/IP Stack - Remote Procedure Call
- SOME/IP Stack - Service Discovery
- SOME/IP Stack - Publish/Subscribe

The Enhanced Testability Service further allows different categories of tests, e.g. when used in component testing scenarios for devices under test (DUTs). These include positive tests (testing using valid messages), negative tests (testing error handling), load testing, and regression testing.

5.1.4.1.2 References

[1] SOME/IP Protocol Specification AUTOSAR FO Release 1.1.0

[2] SOME/IP Service Discovery Protocol Specification AUTOSAR FO Release 1.1.0

5.1.4.2 Enhanced Testability Service

Table 1: List of Methods

Method	ID	Fire&Forget
checkByteOrder	31 (0x1F)	

clientServiceActivate	47 (0x2F)	X
clientServiceDeactivate	48 (0x30)	X
clientServiceCallEchoUINT8Array	49 (0x31)	
clientServiceSubscribeEventgroup	50 (0x32)	X
echoCommonDatatypes	35 (0x23)	
echoENUM	23 (0x17)	
echoExtendedDatatypes	36 (0x24)	
echoExtendedDatatypes16BitLengthAndTypeField	57 (0x39)	
echoExtendedDatatypes8BitLengthAndTypeField	56 (0x38)	
echoFLOAT64	18 (0x12)	
echoInt64	52 (0x34)	
echoINT8	14 (0x0E)	
echoStaticUINT8Array	54 (0x36)	
echoSTRUCT	24 (0x18)	
echoSTRUCTSimple	64 (0x40)	
echoTYPEDEF	26 (0x1A)	
echoUINT64	51 (0x33)	
echoUINT8	8 (0x08)	
echoUINT8Array	9 (0x09)	
echoUINT8Array8BitLength	62 (0x3E)	
echoUINT8Array16BitLength	63 (0x3F)	
echoUINT8Array2Dim	53 (0x35)	
echoUINT8ArrayMinSize	55 (0x37)	
echoUINT8E2E	11 (0x0B)	
echoUINT8RELIABLE	10 (0x0A)	
echoUNION	25 (0x19)	
echoUTF16DYNAMIC	22 (0x16)	
echoUTF16FIXED	20 (0x14)	
echoUTF8DYNAMIC	21 (0x15)	

echoUTF8FIXED	19 (0x13)	
resetInterface	1 (0x01)	X
suspendInterface	2 (0x02)	X
triggerEventUINT8	3 (0x03)	X
triggerEventUINT8Array	4 (0x04)	X
triggerEventUINT8E2E	6 (0x06)	X
triggerEventUINT8Reliable	5 (0x05)	X
triggerEventUINT8Multicast	58 (0x3A)	X
clientServiceGetLastValueOfEventTCP	59 (0x3B)	
clientServiceGetLastValueOfEventUDPUnicast	60 (0x3C)	
clientServiceGetLastValueOfEventUDPMulticast	61 (0x3D)	
echoBitfields	65 (0x41)	

Table 2: List of Events and Fields

Type	Name	ID	Eventgroups		
			0x0002	0x0005	0x0006
Event	TestEventUINT8	0x8001	x	x	
Event	TestEventUINT8Array	0x8002	x	x	
Event	TestEventUINT8E2E	0x8004	x	x	
Event	TestEventUINT8Reliable	0x8003	x		
Event	TestEventUINT8Multicast	0x800B	x	x	x
Field	InterfaceVersion	0x8005 (Notify)	x	x	
		0x25 (Getter)	x	x	
Field	TestFieldUINT8	0x8006 (Notify)	x	x	
		0x26 (Getter)	x	x	
		0x27 (Setter)	x	x	
Field	TestFieldUINT8Array	0x8007 (Notify)	x	x	
		0x28 (Getter)	x	x	
		0x29 (Setter)	x	x	
Field	TestFieldUINT8Reliable	0x8008 (Notify)	x		
		0x2A (Getter)	x		
		0x2B (Setter)	x		

5.1.4.3 SOME/IP Service Discovery (SOME/IP-SD)

Several methods of the Enhanced Testability Service are used to test SOME/IP-SD.

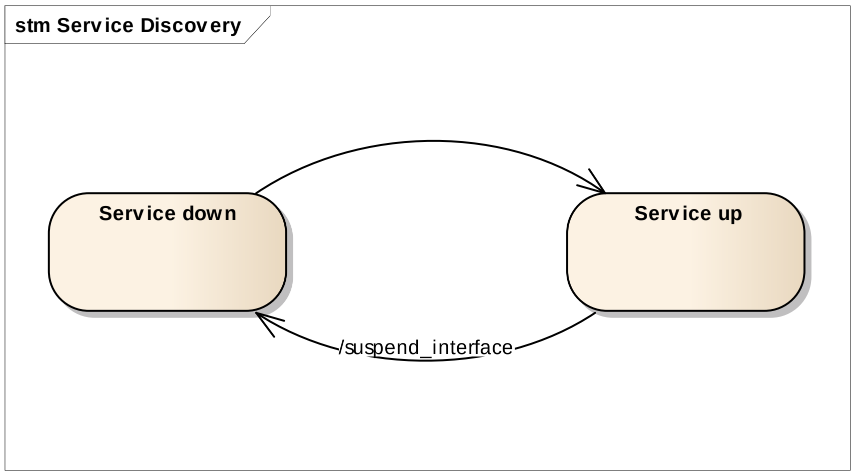


Figure 1: SOME/IP-SD state machine of testing methods

Method	ID
resetInterface	1 (0x01)

This method resets the interface to default values.

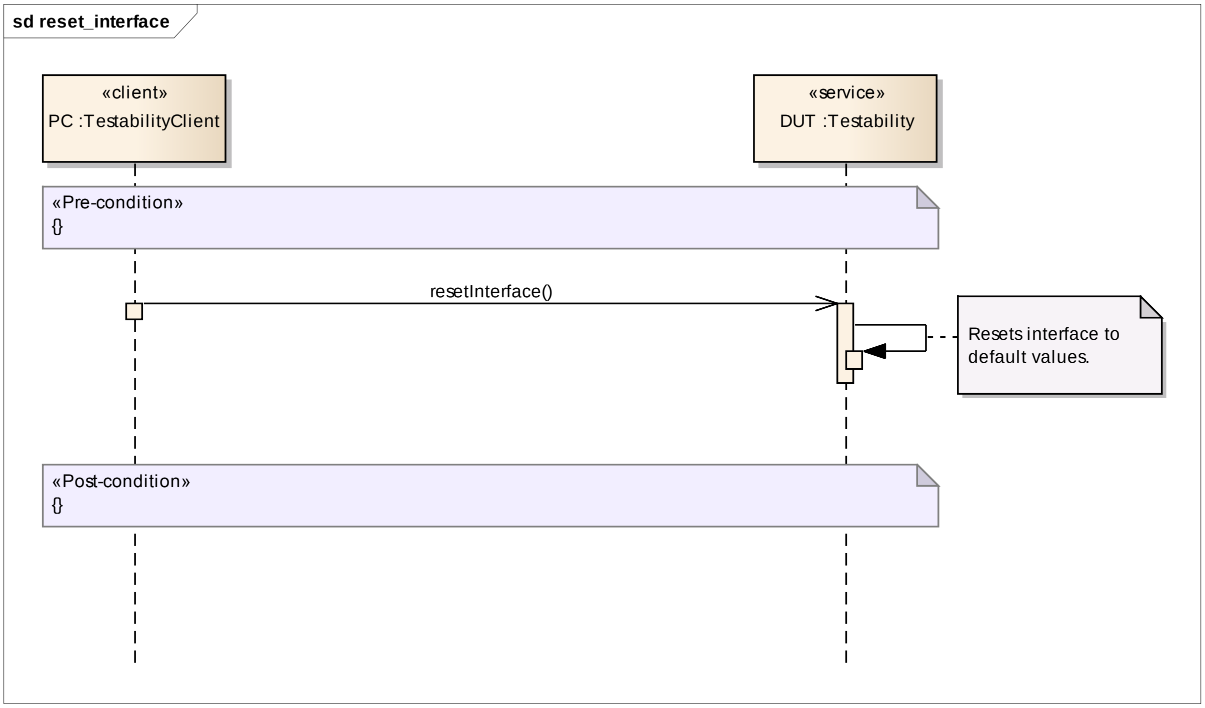


Figure 2: resetInterface()

5.1.4.3.1 Suspending an SOME/IP Interface: suspendInterface()

Method	ID
suspendInterface	2 (0x02)

This method allows to suspend the Enhanced Testability Service; thus, forcing the DUT to stop offering the service and start to reoffer the service after a given time.

The input parameters include:

- Start – wait time in seconds before the service shall be stopped
- Duration – wait time in seconds during between stopping the service and reoffering it again.

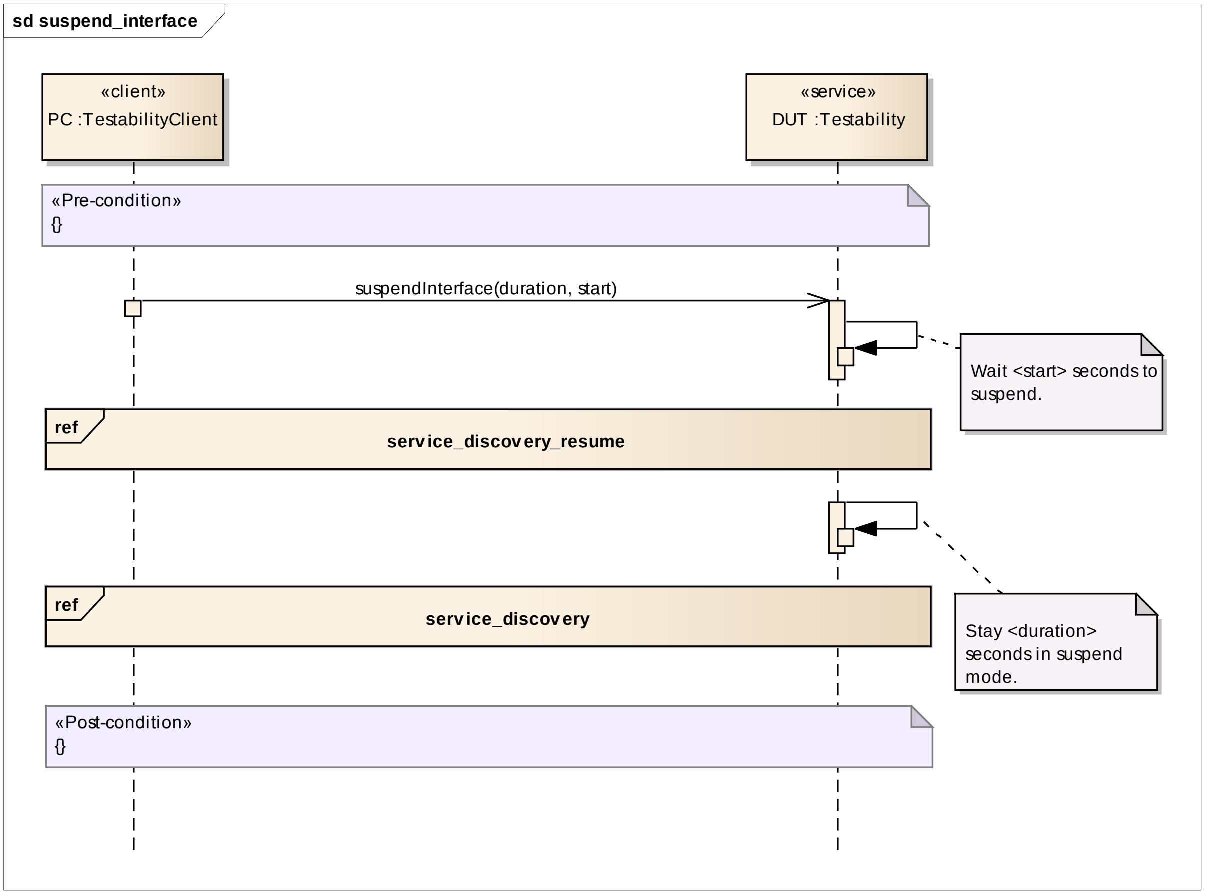


Figure 3: suspendInterface()

5.1.4.3.2 Testing Client Interaction

In order to test the correct function as a client, the Enhanced Testability Service supports the following methods:

Method	ID
clientServiceActivate	47 (0x2F)
clientServiceDeactivate	48 (0x30)
clientServiceCallEchoUINT8Array	49 (0x31)
clientServiceSubscribeEventgroup	50 (0x32)
clientServiceGetLastValueOfEventTCP	59 (0x3B)
clientServiceGetLastValueOfEventUDPUnicast	60 (0x3C)
clientServiceGetLastValueOfEventUDPMulticast	61 (0x3D)

The commands clientServiceActivate and clientServiceDeactivate shall instruct the DUT, if it shall start its SOME/IP client; thus, they control when the system shall start finding the Enhanced Testability Service with the special configurable Instance ID (for example 0x00f4).

Using clientServiceEchoUINT8Array the system shall call back the echoUINT8Array method with the parameter given and shall return the returned value.

The clientServiceSubscribeEventgroup shall trigger the subscription behavior in Service Discovery.

To ease testing all methods shall support a wait time, so that the reaction of the system can be delayed.

Following methods return the last received value of TestEventUINT8Reliable, TestEventUINT8, and TestEventUINT8Multicast respectively:

- clientServiceGetLastValueOfEventTCP
- clientServiceGetLastValueOfEventUDPUnicast
- clientServiceGetLastValueOfEventUDPMulticast

5.1.4.4 SOME/IP Serialization

5.1.4.4.1 Checking the Byte Order: checkByteOrder()

Method	ID
checkByteOrder	31 (0x1F)

This method is used to test the correct handling of the byte order.

The input parameters include a uint8 and a big endian uint16, which shall be added and returned as output parameter in uint32 big endian.

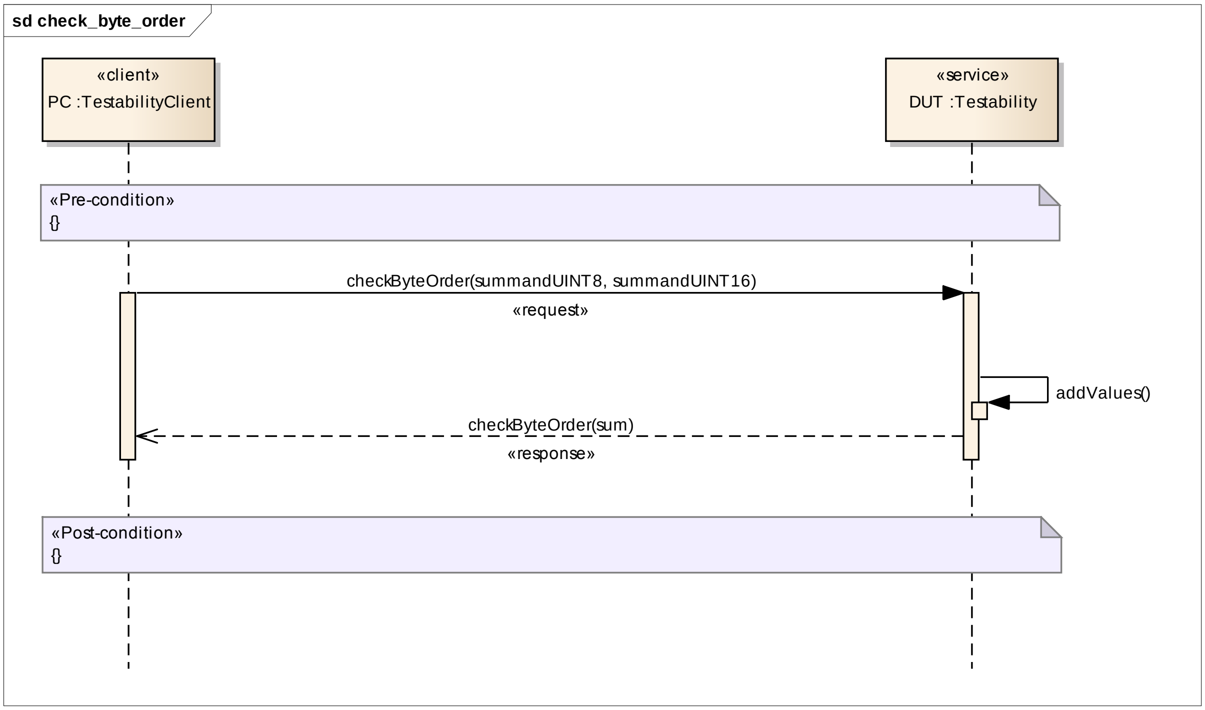


Figure 4: checkByteOrder()

5.1.4.4.1 Common Data Types: echoCommonDatatypes()

Method	ID
echoCommonDatatypes	35 (0x23)

This method can be used to test the common data types.

The input parameters shall be echoed back in reversed order.

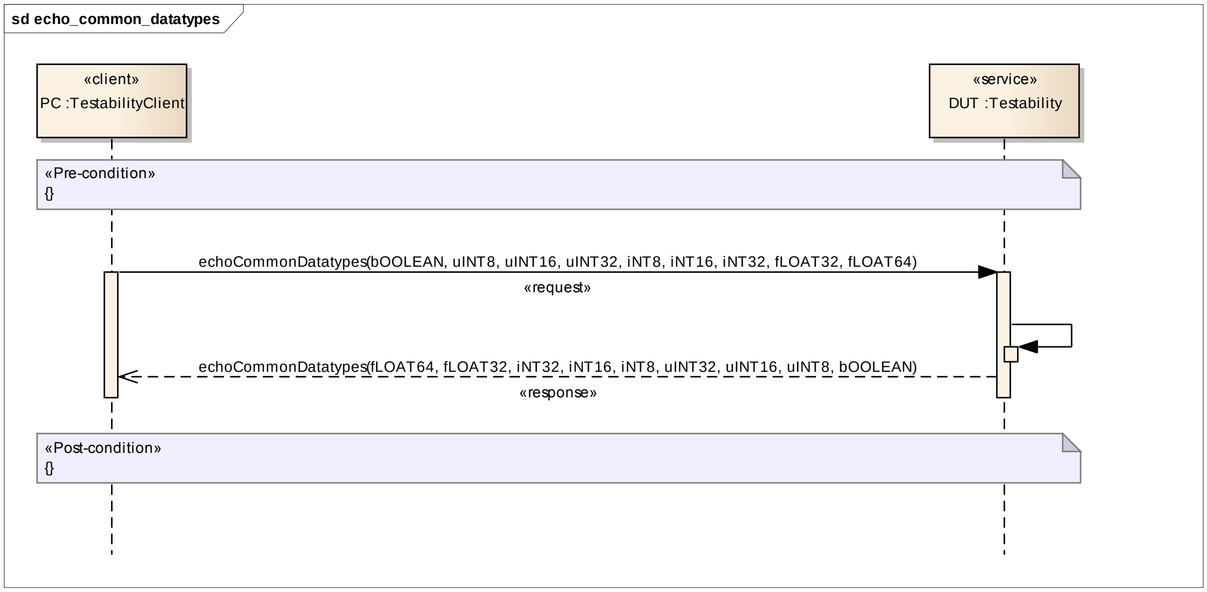


Figure 5: echoCommonDatatypes()

5.1.4.4.2 Complex Data Types: echoExtendedDatatypes

Method	ID
echoExtendedDatatypes	36 (0x24)
echoExtendedDatatypes16BitLengthAndTypeField	57 (0x39)
echoExtendedDatatypes8BitLengthAndTypeField	56 (0x38)

This method is used to test the complex data types.

The regular variant uses the recommended size of 32bit for length fields, type fields, and array length fields.

The other variants change this to 16 bits or 8 bits respectively.

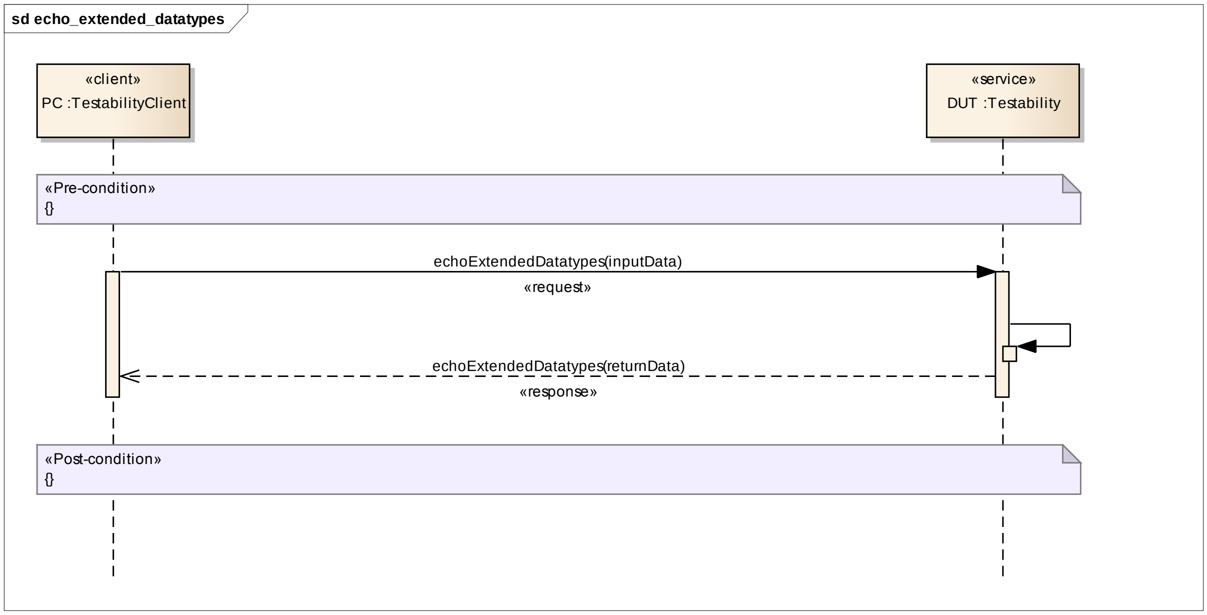


Figure 6: echoExtendedDatatypes()

5.1.4.4.3 Echoing data types

Method	ID
echoCommonDatatypes	35 (0x23)
echoENUM	23 (0x17)
echoExtendedDatatypes	36 (0x24)
echoExtendedDatatypes16BitLengthAndTypeField	57 (0x39)
echoExtendedDatatypes8BitLengthAndTypeField	56 (0x38)
echoFLOAT64	18 (0x12)
echoInt64	52 (0x34)
echoINT8	14 (0x0E)
echoStaticUINT8Array	54 (0x36)
echoSTRUCT	24 (0x18)
echoSTRUCTSimple	64 (0x40)
echoTYPEDEF	26 (0x1A)
echoUINT64	51 (0x33)
echoUINT8	8 (0x08)
echoUINT8Array	9 (0x09)
echoUINT8Array8BitLength	62 (0x3E)
echoUINT8Array16BitLength	63 (0x3F)
echoUINT8Array2Dim	53 (0x35)
echoUINT8ArrayMinSize	55 (0x37)
echoUINT8E2E	11 (0x0B)
echoUINT8RELIABLE	10 (0x0A)
echoUNION	25 (0x19)
echoUTF16DYNAMIC	22 (0x16)

echoUTF16FIXED	20 (0x14)
echoUTF8DYNAMIC	21 (0x15)
echoUTF8FIXED	19 (0x13)
echoBitfields	65 (0x41)

A set of methods exist to test a multitude of data types (such as bool, uint8, uint16, uint32, …). Only three examples are shown in the diagram. Keep in mind that some of the echo-routines echo back the parameters in another order as the parameters coming in. The configuration (FIBEX/ARXML) describes this in detail.

The method echoUINT8() returns the UINT8 value. In this case it sent via UDP.

The echoUDP8Reliable() uses TCP instead.

The echoUINT8E2E() uses End-to-End-protection.

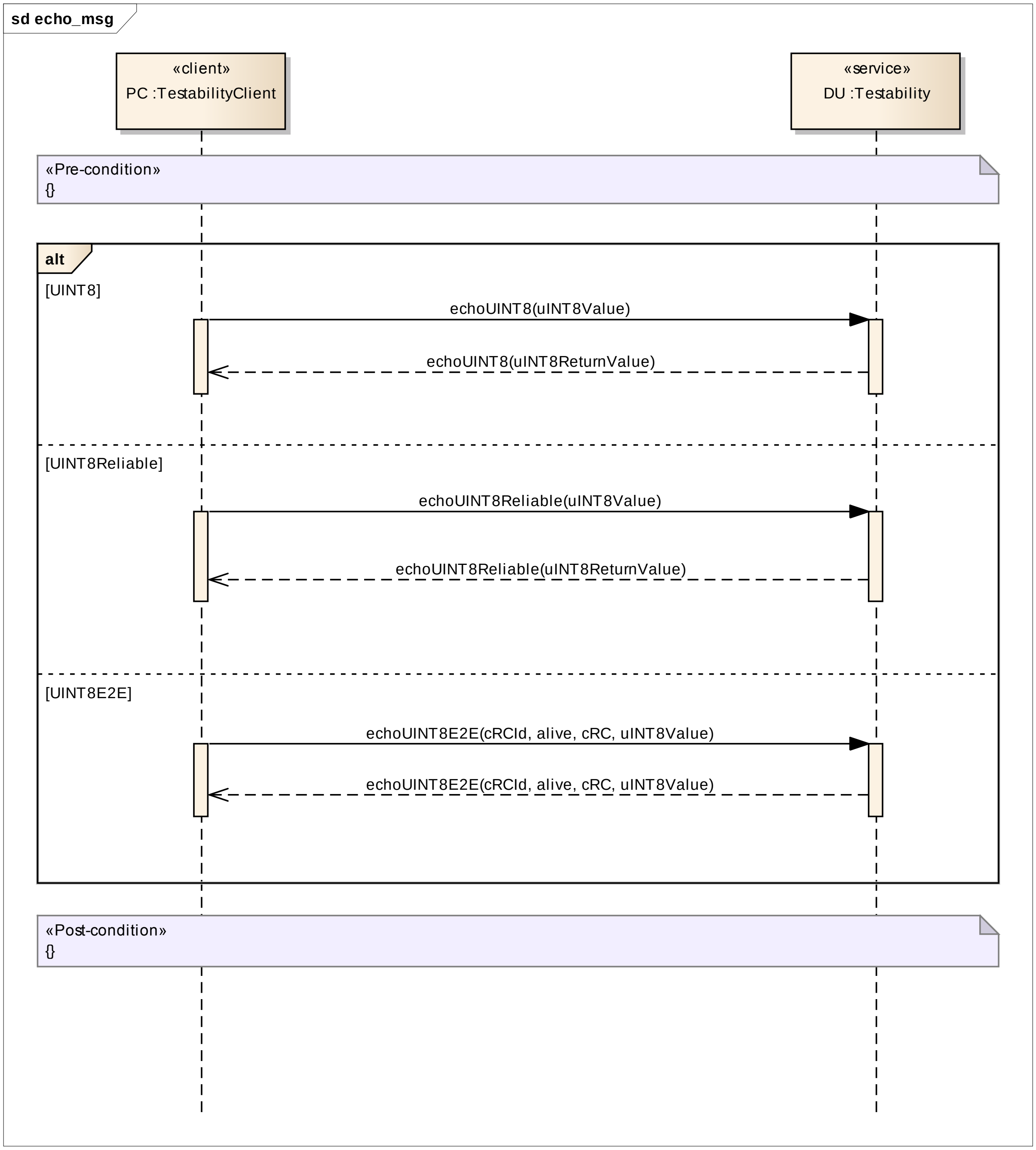


Figure 7: echoUINT8(),echUINT8reliable(),echoUINT8E2E()

5.1.4.4.4 Testing Events

Method	ID
triggerEventUINT8	3 (0x03)
triggerEventUINT8Array	4 (0x04)

triggerEventUINT8E2E	6 (0x06)
triggerEventUINT8Reliable	5 (0x05)
triggerEventUINT8Multicast	58 (0x3A)

After the method triggerEventUINT8Reliable (start, duration, debounceTime) was invoked, the service Testability waits <start> seconds and triggers a periodical event testEventUINT8Reliable(uINT8Value) with the passed duration and debounce time. The transport protocol of the event message is TCP.

There are more test events and the corresponding trigger methods available also for other passed datatypes. The diagram below only shows one example.

The full list can be found in the service catalog.

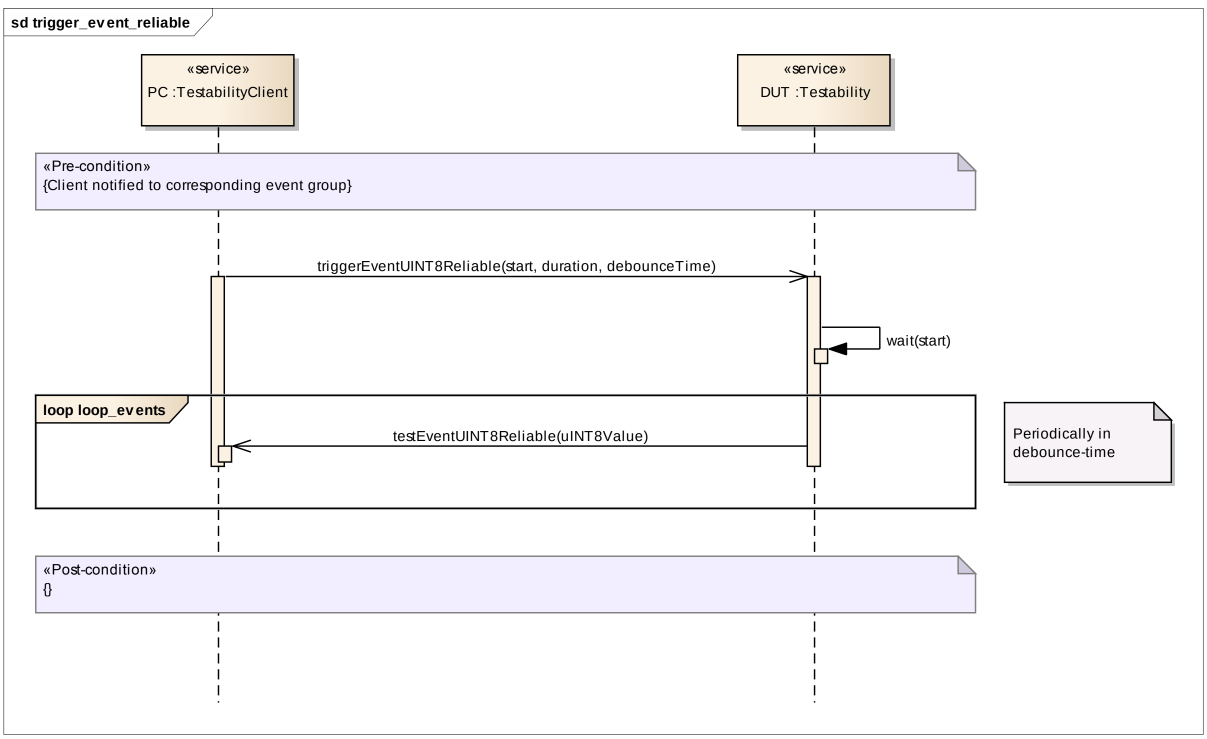


Figure 8: triggerEventUINT8Reliable()

After the method triggerEventUINT8Array(start, duration, debounceTime) was invoked, the service Testability waits <start> seconds and triggers a periodical event testEventUINT8Array(uINT8Array) with the passed duration and debounce time.

The transport protocol for both the trigger method and the event is UDP.

There are more test events and the corresponding trigger methods available also for other passed datatypes. The diagram below only shows one example.

The full list can be found in the service catalog.

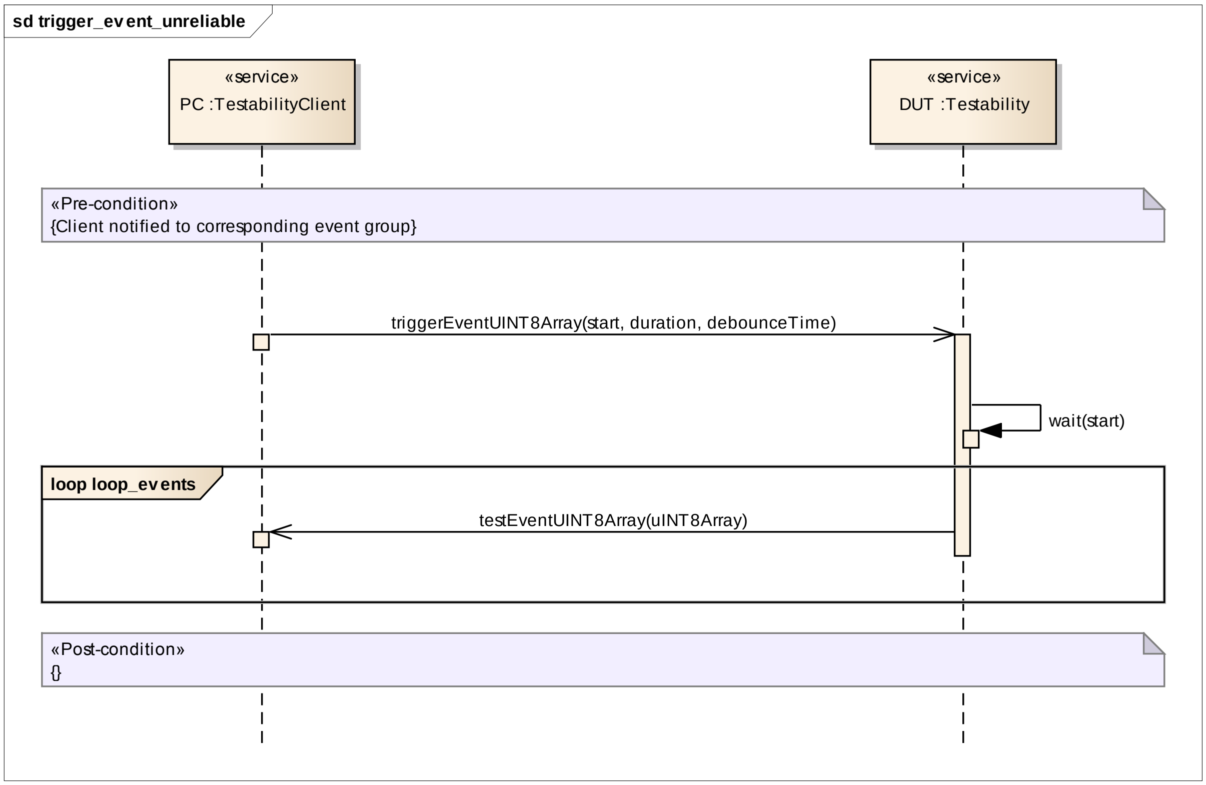


Figure 9: triggerEventUINT8Array()

The other triggerEvent methods run as follows: E2E methods always use End-to-End protection.

5.1.4.4.5 Testing fields (getter, setter, notify)

Different test fields are available to test the corresponding getter methods, both unreliable via UDP and reliable via TCP. The full list of test fields is documented in the service catalog.

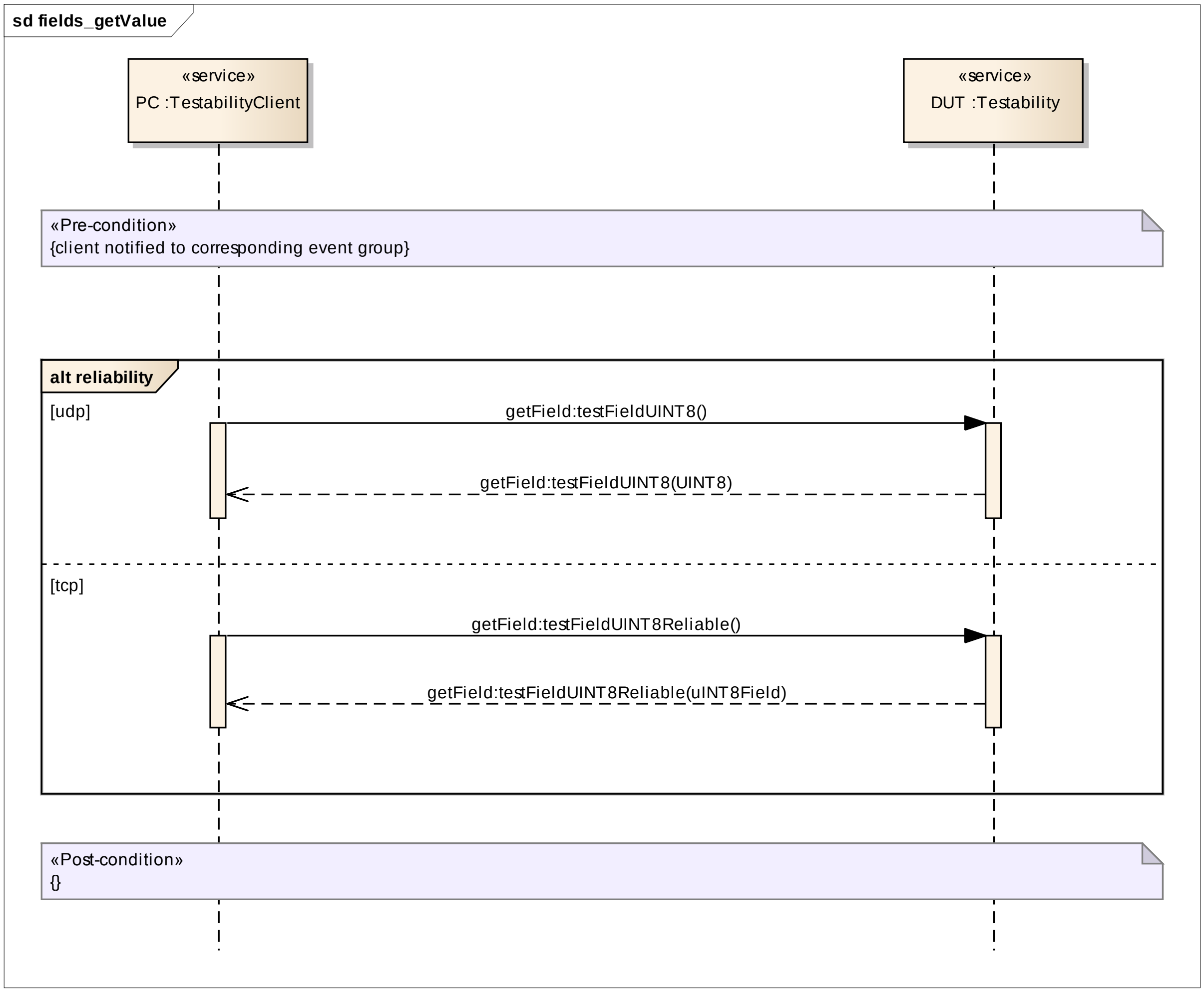


Figure 10: getField:testFieldUINT8(),getField:testFieldUINT8Reliable()

Different test fields are available to to test the corresponding setter methods, both unreliable via UDP and reliable via TCP. The full list of test fields is documented in the service catalog.

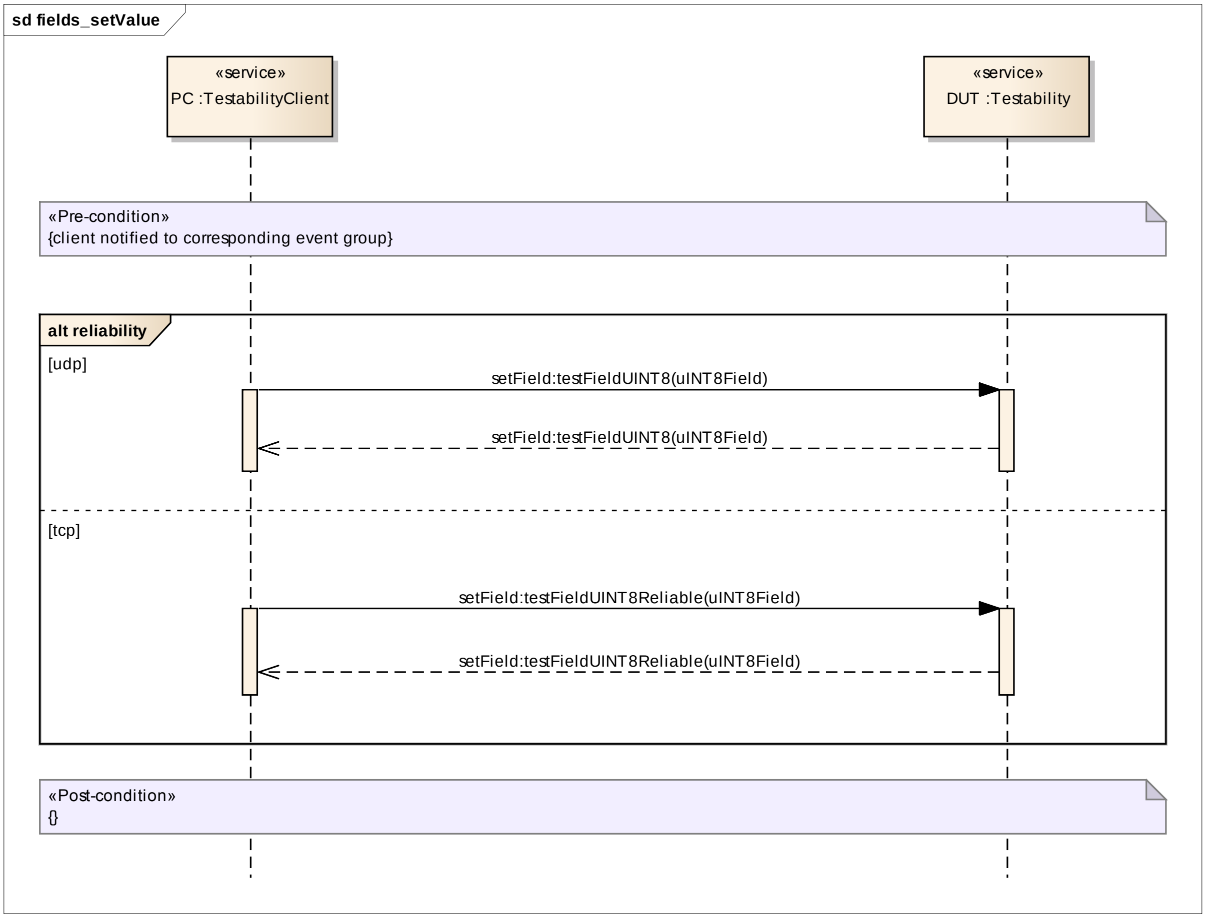


Figure 11: setField:testFieldUINT8(),setField:testFieldUINT8Reliable()

Different test fields are available to test the corresponding notifier events, both unreliable via UDP and reliable via TCP. Additionally, the value of the test fields is updated periodically to trigger a notification event. The full list of test fields is documented in the service catalog.

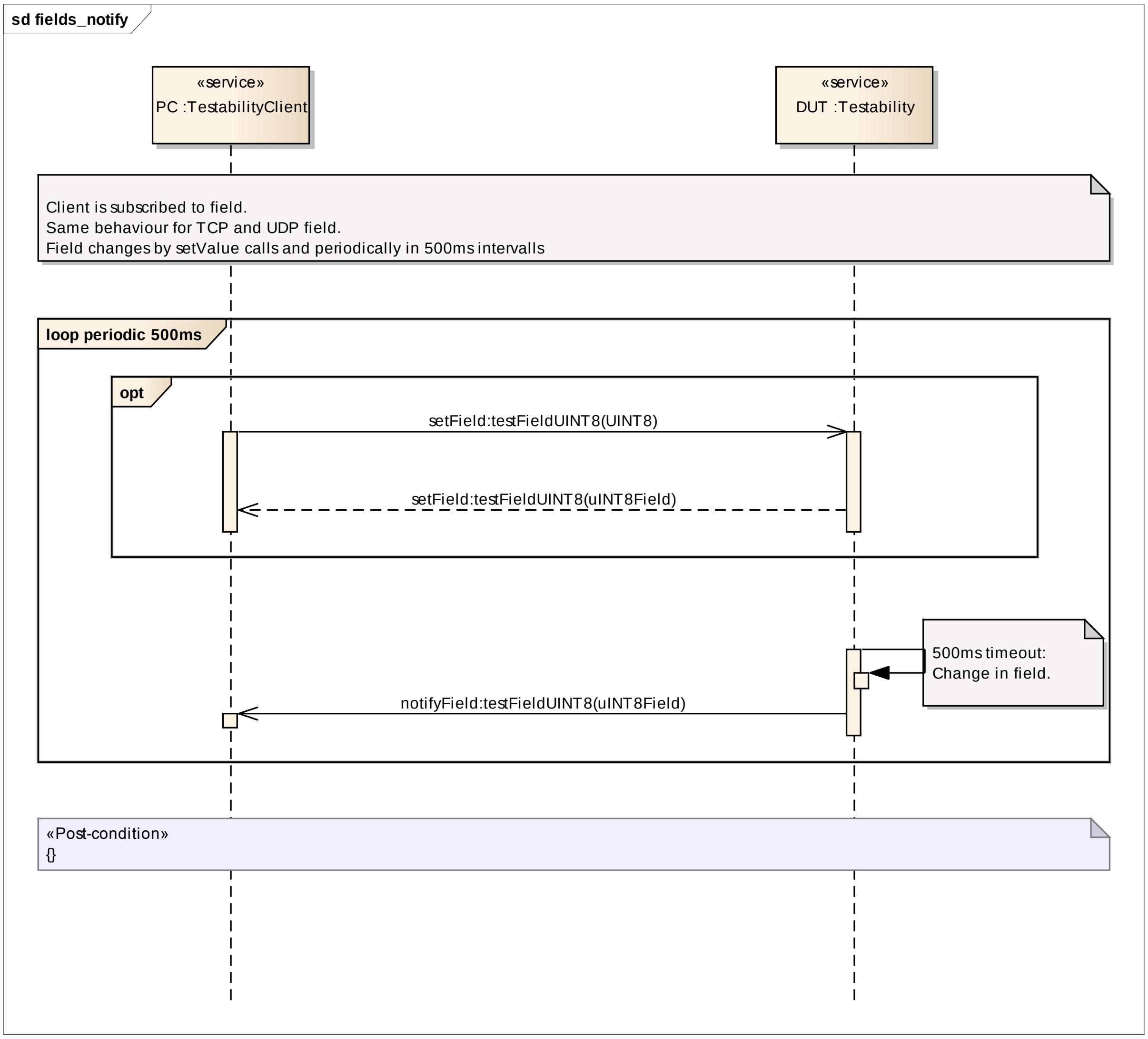


Figure 12: notifyField:testFieldUINT8()

5.1.4.5 SOME/IP Service Discovery

Testing SOME/IP-SD does not require additional testing methods but can be achieved with the above mentioned commands, especially from Sections 5.1.4.3.2 and 5.1.4.4.

5.1.5 Test Cases SOME/IP Server

5.1.5.1 Setup Verification

SOMEIPSRV_SETUP_01: Service Discovery Offer Message

Synopsis	Setup Verification for SOME/IP and SOME/IP-SD TESTER expects Service Discovery Offer Message from DUT
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `- Service ID : <SERVICE-ID-1>` `- Number Of Instances : 1` `2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `3. DUT: Sends SOME/IP Notification Message` `4. TESTER: Verify that received SOME/IP Notification Message contains:` `- Entry Type is set to 0x01` `5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `- Service ID : <SERVICE-ID-1>`
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Entry Type is set to 0x01
Reference	Setup Verification (MUST)
Notes	

SOMEIPSRV_SETUP_02: DUT Sends SOME/IP Response Message to SOME/IP Request Message

Synopsis	Setup Verification
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 4. DUT: Sends SOME/IP Response Message 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Response Message
Reference	Setup Verification (MUST)
Notes	

5.1.5.2 Message Format

SOMEIPSRV_FORMAT_01: Client ID

Synopsis	The Client ID shall be set statically to 0x0000
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Client ID is set to 0x0000 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Client ID is set to 0x0000
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00033] Page 23 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_02: Session ID

Synopsis	After initialization of the Service Discovery Module, the Session ID for messages sent by the local ECU shall be 0x0001.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto (<SERVICE-ID-1-INITIAL-WAIT-TIME> + <CLIENT1-LISTEN-TIME>) second) on <DIface-0> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Session ID is set to 0x0001 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Session ID is set to 0x0001
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00034] Page 23 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_03: Protocol Version

Synopsis	The value for the Protocol Version field shall be statically set to 0x01
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Protocol Version is set to 0x01 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Protocol Version is set to 0x01
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00141] Page 24 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_04: Interface Version

Synopsis	The value for the Interface Version field shall be statically set to 0x01
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Interface Version is set to 0x01 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Interface Version is set to 0x01
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00143] Page 24 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_05: Message Type

Synopsis	The value for the Message Type field shall be statically set to 0x02
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>` `        - Number Of Instances : 1` `2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `3. DUT: Sends SOME/IP Notification Message` `4. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>`
Pass Criteria	3. DUT: Sends SOME/IP Notification Message
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00145] Page 24 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_06: Return Code

Synopsis	The Return Code field shall be statically set to 0x00 Note: This test case is for SD message
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>` `        - Number Of Instances : 1` `2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `3. DUT: Sends SOME/IP Notification Message` `4. TESTER: Verify that received SOME/IP Notification Message contains:` `        - Return Code is set to 0x00` `5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>`
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Return Code is set to 0x00
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00147] Page 25 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_07: Reboot Flag

Synopsis	The Reboot Flag shall be set to '1' for all messages after reboot until the Session ID of the Request ID field wraps and thus starts with 0x0001 again. After that the Reboot Flag shall be set to '0'.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>` `        - Number Of Instances : 1` `2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `3. DUT: Sends SOME/IP Notification Message` `4. TESTER: Verify that received SOME/IP Notification Message contains:` `        - Reboot Flag bit is set to 1` `5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>`
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Reboot Flag bit is set to 1
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00151] Page 25 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_08: Unicast Flag

Synopsis	The Unicast Flag of the Flag field shall be set to Unicast Flag and shall be set to '1', meaning: This ECU supports receiving Unicast messages.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Unicast Flag bit is set to 1 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Unicast Flag bit is set to 1
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00153] Page 26 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_09: Undefined bits in the Flag field

Synopsis	Undefined bits within the Flag field shall be statically set to '0'
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `- Service ID : <SERVICE-ID-1>` `- Number Of Instances : 1` `2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `3. DUT: Sends SOME/IP Notification Message` `4. TESTER: Verify that received SOME/IP Notification Message contains:` `- Six Flag bits other than Reboot and Unicast bits is set to 0` `5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `- Service ID : <SERVICE-ID-1>`
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Six Flag bits other than Reboot and Unicast bits is set to 0
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00154] Page 26 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_10: Reserved bits

Synopsis	All bits of the Reserved field shall be statically set to 0 binary.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Reserved is set to 0x000000 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Reserved is set to 0x000000
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00156] Page 26 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_11: Length of the Type 1 Entry

Synopsis	The length of the Type 1 Entry shall be 16 bytes Note : Offer Service is unnder Entry Type 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Entry Type is set to 0x01 - Entry Length is set to (NumberOfEntries*16) 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 4. TESTER: Verify that received SOME/IP Notification Message contains: - Entry Type is set to 0x01 - Entry Length is set to (NumberOfEntries*16)
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00160] Page 27 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_12: Index First Option Run of the Type 1 Entry

Synopsis	The "Index First Option Run" field of the Type 1 Entry format layout shall carry the index of the first option of the first option run of this entry in the option array. Note : Offer Service is under Entry Type 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Index 1st Option in Entry Array is set to 0 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Index 1st Option in Entry Array is set to 0
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00164] Page 28 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_13: Number of Option 1 of the Type 1 Entry

Synopsis	[SWS_SD_00168] The "Number of Option 1" of the Type 1 Entry format layout shall carry the number of options the first option run uses. Note : Offer Service is under Entry Type 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Num of Opt 1 in Entry Array is greater than or equal to 1 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Num of Opt 1 in Entry Array is greater than or equal to 1
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00168] Page 28 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_14: Service ID field of the Type 1 Entry

Synopsis	The Service ID field of the Type 1 Entry format layout shall carry the Service ID of the service, statically configured using the parameter SdServerServiceID and SdClientServiceID, depending on being a server or client entry. Note : Offer Service is unnder Entry Type 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>` `        - Number Of Instances : 1` `2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through` `        <DIface-0> containing :` `        - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE` `        - Service ID in Entry Array set to <SERVICE-ID-1>` `3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `        - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE` `        - Service ID in Entry Array set to <SERVICE-ID-1>` `4. DUT: Sends SOME/IP Notification Message` `5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>`
Pass Criteria	4. DUT: Sends SOME/IP Notification Message
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00173] Page 29 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_15: Instance ID field of the Type 1 Entry

Synopsis	The Instance ID field of the Type 1 Entry format layout shall carry the Instance ID of the service, statically configured using the parameter SdServerServiceInstanceID and SdClientServiceInstanceID, depending on being a server or client entry. Note : Offer Service is unnder Entry Type 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Instance ID in Entry Array is set to <SERVICE-ID-1-INSTANCE-ID> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Instance ID in Entry Array is set to <SERVICE-ID-1-INSTANCE-ID>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00175] Page 29 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_16: Major Version field of the Type 1 Entry

Synopsis	The Major Version field of the Type 1 Entry format layout shall carry the SdServerServiceMajorVersion and SdClientServiceMajorVersion, depending on being a server or client entry. Note : Offer Service is unnder Entry Type 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Major Version in Entry Array is set to <SERVICE-ID-1-MAJ-VER> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Major Version in Entry Array is set to <SERVICE-ID-1-MAJ-VER>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00178] Page 29 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_17: TTL field of the Type 1 Entry

Synopsis	The TTL field of the Type 1 Entry format layout defines the lifetime of the entry in seconds configured using the parameter SdServerTimerTTL and SdClientTimerTTL, except for Stop-Entries, which have a TTL of 0 Note : Offer Service is unnder Entry Type 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to <SERVICE-ID-1-TTL> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to <SERVICE-ID-1-TTL>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00180] Page 29 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_18: Minor Version field of the Type 1 Entry

Synopsis	The Minor Version field of the Type 1 Entry format layout shall carry the SdServerServiceMinorVersion and SdClientServiceMinorVersion. Note : Offer Service is unnder Entry Type 1
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Minor Version in Entry Array is set to <SERVICE-ID-1-MINOR-VER> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Minor Version in Entry Array is set to <SERVICE-ID-1-MINOR-VER>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00182] Page 30 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_19: Type field of the Type 2 Entry

Synopsis	The Type field of the Type 2 Entry format layout shall carry one of the following values, depending of the purpose of the sent message: 0x06 to encode SubscribeEventgroup and StopSubscribeEventgroup 0x07 to encode SubscribeEventgroupAck and SubscribeEventgroupNack Note : SubscribeEventgroupAck is unnder Entry Type 2 Test case is to verify SubscribeEventgroupAck
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `- Service ID : <SERVICE-ID-1>` `- Number Of Instances : 1` `2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through` `<DIface-0> containing :` `- Entry Type set to SOMEIP_ENTRY_FIND_SERVICE` `- Service ID in Entry Array set to <SERVICE-ID-1>` `3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `- Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE` `- Service ID in Entry Array set to <SERVICE-ID-1>` `4. DUT: Sends SOME/IP Notification Message` `5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through` `<DIface-0> containing :` `- Entry Type set to SOMEIP_ENTRY_SUBSCRIBE` `- Service ID in Entry Array set to <SERVICE-ID-1>` `- EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1>` `6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `- Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK` `- Service ID in Entry Array set to <SERVICE-ID-1>` `- SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1>` `7. DUT: Sends SOME/IP Notification Message` `8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `- Service ID : <SERVICE-ID-1>`
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00385] Page 30 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_20: Length of Type 2 Entries

Synopsis	The length of Type 2 Entries shall be 16 bytes. Note : SubscribeEventgroupAck is unnder Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Entry Length is set to (NumberOfEntries*16) 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Entry Length is set to (NumberOfEntries*16)
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00183] Page 30 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_21: Index First Option Run field of the Type 2 Entry

Synopsis	The "Index First Option Run" field of the Type 2 Entry format layout shall carry the index of the first option of the first option run of this entry in the option array. Note : SubscribeEventgroupAck is under Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Index 1st Option in Entry Array is set to 0 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Index 1st Option in Entry Array is set to 0
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00386] Page 31 'Message format' (MUST)

SOMEIPSRV_FORMAT_22: Number of Option 1 of the Type 2 Entry

Synopsis	The "Number of Option 1" of the Type 2 Entry format layout shall carry the number of options the first option run uses. Note : SubscribeEventgroupAck is under Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Num of Opt 1 in Entry Array is greater than or equal to 1 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Num of Opt 1 in Entry Array is greater than or equal to 1
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3

	[SWS_SD_00188] Page 31 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_23: Service ID field of the Type 2 Entry

Synopsis	The Service ID field of the Type 2 Entry format layout shall carry the Service ID of the eventgroups service, statically configured using the parameter SdServerServiceID and SdClientServiceID, depending on being a server or client entry. Note : SubscribeEventgroupAck is under Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00193] Page 32 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_24: Instance ID field of the Type 2 Entry

Synopsis	The Instance ID field of the Type 2 Entry format layout shall carry the Instance ID of the eventgroups service statically configured using the parameter SdServerServiceInstanceID and SdClientServiceInstanceID, depending on being a server or client entry. Note : SubscribeEventgroupAck is unnder Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Instance ID in Entry Array is set to <SERVICE-ID-1-INSTANCE-ID> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Instance ID in Entry Array is set to <SERVICE-ID-1-INSTANCE-ID>

Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00195] Page 32 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_25: Major Version field of the Type 2 Entry

Synopsis	The Major Version field of the Type 2 Entry format layout shall carry the SdServerServiceMajorVersion and SdClientServiceMajorVersion, depending on being a server or client entry. Note : SubscribeEventgroupAck is under Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Major Version in Entry Array is set to <SERVICE-ID-1-MAJ-VER> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>

Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Major Version in Entry Array is set to <SERVICE-ID-1-MAJ-VER>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00198] Page 32 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_26: TTL field of the Type 2 Entry Entry

Synopsis	The TTL field of the Type 2 Entry Entry format layout defines the lifetime of the entry in seconds configured using the parameter SdServerTimerTTL and SdClientTimerTTL, except for Stop- or Nack-Entries, which use a TTL of 0. Note : SubscribeEventgroupAck is under Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message

	8. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to <SERVICE-ID-1-TTL> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to <SERVICE-ID-1-TTL>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00200] Page 32 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_27: Reserved field, which follows the TTL field of the Type 2 Entry

Synopsis	The Reserved field, which follows the TTL field of the Type 2 Entry format layout, shall be statically set to 0x0000. Note : SubscribeEventgroupAck is under Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK

	- Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Reserved in Entry Array is set to 0x0000 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Reserved in Entry Array is set to 0x0000
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00202] Page 33 'Message format' (MUST)
Notes	

SOMEIPSRV_FORMAT_28: Eventgroup ID field of the Type 2 Entry

Synopsis	The Eventgroup ID field of the Type 2 Entry format layout shall carry the ID of an Eventgroup, configured using the parameter SdConsumedEventGroupID. Note : SubscribeEventgroupAck is under Entry Type 2
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to

	<EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3 [SWS_SD_00204] Page 33 'Message format' (MUST)
Notes	

5.1.5.3 Options Array

SOMEIPSRV_OPTIONS_01: Length field of the IPv4 Endpoint Option

Synopsis	The Length field of the IPv4 Endpoint Option shall be set to 0x0009
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 1 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Length of IPv4 EndPoint Option is set to 0x0009 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Length of IPv4 EndPoint Option is set to 0x0009
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00209] Page 36 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_02: Type field of the IPv4 Endpoint Option

Synopsis	The Type field of the IPv4 Endpoint Option shall be statically set to 0x04.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Type of Option Array is set to SOMEIP_OPTION_IPV4_ENDPOINT 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Type of Option Array is set to SOMEIP_OPTION_IPV4_ENDPOINT
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00210] Page 36 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_03: Reserved field of the IPv4 Endpoint Option

Synopsis	The Reserved field of the IPv4 Endpoint Option (followed by the IPv4-Address field) of the Configuration Option segment shall be statically set to 0x00.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 1 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - First Reserved Field in IPv4 EndPoint Option is set to 0x00 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - First Reserved Field in IPv4 EndPoint Option is set to 0x00
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00211] Page 36 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_04: IPv4-Address field of the IPv4 Endpoint Option

Synopsis	The IPv4-Address field [32 bits] of the IPv4 Endpoint Option shall be set to the local IP address of the relevant Service or Eventgroup.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 1 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - IP Address in IPv4 EndPoint Option is set to <SERVER1-IP-ADDR> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - IP Address in IPv4 EndPoint Option is set to <SERVER1-IP-ADDR>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00212] Page 37 'Options Array' (SHOULD)
Notes	

SOMEIPSRV_OPTIONS_05: Reserved field of the IPv4 Endpoint Option

Synopsis	The Reserved field of the IPv4 Endpoint Option shall statically be set to 0x00.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 1 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Second Reserved Field in IPv4 EndPoint Option is set to 0x00 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Second Reserved Field in IPv4 EndPoint Option is set to 0x00
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00213] Page 37 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_06: Layer 4 Protocol field of the IPv4 Endpoint Option

Synopsis	The Layer 4 Protocol field [8 bits] (L4-Proto) of the IPv4 Endpoint Option shall be set to one of the following values, depending on the port specified: 0x06: TCP 0x11: UDP NOTE: Checking For UDP.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 1 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Transport Protocol in IPv4 EndPoint Option is set to 0x11 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Transport Protocol in IPv4 EndPoint Option is set to 0x11
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00214] Page 37 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_07: Port Number field of the IPv4 Endpoint Option

Synopsis	The Port Number field [16 bits] of the IPv4 Endpoint Option shall carry the UDP or TCP port number for the service instance or Eventgroup.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 1 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Port Number in IPv4 EndPoint Option is set to <SERVICE-ID-1-UDP-PORT> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Port Number in IPv4 EndPoint Option is set to <SERVICE-ID-1-UDP-PORT>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00215] Page 37 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_08: Length field of the IPv4 Multicast Option

Synopsis	The Length field [16 bits] of the IPv4 Multicast Option shall be set to 0x0009.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_MULTICAST 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Length of IPv4 Multicast Option is set to 0x0009 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Length of IPv4 Multicast Option is set to 0x0009
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00390] Page 39 'Options Array' (MUST)

SOMEIPSRV_OPTIONS_09: Type field of the IPv4 Multicast Option

Synopsis	The Type field [8 bits] of the IPv4 Multicast Option shall be statically set to 0x14.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Type of Option Array is set to SOMEIP_OPTION_IPV4_MULTICAST 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Type of Option Array is set to SOMEIP_OPTION_IPV4_MULTICAST
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9

	[SWS_SD_00391] Page 39 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_10: Reserved field after IPv4 Address of the IPv4 Multicast Option

Synopsis	The Reserved field [8 bits] of the IPv4 Multicast Option (followed by the IPv4-Address field) of the Configuration Option segment shall be statically set to 0x00.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_MULTICAST 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - First Reserved Field in IPv4 Multicast Option is set to 0x00 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>

Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - First Reserved Field in IPv4 Multicast Option is set to 0x00
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00392] Page 39 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_11: IPv4-Address field of the IPv4 Multicast Option

Synopsis	The IPv4-Address field [32 bits] of the IPv4 Multicast Option shall be set to the Multicast IP address of the Eventgroup.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_MULTICAST 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains:

	- IP Address in IPv4 Multicast Option is set to <EVENT-GROUP-ID-1-SI-1-MulticastAddr> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - IP Address in IPv4 Multicast Option is set to <EVENT-GROUP-ID-1-SI-1-MulticastAddr>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00393] Page 39 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_12: Reserved field of the IPv4 Multicast Option

Synopsis	The Reserved field [8 bits] of the IPv4 Multicast Option shall statically be set to 0x00.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup

	Ack Entry set to <EVENT-GROUP-ID-1-SI-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_MULTICAST 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Second Reserved Field in IPv4 Multicast Option is set to 0x00 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Second Reserved Field in IPv4 Multicast Option is set to 0x00
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00394] Page 39 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_13: Layer 4 Protocol field of the IPv4 Multicast Option for UDP

Synopsis	The Layer 4 Protocol field [8 bits] (L4-Proto) of the IPv4 Multicast Option shall be set to 0x11 (UDP).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE

	- Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_MULTICAST 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Transport Protocol in IPv4 Multicast Option is set to 0x11 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Transport Protocol in IPv4 Multicast Option is set to 0x11
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00395] Page 39 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_14: Port Number field of the IPv4 Multicast Option

Synopsis	The Port Number field [16 bits] of the IPv4 Multicast Option shall carry the port number for transporting Multicast Events of the Eventgroup.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE

	- Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Entry Array set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_MULTICAST 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Port Number in IPv4 Multicast Option is set to <SERVICE-ID-1-UDP-PORT> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - Port Number in IPv4 Multicast Option is set to <SERVICE-ID-1-UDP-PORT>
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00396] Page 39 'Options Array' (MUST)
Notes	

SOMEIPSRV_OPTIONS_15: Layer 4 Protocol field of the IPv4 Endpoint Option for TCP

Synopsis	The Layer 4 Protocol field [8 bits] (L4-Proto) of the IPv4 Endpoint Option shall be set to one of the following values, depending on the port specified: 0x06: TCP 0x11: UDP NOTE: Checking For TCP.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-2> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-2> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-2> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 1 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Transport Protocol in IPv4 EndPoint Option is set to 0x06 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-2>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Transport Protocol in IPv4 EndPoint Option is set to 0x06
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.3.9 [SWS_SD_00214] Page 37 'Options Array' (MUST)
Notes	

5.1.5.4 Service Discovery Messages

SOMEIPSRV_SD_MESSAGE_01: Instance ID if all service instances shall be returned

Synopsis	Instance ID shall be set to 0xFFFF, if all service instances shall be returned. It shall be set to the Instance ID of a specific service instance, if just a single service instance shall be returned. NOTE: Checking For Instance ID 0xFFFF.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 2 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - SOME/IP Find Service Entry Service Instance ID set to 0xFFFF - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Expected Offer Services' Service ID set to <SERVICE-ID-1> - Number Of Expected Offer Service Entries set to 2 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Number of expected Offer Service Entry is set to 2 6. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 2 to <extractedInstID2> 7. TESTER: Verify that received SOME/IP Notification Message contains: - Service Instance ID of Offer Service Entry 1 is not set to extractedInstID2 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Number of expected Offer Service Entry is set to 2 7. TESTER: Verify that received SOME/IP Notification Message contains: - Service Instance ID of Offer Service Entry 1 is not set to extractedInstID2

Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.1 Find Service Entry [TR_SOMEIP_00351] Page 66 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_02: Instance ID if a specific instance shall be returned

Synopsis	Instance ID shall be set to 0xFFFF, if all service instances shall be returned. It shall be set to the Instance ID of a specific service instance, if just a single service instance shall be returned. NOTE: Checking For Specific Instance ID.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 2 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - SOME/IP Find Service Entry Service Instance ID set to 0xFFFF - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Expected Offer Services' Service ID set to <SERVICE-ID-1> - Number Of Expected Offer Service Entries set to 2 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Number of expected Offer Service Entry is set to 2 6. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 2 to <extractedInstID2> 7. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 8. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - SOME/IP Find Service Entry Service Instance ID set to extractedInstID1 - Service ID in Entry Array set to <SERVICE-ID-1> 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1>

	10. DUT: Sends SOME/IP Notification Message 11. TESTER: Verify that received SOME/IP Notification Message contains: - Number of expected Offer Service Entry is set to 1 12. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - SOME/IP Find Service Entry Service Instance ID set to extractedInstID2 - Service ID in Entry Array set to <SERVICE-ID-1> 13. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 14. DUT: Sends SOME/IP Notification Message 15. TESTER: Verify that received SOME/IP Notification Message contains: - Number of expected Offer Service Entry is set to 1 16. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Number of expected Offer Service Entry is set to 2 10. DUT: Sends SOME/IP Notification Message 11. TESTER: Verify that received SOME/IP Notification Message contains: - Number of expected Offer Service Entry is set to 1 14. DUT: Sends SOME/IP Notification Message 15. TESTER: Verify that received SOME/IP Notification Message contains: - Number of expected Offer Service Entry is set to 1
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.1 Find Service Entry [TR_SOMEIP_00351] Page 66 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_03: Major Version when any version shall be returned

Synopsis	Major Version shall be set to 0xFF, that means that services with any version shall be returned. If set to value different than 0xFF, services with this specific major version shall be returned only. NOTE: Checking For Major Version 0xFF.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>` `        - Number Of Instances : 1` `2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through` `        <DIface-0> containing :` `        - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE` `        - SOME/IP Find Service Entry Service Major Version set to 0xFF` `        - SOME/IP Find Service Entry TTL set to 0xFFFFFF` `        - Service ID in Entry Array set to <SERVICE-ID-1>` `3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `        - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE` `        - Service ID in Entry Array set to <SERVICE-ID-1>` `4. DUT: Sends SOME/IP Notification Message` `5. TESTER: Verify that received SOME/IP Notification Message contains:` `        - Major Version in Entry Array is set to` `          <SERVICE-ID-1-MAJ-VER>` `6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>`
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Major Version in Entry Array is set to <SERVICE-ID-1-MAJ-VER>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.1 Find Service Entry [TR_SOMEIP_00351] Page 66 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_04: Major Version when a specific version shall be returned

Synopsis	Major Version shall be set to 0xFF, that means that services with any version shall be returned. If set to value different than 0xFF, services with this specific major version shall be returned only. NOTE: Checking For Specific Major Version.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - SOME/IP Find Service Entry Service Major Version set to <SERVICE-ID-1-MAJ-VER> - SOME/IP Find Service Entry TTL set to 0xFFFFFF - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Major Version in Entry Array is set to <SERVICE-ID-1-MAJ-VER> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Major Version in Entry Array is set to <SERVICE-ID-1-MAJ-VER>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.1 Find Service Entry [TR_SOMEIP_00351] Page 66 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_05: Minor Version when any version shall be returned

Synopsis	Minor Version shall be set to 0xFFFF FFFF, that means that services with any version shall be returned. If set to a value different to 0xFFFF FFFF, services with this specific minor version shall be returned only. NOTE: Checking For Minor Version 0xFFFFFFFF.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - SOME/IP Find Service Entry Service Minor Version set to 0xFFFFFFFF - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Minor Version in Entry Array is set to <SERVICE-ID-1-MINOR-VER> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Minor Version in Entry Array is set to <SERVICE-ID-1-MINOR-VER>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.1 Find Service Entry [TR_SOMEIP_00351] Page 66 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_06: Minor Version when a specific version shall be returned

Synopsis	Minor Version shall be set to 0xFFFF FFFF, that means that services with any version shall be returned. If set to a value different to 0xFFFF FFFF, services with this specific minor version shall be returned only. NOTE: Checking For Specific Minor Version.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - SOME/IP Find Service Entry Service Minor Version set to <SERVICE-ID-1-MINOR-VER> - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Minor Version in Entry Array is set to <SERVICE-ID-1-MINOR-VER> 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Minor Version in Entry Array is set to <SERVICE-ID-1-MINOR-VER>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.1 Find Service Entry [TR_SOMEIP_00351] Page 66 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_07: TTL and the lifetime of a service instance

Synopsis	TTL shall be set to the lifetime of the service instance. After this lifetime the service instance shall considered not been offered.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to <SERVICE-ID-1-TTL> 6. TESTER: Wait till <SERVICE-ID-1-TTL> To Timeout offer service from DUT 7. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 8. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 9. DUT: Does not send SOME/IP Response Message 10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to <SERVICE-ID-1-TTL>

	9. DUT: Does not send SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.2 Offer Service Entry [TR_SOMEIP_00356] Page 67 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_08: Offer Service entries for IPv4

Synopsis	Offer Service entries shall always reference at least an IPv4 or IPv6 Endpoint Option to signal how the service is reachable. NOTE: Test is for IPv4.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.2 Offer Service Entry [TR_SOMEIP_00357] Page 67 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_09: Endpoint Options Port Numer

Synopsis	The IP addresses and port numbers of the Endpoint Options shall also be used for transporting events and notification events, In the case of UDP this information is used for the source address and the source port of the events and notification events. NOTE: This test is for UDP port.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1 6. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option Transport Port to <extractedport> 7. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 8. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1>

	10. DUT: Sends SOME/IP Notification Message 11. DUT CONFIGURE: Trigger Event on <DIface-0> with the following informations - Event ID : <EVENT-ID-1-EG-ID-1> - Event Group ID : <EVENT-GROUP-ID-1-SI-1> - Service ID : <SERVICE-ID-1> 12. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Message Expected UDP Src Port set to extractedport - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-1> 13. DUT: Sends SOME/IP Notification Message 14. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1 10. DUT: Sends SOME/IP Notification Message 13. DUT: Sends SOME/IP Notification Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.2 Offer Service Entry [TR_SOMEIP_00360],[TR_SOMEIP_00361] Page (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_10: Stop offer service and TTL field

Synopsis	The stop offer service entry type shall be used to stop offering service instances. Stop Offer Service entries shall set the entry fields exactly like the Offer Service entry they are stopping, except: TTL shall be set to 0x000000. NOTE: Check For TTL.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_STOP_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to 0
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message 8. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to 0
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.1.3 Stop Offer Service Entry [TR_SOMEIP_00363] page 68 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_11: Subscribe Eventgroup entry type

Synopsis	The Subscribe Eventgroup entry type shall be used to subscribe to an eventgroup.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1 6. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option Transport Port to <extractedport> 7. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 8. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1>

	10. DUT: Sends SOME/IP Notification Message 11. DUT CONFIGURE: Trigger Event on <DIface-0> with the following informations - Event ID : <EVENT-ID-1-EG-ID-1> - Event Group ID : <EVENT-GROUP-ID-1-SI-1> - Service ID : <SERVICE-ID-1> 12. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Message Expected UDP Dest Port set to extractedport - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-1> 13. DUT: Sends SOME/IP Notification Message 14. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1 10. DUT: Sends SOME/IP Notification Message 13. DUT: Sends SOME/IP Notification Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.1 Subscribe Eventgroup Entry [TR_SOMEIP_00385] Page 68 (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_12: Stop Subscribe Eventgroup entry

Synopsis	The Stop Subscribe Eventgroup entry type shall be used to stop subscribing to eventgroups. Stop Subscribe Eventgroup entries shall set the entry fields exactly like the Subscribe Eventgroup entry they are stopping, except: TTL shall be set to 0x000000.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-

```
0>
        - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE
        - Service ID in Entry Array set to <SERVICE-ID-1>
4. DUT: Sends SOME/IP Notification Message
5. TESTER: Verify that received SOME/IP Notification Message
contains:
        - SOMEIP Expected Number Of IPv4 Endpoint Option is greater
           than or equal to 1
6. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option
        Transport Port to <extractedport>
7. TESTER: Extracts the content of Service Instance ID of Offer
Service
        Entry 1 to <extractedInstID1>
8. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT
through
        <DIface-0> containing :
        - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE
        - Service ID in Subscribe Eventgroup Entry set to
           <SERVICE-ID-1>
        - EventGroup ID in Subscribe Eventgroup Entry set to
           <EVENT-GROUP-ID-1-SI-1>
        - Service Instance ID in Subscribe Eventgroup Entry set to
           extractedInstID1
9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-
0>
        - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK
        - SOME/IP Expected Service ID in Subscribe EventGroup Ack
           Entry set to <SERVICE-ID-1>
        - SOME/IP Expected Service Instance ID in Subscribe
           EventGroup Ack Entry set to extractedInstID1
        - SOME/IP Expected EventGroup ID in Subscribe EventGroup
Ack
           Entry set to <EVENT-GROUP-ID-1-SI-1>
10. DUT: Sends SOME/IP Notification Message
11. DUT CONFIGURE: Trigger Event on <DIface-0> with the following
informations
        - Event ID : <EVENT-ID-1-EG-ID-1>
        - Event Group ID : <EVENT-GROUP-ID-1-SI-1>
        - Service ID : <SERVICE-ID-1>
12. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-
0>
        - SOME/IP Message Expected UDP Dest Port set to
extractedport
        - SOME/IP Expected Service ID set to <SERVICE-ID-1>
        - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK
        - SOME/IP Expected Event ID in SOME/IP Header set to
           <EVENT-ID-1-EG-ID-1>
13. DUT: Sends SOME/IP Notification Message
14. TESTER:  <CLIENT-1> Sends SOME/IP Notification Message to DUT
         through <DIface-0> containing :
        - Entry Type set to SOMEIP_ENTRY_STOP_SUBSCRIBE
        - SOME/IP Subscribe Eventgroup Entry TTL set to 0
        - Service ID in Subscribe Eventgroup Entry set to
           <SERVICE-ID-1>
        - EventGroup ID in Subscribe Eventgroup Entry set to
           <EVENT-GROUP-ID-1-SI-1>
```

	- Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 - Major Version in Subscribe EventGroup Entry set to <SERVICE-ID-1-MAJ-VER> 15. DUT CONFIGURE: Trigger Event on <DIface-0> with the following informations - Event ID : <EVENT-ID-1-EG-ID-1> - Event Group ID : <EVENT-GROUP-ID-1-SI-1> - Service ID : <SERVICE-ID-1> 16. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Message Expected UDP Dest Port set to extractedport - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-1> 17. DUT: Does not send SOME/IP Notification Message 18. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1 10. DUT: Sends SOME/IP Notification Message 13. DUT: Sends SOME/IP Notification Message 17. DUT: Does not send SOME/IP Notification Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.2 Stop Subscribe Eventgroup Entry (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_13: Subscribe Eventgroup Acknowledgment entry type

Synopsis	The Subscribe Eventgroup Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was accepted. Subscribe Eventgroup Acknowledgment entries shall set the entry fields in the following way: Type shall be set to 0x07 (SubscribeEventgroupAck). Service ID, Instance ID, Major Version, Eventgroup ID, TTL, and Reserved shall be the same value as in the Subcribe that is being answered.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - SOME/IP Subscribe Eventgroup Entry TTL set to 0xFFFFFF - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 - Major Version in Subscribe EventGroup Entry set to <SERVICE-ID-1-MAJ-VER> - Reserved Field in Subscribe EventGroup Ack Entry set to 0 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry is set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe

	`` EventGroup Ack Entry is set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry is set to <EVENT-GROUP-ID-1-SI-1> - TTL in Subscribe EventGroup Ack Entry is set to 0xFFFFFF - Reserved Field in Subscribe EventGroup Ack Entry is set to 0 - Major Version in Subscribe EventGroup Ack Entry is set to <SERVICE-ID-1-MAJ-VER> 10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> ```
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry is set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry is set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry is set to <EVENT-GROUP-ID-1-SI-1> - TTL in Subscribe EventGroup Ack Entry is set to 0xFFFFFF - Reserved Field in Subscribe EventGroup Ack Entry is set to 0 - Major Version in Subscribe EventGroup Ack Entry is set to <SERVICE-ID-1-MAJ-VER>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.3 Subscribe Eventgroup Acknowledgement (Subscribe Eventgroup (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_14: Subscribe Eventgroup Negative Acknowledgment entry type

Synopsis	The Subscribe Eventgroup Negative Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was NOT accepted. Subscribe Eventgroup Negative Acknowledgment entries shall set the entry fields in the following way: Type shall be set to 0x07 (SubscribeEventgroupAck). Service ID, Instance ID, Major Version, Eventgroup ID and Reserved shall be the same value as in the subscribe that is being answered. The TTL shall be set to 0x000000. NOTE: Check For TTL Field and Type.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	``` 1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> ```

	- Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - SOME/IP Subscribe Eventgroup Entry TTL set to 0xFFFFFF - Service ID in Subscribe Eventgroup Entry set to <UNKNOWN-SERVICE-ID> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 - Major Version in Subscribe EventGroup Entry set to <SERVICE-ID-1-MAJ-VER> - Reserved Field in Subscribe EventGroup Ack Entry set to 0 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <UNKNOWN-SERVICE-ID> 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Subscribe EventGroup Ack Entry is set to 0 10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Subscribe EventGroup Ack Entry is set to 0
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.4 Subscribe Eventgroup Negative Acknowledgement (Subscribe (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_15: Subscribe Eventgroup Negative Acknowledgment entry type

Synopsis	The Subscribe Eventgroup Negative Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was NOT accepted. Subscribe Eventgroup Negative Acknowledgment entries shall set the entry fields in the following way: Type shall be set to 0x07 (SubscribeEventgroupAck). Service ID, Instance ID, Major Version, Eventgroup ID and Reserved shall be the same value as in the subscribe that is being answered. The TTL shall be set to 0x000000. NOTE: Checking for later Condition that Service ID, Instance ID, Major Version, Eventgroup ID and Reserved shall be the same value as in the subscribe .
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `- Service ID : <SERVICE-ID-1>` `- Number Of Instances : 1` `2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through` `<DIface-0> containing :` `- Entry Type set to SOMEIP_ENTRY_FIND_SERVICE` `- Service ID in Entry Array set to <SERVICE-ID-1>` `3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `- Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE` `- Service ID in Entry Array set to <SERVICE-ID-1>` `4. DUT: Sends SOME/IP Notification Message` `5. TESTER: Extracts the content of Service Instance ID of Offer Service` `Entry 1 to <extractedInstID1>` `6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through` `<DIface-0> containing :` `- Entry Type set to SOMEIP_ENTRY_SUBSCRIBE` `- SOME/IP Subscribe Eventgroup Entry TTL set to 0xFFFFFF` `- Service ID in Subscribe Eventgroup Entry set to <UNKNOWN-SERVICE-ID>` `- EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1>` `- Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1` `- Major Version in Subscribe EventGroup Entry set to <SERVICE-ID-1-MAJ-VER>` `- Reserved Field in Subscribe EventGroup Ack Entry set to 0` `7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `- Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK` `- SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <UNKNOWN-SERVICE-ID>`

	`8. DUT: Sends SOME/IP Notification Message` `9. TESTER: Verify that received SOME/IP Notification Message contains:` `- SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry is set to extractedInstID1` `- SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry is set to <EVENT-GROUP-ID-1-SI-1>` `- TTL in Subscribe EventGroup Ack Entry is set to 0` `- Reserved Field in Subscribe EventGroup Ack Entry is set to 0` `- Major Version in Subscribe EventGroup Ack Entry is set to <SERVICE-ID-1-MAJ-VER>` `10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations` `- Service ID : <SERVICE-ID-1>`
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry is set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry is set to <EVENT-GROUP-ID-1-SI-1> - TTL in Subscribe EventGroup Ack Entry is set to 0 - Reserved Field in Subscribe EventGroup Ack Entry is set to 0 - Major Version in Subscribe EventGroup Ack Entry is set to <SERVICE-ID-1-MAJ-VER>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.4 Subscribe Eventgroup Negative Acknowledgement (Subscribe (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_16: Subscribe Eventgroup Negative Acknowledgment entry type

Synopsis	The Subscribe Eventgroup Negative Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was NOT accepted. Subscribe Eventgroup Negative Acknowledgment entries shall set the entry fields in the following way: Type shall be set to 0x07 (SubscribeEventgroupAck). Service ID, Instance ID, Major Version, Eventgroup ID and Reserved shall be the same value as in the subscribe that is being answered. The TTL shall be set to 0x000000. Reasons for sending a Subscribe Eventgroup Negative Acknowledgment include: Combination of Service ID, Instance ID, Eventgroup ID, and Major Version is unknown. NOTE: Test For Unknown Service ID.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - SOME/IP Subscribe Eventgroup Entry TTL set to 0xFFFFFF - Service ID in Subscribe Eventgroup Entry set to <UNKNOWN-SERVICE-ID> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 - Major Version in Subscribe EventGroup Entry set to <SERVICE-ID-1-MAJ-VER> - Reserved Field in Subscribe EventGroup Ack Entry set to 0 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK 8. DUT: Sends SOME/IP Notification Message

	9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry is set to <UNKNOWN-SERVICE-ID> - TTL in Subscribe EventGroup Ack Entry is set to 0 10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry is set to <UNKNOWN-SERVICE-ID> - TTL in Subscribe EventGroup Ack Entry is set to 0
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.4 Subscribe Eventgroup Negative Acknowledgement (Subscribe (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_17: Subscribe Eventgroup Negative Acknowledgment entry type

Synopsis	The Subscribe Eventgroup Negative Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was NOT accepted. Subscribe Eventgroup Negative Acknowledgment entries shall set the entry fields in the following way: Type shall be set to 0x07 (SubscribeEventgroupAck). Service ID, Instance ID, Major Version, Eventgroup ID and Reserved shall be the same value as in the subscribe that is being answered. The TTL shall be set to 0x000000. Reasons for sending a Subscribe Eventgroup Negative Acknowledgment include: Combination of Service ID, Instance ID, Eventgroup ID, and Major Version is unknown. NOTE: Test For Unknown Instance ID.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1>

	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - SOME/IP Subscribe Eventgroup Entry TTL set to 0xFFFFFF - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1+1 - Major Version in Subscribe EventGroup Entry set to <SERVICE-ID-1-MAJ-VER> - Reserved Field in Subscribe EventGroup Ack Entry set to 0 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry is set to extractedInstID1+1 - TTL in Subscribe EventGroup Ack Entry is set to 0 10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry is set to extractedInstID1+1 - TTL in Subscribe EventGroup Ack Entry is set to 0
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.4 Subscribe Eventgroup Negative Acknowledgement (Subscribe (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_18: Subscribe Eventgroup Negative Acknowledgment entry type

Synopsis	The Subscribe Eventgroup Negative Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was NOT accepted. Subscribe Eventgroup Negative Acknowledgment entries shall set the entry fields in the following way: Type shall be set to 0x07 (SubscribeEventgroupAck). Service ID, Instance ID, Major Version, Eventgroup ID and Reserved shall be the same value as in the subscribe that is being answered. The TTL shall be set to 0x000000. Reasons for sending a Subscribe Eventgroup Negative Acknowledgment include: Combination of Service ID, Instance ID, Eventgroup ID, and Major Version is unknown. NOTE: Test For Unknown Major Version.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - SOME/IP Subscribe Eventgroup Entry TTL set to 0xFFFFFF - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 - Major Version in Subscribe EventGroup Entry set to <SERVICE-ID-1-MAJ-VER+1> - Reserved Field in Subscribe EventGroup Ack Entry set to 0 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack

	Entry set to <SERVICE-ID-1> 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Subscribe EventGroup Ack Entry is set to 0 - Major Version in Subscribe EventGroup Ack Entry is set to <SERVICE-ID-1-MAJ-VER+1> 10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Subscribe EventGroup Ack Entry is set to 0 - Major Version in Subscribe EventGroup Ack Entry is set to <SERVICE-ID-1-MAJ-VER+1>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.4 Subscribe Eventgroup Negative Acknowledgement (Subscribe (MUST)
Notes	

SOMEIPSRV_SD_MESSAGE_19: Subscribe Eventgroup Negative Acknowledgment entry type

Synopsis	The Subscribe Eventgroup Negative Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was NOT accepted. Subscribe Eventgroup Negative Acknowledgment entries shall set the entry fields in the following way: Type shall be set to 0x07 (SubscribeEventgroupAck). Service ID, Instance ID, Major Version, Eventgroup ID and Reserved shall be the same value as in the subscribe that is being answered. The TTL shall be set to 0x000000. Reasons for sending a Subscribe Eventgroup Negative Acknowledgment include: Combination of Service ID, Instance ID, Eventgroup ID, and Major Version is unknown. NOTE: Test For Unknown Event Group ID.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>

	- Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - SOME/IP Subscribe Eventgroup Entry TTL set to 0xFFFFFF - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <UNKNOWN-EVENT-GROUP-ID> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 - Major Version in Subscribe EventGroup Entry set to <SERVICE-ID-1-MAJ-VER> - Reserved Field in Subscribe EventGroup Ack Entry set to 0 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry is set to <UNKNOWN-EVENT-GROUP-ID> - TTL in Subscribe EventGroup Ack Entry is set to 0 10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 9. TESTER: Verify that received SOME/IP Notification Message contains: - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry is set to <UNKNOWN-EVENT-GROUP-ID> - TTL in Subscribe EventGroup Ack Entry is set to 0
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.4.2.4 Subscribe Eventgroup Negative Acknowledgement (Subscribe (MUST)
Notes	

5.1.5.5 Service Discovery Communication Behavior

SOMEIPSRV_SD_BEHAVIOR_01: After messages in the Repetition Phase the delay is doubled

Synopsis	After sending the first message the Repetition Phase of this Service Instance/these Service Instances is entered. After each message sent in the Repetition Phase the delay is doubled.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: <CLIENT-1> Listens (upto <SERVICE-ID-1-REP-BASE-INTV*3+ParamToleranceTime> second) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 5. DUT: Sends 2 SOME/IP Notification Messages 6. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is greater than (<SERVICE-ID-1-REP-BASE-INTV*2> - <ParamToleranceTimeMillisec>) micro second 7. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is less than (<SERVICE-ID-1-REP-BASE-INTV*2> + <ParamToleranceTimeMillisec>) micro second 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 5. DUT: Sends 2 SOME/IP Notification Messages 6. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is greater than

	(<SERVICE-ID-1-REP-BASE-INTV*2> - <ParamToleranceTimeMillisec>) micro second 7. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is less than (<SERVICE-ID-1-REP-BASE-INTV*2> + <ParamToleranceTimeMillisec>) micro second
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.5.1 Startup Behavior [TR_SOMEIP_00404], [TR_SOMEIP_00406] Page 71 (SHOULD)
Notes	

SOMEIPSRV_SD_BEHAVIOR_02: Main Phase Offer Messages and Publish Messages cyclically

Synopsis	In the Main Phase Offer Messages and Publish Messages shall be sent cyclically if a CYCLIC_OFFER_DELAY is configured.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Wait till <SERVICE-ID-1-TOTAL-REP-INTV> For DUT to enter Main Phase 5. TESTER: <CLIENT-1> Listens (upto <SERVICE-ID-1-CYCLE-INTV*2+ParamToleranceTime> second) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 6. DUT: Sends 2 SOME/IP Notification Messages 7. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is greater than (<SERVICE-ID-1-CYCLE-INTV> - <ParamToleranceTime>) second 8. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is less than (<SERVICE-ID-1-CYCLE-INTV> + <ParamToleranceTime>) second 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 6. DUT: Sends 2 SOME/IP Notification Messages 7. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is greater than

	(<SERVICE-ID-1-CYCLE-INTV> - <ParamToleranceTime>) second 8. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is less than (<SERVICE-ID-1-CYCLE-INTV> + <ParamToleranceTime>) second
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.5.1 Startup Behavior [TR_SOMEIP_00412] Page 71 (SHOULD)
Notes	

SOMEIPSRV_SD_BEHAVIOR_03: Response if the last offer was more than half of cyclic offer delay

Synopsis	Find messages received with the Unicast Flag set to 1, shall be answered with a multicast RESPONSE if the last offer was sent 1/2 CYCLIC_OFFER_DELAY or longer ago.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Wait till <SERVICE-ID-1-TOTAL-REP-INTV> For DUT to enter Main Phase 5. TESTER: <CLIENT-1> Listens (upto <SERVICE-ID-1-CYCLE-INTV> second) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 6. DUT: Sends SOME/IP Notification Message 7. TESTER: Wait till (<SERVICE-ID-1-CYCLE-INTV>/2) For DUT to consume 1/2 Cyclic_Offer_Delay 8. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>

	- Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Message Expected IP Dst Address set to <SOME_IP_MULTICAST_IP_ADDR> 10. DUT: Sends SOME/IP Notification Message 11. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 6. DUT: Sends SOME/IP Notification Message 10. DUT: Sends SOME/IP Notification Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.5.2 Server Answer Behavior [TR_SOMEIP_00423] Page 73 (SHOULD)
Notes	

SOMEIPSRV_SD_BEHAVIOR_04: Response for Find messages with Unicast Flag set to 0

Synopsis	Find messages received with Unicast Flag set to 0 (multicast), shall be answered with a multicast response.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. DUT: Sends SOME/IP Notification Message 4. TESTER: Wait till <SERVICE-ID-1-TOTAL-REP-INTV> For DUT to enter Main Phase 5. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Unicast Flag in Notification Message set to 0 - Service ID in Entry Array set to <SERVICE-ID-1> 6. TESTER: <CLIENT-1> Listens (upto <SERVICE-ID-1-CYCLE-INTV> second) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - SOME/IP Message Expected IP Dst Address set to <SOME_IP_MULTICAST_IP_ADDR> 7. DUT: Sends SOME/IP Notification Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	3. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Notification Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.7.5.2 Server Answer Behavior [TR_SOMEIP_00423] Page 73 (SHOULD)
Notes	

SOMEIPSRV_SD_BEHAVIOR_05: Stop offering Service Instances

Synopsis	The Stop Offer Service entry type shall be used to stop offering Service Instances.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 8. DUT: Sends SOME/IP Notification Message 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> 10. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>

	- Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 11. DUT: Sends SOME/IP Notification Message 12. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to 0 13. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 14. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 15. DUT: Does not send SOME/IP Notification Message 16. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 11. DUT: Sends SOME/IP Notification Message 12. TESTER: Verify that received SOME/IP Notification Message contains: - TTL in Entry Array is set to 0 15. DUT: Does not send SOME/IP Notification Message
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.4.5 [SWS_SD_00422] Page 48 'StopOfferService entry' (SHOULD)
Notes	

SOMEIPSRV_SD_BEHAVIOR_06: Stop subscribing to an Eventgroup

Synopsis	To stop subscribing to an Eventgroup, the StopSubscribeEventgroup entry shall be used.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 8. DUT: Sends SOME/IP Notification Message 9. DUT CONFIGURE: Trigger Event on <DIface-0> with the following informations - Event ID : <EVENT-ID-1-EG-ID-1> - Event Group ID : <EVENT-GROUP-ID-1-SI-1>

	<pre> - Service ID : <SERVICE-ID-1> 10. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-1> 11. DUT: Sends SOME/IP Notification Message 12. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_STOP_SUBSCRIBE - SOME/IP Subscribe Eventgroup Entry TTL set to 0x000000 - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1-INSTANCE-ID> 13. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-1> 14. DUT: Does not send SOME/IP Notification Message 15. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1></pre>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 11. DUT: Sends SOME/IP Notification Message 14. DUT: Does not send SOME/IP Notification Message
Reference	Specification of Service Discovery V1.2.0 R4.1 Rev 3 Section 7.4.6.2 Page 50 'StopSubscribeEventgroup entry' (SHOULD)
Notes	

5.1.5.6 SOME/IP Basic Functionality

SOMEIPSRV_BASIC_01: Define service using the Service ID

Synopsis	A service shall be identified using the Service-ID.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.1 Definition of Identifiers [TR_SOMEIP_00001] Page 14 (MUST)
Notes	

SOMEIPSRV_BASIC_02: Reserved values for Service Instance IDs

Synopsis	The Service-Instance-IDs of 0x0000 and 0xFFFF shall not be used for a service, since 0x0000 is reserved and 0xFFFF is used to describe all service instances.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - SOME/IP Find Service Entry Service Instance ID set to 0xFFFF - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Service Instance ID of Offer Service Entry 1 is not set to 0 - Service Instance ID of Offer Service Entry 1 is not set to 0xFFFF 6. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - Service Instance ID of Offer Service Entry 1 is not set to 0 - Service Instance ID of Offer Service Entry 1 is not set to 0xFFFF
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.1 Definition of Identifiers [TR_SOMEIP_00008] Page 14 (MUST)
Notes	

SOMEIPSRV_BASIC_03: Method ID of a notification has highest bit set to 1

Synopsis	Methods shall use Method-IDs with the highest bit set to 0, while the Method-IDs highest bit shall be set to 1 for events and notifications of fields. NOTE: Checking for notification.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1 6. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option Transport Port to <extractedport> 7. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 8. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1

	- SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 10. DUT: Sends SOME/IP Notification Message 11. DUT CONFIGURE: Trigger Event on <DIface-0> with the following informations - Event ID : <EVENT-ID-1-EG-ID-1> - Event Group ID : <EVENT-GROUP-ID-1-SI-1> - Service ID : <SERVICE-ID-1> 12. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Message Expected UDP Dest Port set to extractedport - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-1> 13. DUT: Sends SOME/IP Notification Message 14. TESTER: Verify that received SOME/IP Notification Message contains: - 0-Bit field of Method/EventID in SOME/IP Packet Header is set to 1 15. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 5. TESTER: Verify that received SOME/IP Notification Message contains: - SOMEIP Expected Number Of IPv4 Endpoint Option is greater than or equal to 1 10. DUT: Sends SOME/IP Notification Message 13. DUT: Sends SOME/IP Notification Message 14. TESTER: Verify that received SOME/IP Notification Message contains: - 0-Bit field of Method/EventID in SOME/IP Packet Header is set to 1
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.1 Definition of Identifiers [TR_SOMEIP_00011] Page 14 (MUST)
Notes	

5.1.5.7 Specification of the SOME/IP on-wire format

SOMEIPSRV_ONWIRE_01: IP addresses and port number of the Reponse message

Synopsis	For the response and error message the IP addresses and port number of the transport protocol shall match the request message. NOTE: Checking for Response message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Message Expected IP Src Address set to <SERVER1-IP-ADDR> - SOME/IP Message Expected IP Dst Address set to <CLIENT1-IP-ADDR> - SOME/IP Message Expected UDP Src Port set to <SERVICE-ID-1-UDP-PORT> - SOME/IP Message Expected UDP Dest Port set to <CLIENT1-UDP-PORT> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message

	7. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.1.1 Mapping of IP Addresses and Ports in Response and Error (MUST)
Notes	

SOMEIPSRV_ONWIRE_02: MSB of Method ID in Response Message.

Synopsis	With 16 Bit Service-ID and a 16 Bit Method-ID starting with a 0-Bit, this allows for up to 65536 services with up to 32768 methods each. NOTE: Checking for 0 bit in MSB of Method ID in Response Message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - 0-Bit field of Method/EventID in SOME/IP Packet Header is set to 0 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - 0-Bit field of Method/EventID in SOME/IP Packet Header is set to 0
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.2.1 Structure of the Message ID [TR_SOMEIP_00038] Page 18 (MUST)
Notes	

SOMEIPSRV_ONWIRE_03: Copy Request ID from the request to the response message.

Synopsis	When generating a response message, the server has to copy the Request ID from the request to the response message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - SOME/IP Packet Expected Header Request ID is set to <CLIENT1-CURR-REQUEST-ID> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - SOME/IP Packet Expected Header Request ID is set to <CLIENT1-CURR-REQUEST-ID>

Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.4 Request ID [32 Bit] [TR_SOMEIP_00043] Page 19 (MUST)
Notes	

SOMEIPSRV_ONWIRE_04: Request IDs may be reused if response arrived

Synopsis	Request IDs might be reused as soon as the response arrived or is not expected to arrive anymore (timeout). NOTE: Testing first condition where response arrived.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-

	0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 10. DUT: Sends SOME/IP Response Message 11. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message 10. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.4 Request ID [32 Bit] [TR_SOMEIP_00044] Page 19 (MUST)
Notes	

SOMEIPSRV_ONWIRE_05: Protocol Version

Synopsis	Protocol Version is an 8 Bit field containing the SOME/IP protocol version, which currently shall be set to 0x01.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Protocol Version set to 0x01 - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Protocol Version set to 0xFF - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-

	0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_WRONG_PROTOCOL_VERSION 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Protocol Version set to 0x01 - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Protocol Version set to 0x01 - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 10. DUT: Sends SOME/IP Response Message 11. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 10. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.5 Protocol Version [8 Bit] [TR_SOMEIP_00052] Page 20 (MUST)
Notes	

SOMEIPSRV_ONWIRE_06: Interface Version

Synopsis	Interface Version is an 8 Bit field that contains the Major Version of the Service Interface.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - Interface Version is set to <SERVICE-ID-1-INTF-VER-MAJ> 9. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <UNKNOWN-INTF-VER-MAJ-SI-1> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to

	<METHOD-ID-1-SI-1> 10. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Interface Version set to <UNKNOWN-INTF-VER-MAJ-SI-1> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_WRONG_INTERFACE_VERSION 11. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 12. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - Interface Version is set to <SERVICE-ID-1-INTF-VER-MAJ> 11. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.6 Interface Version [8 Bit] [TR_SOMEIP_00053] Page 20 (MUST)
Notes	

SOMEIPSRV_ONWIRE_07: Message Type and Response after a Request

Synopsis	The Message Type field is used to differentiate different types of messages and shall contain the following values. 0x00 REQUEST A request expecting a response (even void). 0x80 RESPONSE The response message. NOTE: Testing Response message after sending a request.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through

	<DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.7 Message Type [8 Bit] [TR_SOMEIP_00055] Page 20 (MUST)
Notes	

SOMEIPSRV_ONWIRE_08: Message Type and Request no Return

Synopsis	The Message Type field is used to differentiate different types of messages and shall contain the following values. 0x01 REQUEST_NO_RETURN A fire&forget request.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOMEIP_MSG_TYPE_REQUEST_NO_RETURN Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Does not send SOME/IP Response Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Does not send SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.7 Message Type [8 Bit] [TR_SOMEIP_00055] Page 20 (MUST)

SOMEIPSRV_ONWIRE_09: Message Type and Request ACK

Synopsis	The Message Type field is used to differentiate different types of messages and shall contain the following values. 0x40 REQUEST_ACK Acknowledgment for REQUEST (optional).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOMEIP_MSG_TYPE_REQUEST_ACK Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOMEIP_MSG_TYPE_REQUEST_ACK Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.7 Message Type [8 Bit] [TR_SOMEIP_00055] Page 20 (MAY)

SOMEIPSRV_ONWIRE_10: Message Type and Unkown Service ID

Synopsis	The Message Type field is used to differentiate different types of messages and shall contain the following values. 0x81 ERROR The response containing an error. NOTE: Checking for error UNKNOWN Service ID.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	`1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations` `        - Service ID : <SERVICE-ID-1>` `        - Number Of Instances : 1` `2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through` `        <DIface-0> containing :` `        - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE` `        - Service ID in Entry Array set to <SERVICE-ID-1>` `3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `        - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE` `        - Service ID in Entry Array set to <SERVICE-ID-1>` `4. DUT: Sends SOME/IP Notification Message` `5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through` `        <DIface-0> containing :` `        - SOME/IP Packet Send Header Request ID set to` `           <CLIENT1-CURR-REQUEST-ID>` `        - SOME/IP Send Request Message Service ID set to` `           <SERVICE-ID-1>` `        - SOME/IP Send Request Message Method ID set to` `           <METHOD-ID-1-SI-1>` `6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>` `        - SOME/IP Packet Expected Header Request ID set to` `           <CLIENT1-CURR-REQUEST-ID>` `        - SOME/IP Expected Service ID set to <SERVICE-ID-1>` `        - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1>` `        - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK` `7. DUT: Sends SOME/IP Response Message` `8. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through` `        <DIface-0> containing :` `        - SOME/IP Packet Send Header Request ID set to` `           <CLIENT1-CURR-REQUEST-ID>` `        - SOME/IP Packet Send Header Interface Version set to` `           <SERVICE-ID-1-INTF-VER-MAJ>` `        - SOME/IP Send Request Message Service ID set to` `           <UNKNOWN-SERVICE-ID>` `        - SOME/IP Send Request Message Method ID set to` `           <METHOD-ID-1-SI-1>` `9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-`

	0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <UNKNOWN-SERVICE-ID> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_UNKNOWN_SERVICE 10. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 11. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message 10. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.7 Message Type [8 Bit] [TR_SOMEIP_00055] Page 20 (MUST)
Notes	

SOMEIPSRV_ONWIRE_11: Return code for normal request response

Synopsis	The Return Code is used to signal whether a request was successfully been processed. NOTE: Checking for Normal Request Response Communication i.e. Error Code E_OK.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to

	<METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - Return Code is set to SOMEIP_RET_CODE_E_OK 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - Return Code is set to SOMEIP_RET_CODE_E_OK
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.8 Return Code [8 Bit] [TR_SOMEIP_00058, TR_SOMEIP_00191] Page 21 (MUST)
Notes	

SOMEIPSRV_ONWIRE_12: Return code for an Unknown Method ID error

Synopsis	The Return Code is used to signal whether a request was successfully been processed. NOTE: Checking for Error in case of Unknown Method ID in Request Response Communication i.e. Error Code E_UNKNOWN_METHOD.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID>

	- SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <UNKNOWN-METHOD-ID-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <UNKNOWN-METHOD-ID-SI-1> 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - Return Code is set to SOMEIP_RET_CODE_E_UNKNOWN_METHOD 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - Return Code is set to SOMEIP_RET_CODE_E_UNKNOWN_METHOD
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.2.3.8 Return Code [8 Bit] [TR_SOMEIP_00058, TR_SOMEIP_00191] Page 21 (MUST)
Notes	

5.1.5.8 RPC Protocol specification

SOMEIPSRV_RPC_01: Use a single TCP connection for all Methods

Synopsis	The client and server shall use a single TCP connection for all methods, events, and notifications of a service instance. NOTE: This test is for methods.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-2> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-2> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-2> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-2> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-2> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-2> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-2> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-2>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.1.2 TCP Binding [TR_SOMEIP_00150] Page 33 (MUST)

Notes	

SOMEIPSRV_RPC_02: Use a single TCP connection for all Notifications

Synopsis	The client and server shall use a single TCP connection for all methods, events, and notifications of a service instance. NOTE: This test is for Notifications.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-2> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-2> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-2> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. 7. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-2> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-2> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 8. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-2> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-2> 9. DUT: Sends SOME/IP Notification Message 10. 11. DUT CONFIGURE: Trigger Event on <DIface-0> with the following

	informations - Event ID : <EVENT-ID-1-EG-ID-2> - Event Group ID : <EVENT-GROUP-ID-1-SI-2> - Service ID : <SERVICE-ID-2> 12. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-2> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-2> 13. DUT: Sends SOME/IP Notification Message 14. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-2>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 9. DUT: Sends SOME/IP Notification Message 13. DUT: Sends SOME/IP Notification Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.1.2 TCP Binding [TR_SOMEIP_00150] Page 33 (MUST)
Notes	

SOMEIPSRV_RPC_03: Getter of a field method

Synopsis	The getter of a field shall be a request/response call that has an empty payload in the request message and the value of the field in the payload of the response message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-GET-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-GET-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. TESTER: Extracts the content of SOME/IP Packet Length to <extractedPktLen> 9. TESTER: Verify that received SOME/IP Response Message contains: - SOME/IP Expected Response Message Payload is set to <METHOD-ID-GET-SI-1-RESP-PAYLOAD-SERIALIZED> 10. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message

	7. DUT: Sends SOME/IP Response Message 9. TESTER: Verify that received SOME/IP Response Message contains: - SOME/IP Expected Response Message Payload is set to <METHOD-ID-GET-SI-1-RESP-PAYLOAD-SERIALIZED>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.5 Fields [TR_SOMEIP_00181] Page 38 (MUST)
Notes	

SOMEIPSRV_RPC_04: Fire & forget requests

Synopsis	Requests without response message are called fire&forget. The implementation is basically the same as for Request/Response with the following differences: There is no response message. The message type is set to REQUEST_NO_RETURN (i.e. 0x01).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOMEIP_MSG_TYPE_REQUEST_NO_RETURN Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1>

	- SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Does not send SOME/IP Response Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Does not send SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.3 Fire&Forget Communication [TR_SOMEIP_00170] Page 36 (MUST)
Notes	

SOMEIPSRV_RPC_05: Fire & forget requests shall return no error

Synopsis	Fire & Forget messages shall not return an error. The system shall not return an error message for fire&forget methods.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOMEIP_MSG_TYPE_REQUEST_NO_RETURN Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <UNKNOWN-METHOD-ID-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID>

	- SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <UNKNOWN-METHOD-ID-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_UNKNOWN_METHOD 7. DUT: Does not send SOMEIP_MSG_TYPE_ERROR Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Does not send SOMEIP_MSG_TYPE_ERROR Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.3 Fire&Forget Communication [TR_SOMEIP_00171, TR_SOMEIP_00189] (MUST)
Notes	

SOMEIPSRV_RPC_06: Error handling the the bits of the return code

Synopsis	The Error Handling is based on an 8 Bit Std_returnType . The two most significant bits are reserved and shall be set to 0.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <UNKNOWN-METHOD-ID-SI-1>

	6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <UNKNOWN-METHOD-ID-SI-1> 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - Most Significant 2 bits in Return Code is set to 0 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - Most Significant 2 bits in Return Code is set to 0
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.6.2 Return Code [TR_SOMEIP_00187] Page 38 (MUST)
Notes	

SOMEIPSRV_RPC_07: Ignore the two most significant bits from return code

Synopsis	The receiver of a return code shall ignore the values of the two most significant bits.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing :

	- SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Packet Header Return Code set to 0xC0 - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.6.2 Return Code [TR_SOMEIP_00187] Page 38 (MUST)
Notes	

SOMEIPSRV_RPC_08: Do not reply to messages already carrying an error

Synopsis	Implementations shall not answer with errors to SOME/IP message already carrying an error (i.e. return code 0x01 - 0x1f).
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message

	5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Packet Header Return Code set to 0x01 - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> 7. DUT: Does not send SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Packet Header Return Code set to 0x1f - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> 10. DUT: Does not send SOMEIP_MSG_TYPE_ERROR Message 11. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Does not send SOMEIP_MSG_TYPE_ERROR Message 10. DUT: Does not send SOMEIP_MSG_TYPE_ERROR Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.6.2 Return Code [TR_SOMEIP_00539] Page 40 (MUST)
Notes	

SOMEIPSRV_RPC_09: No payload in Error message

Synopsis	For request/response methods the error message shall copy over the fields of the SOME/IP header (i.e. Message ID, Request ID, and Interface Version) but not the payload. In addition Message Type and Return Code have to be set to the appropriate values. NOTE: Checking for No Payload in Error message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <UNKNOWN-METHOD-ID-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <UNKNOWN-METHOD-ID-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_UNKNOWN_METHOD 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - SOME/IP Packet Header Length is set to 8 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message

	8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - SOME/IP Packet Header Length is set to 8
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.6.2 Return Code [TR_SOMEIP_00190] Page 39 (MUST)
Notes	

SOMEIPSRV_RPC_10: Do not return an error if Message Type is incorrect

Synopsis	The system shall not return an error message for events/notifications and fire&forget methods if the Message Type is set incorrectly to Request or Response. NOTE: Checking for Fire and Forget Method.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-FIRE-FORGET-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-FIRE-FORGET-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_NOT_OK

	7. DUT: Does not send SOMEIP_MSG_TYPE_ERROR Message 8. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Does not send SOMEIP_MSG_TYPE_ERROR Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.6.2 Return Code [TR_SOMEIP_00537] Page 39 (MUST)
Notes	

SOMEIPSRV_RPC_11: Setter of a field and payload

Synopsis	The setter of a field shall be a request/response call that has the desired valued of the field in the payload of the request message and the value that was set to field in the payload of the response message.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-2-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-2-SI-1>

	- SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - SOME/IP Packet Header Length is greater than 8 - SOME/IP Expected Response Message Payload is set to <METHOD-ID-2-SI-1-RESP-SERIALIZED> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOME/IP Response Message 8. TESTER: Verify that received SOME/IP Response Message contains: - SOME/IP Packet Header Length is greater than 8 - SOME/IP Expected Response Message Payload is set to <METHOD-ID-2-SI-1-RESP-SERIALIZED>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.5 Fields [TR_SOMEIP_00182] Page 38 (MUST)
Notes	

SOMEIPSRV_RPC_12: Notifier and event message on field change

Synopsis	The notifier shall send an event message that transports the value of a field on change and follows the rules for events.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 8. DUT: Sends SOME/IP Notification Message 9. DUT CONFIGURE: Trigger Event on <DIface-0> with the following informations - Event ID : <EVENT-ID-1-EG-ID-1> - Event Group ID : <EVENT-GROUP-ID-1-SI-1>

	- Service ID : <SERVICE-ID-1> 10. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-1> 11. DUT: Sends SOME/IP Notification Message 12. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 11. DUT: Sends SOME/IP Notification Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.5 Fields [TR_SOMEIP_00183] Page 38 (MUST)
Notes	

SOMEIPSRV_RPC_13: Different services can share the same port

Synopsis	While different Services shall be able to share the same port number of the transport layer protocol used, multiple Service-Instances of the same service on one single ECU shall listen on different ports per Service-Instance. NOTE: Checking For case - Different Services same port.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-2> - Number Of Instances : 1 3. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - Service ID in Entry Array set to <SERVICE-ID-2> 4. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> - Service ID in Entry Array set to <SERVICE-ID-2> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 1 5. DUT: Sends SOME/IP Notification Message 6. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option Transport Port to <extractedTransPort> 7. TESTER: Set Destination Port Of client1 to extractedTransPort 8. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 9. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0>

	`` - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 10. DUT: Sends SOME/IP Response Message 11. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-2-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-2> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-2> 12. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-2> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-2> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 13. DUT: Sends SOME/IP Response Message 14. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> 15. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-2> ```
Pass Criteria	5. DUT: Sends SOME/IP Notification Message 10. DUT: Sends SOME/IP Response Message 13. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.1.3 Multiple Service-Instances [TR_SOMEIP_00163] Page 35 (MAY)
Notes	

SOMEIPSRV_RPC_14: Different instances of the same service must use different ports

Synopsis	While different Services shall be able to share the same port number of the transport layer protocol used, multiple Service-Instances of the same service on one single ECU shall listen on different ports per Service-Instance. NOTE: Checking For case - Different instances of same Service different port.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 2 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Expected Offer Services' Service ID set to <SERVICE-ID-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 2 - Number Of Expected Offer Service Entries set to 2 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 2 to <extractedInstID2> 6. TESTER: Verify that received SOME/IP Notification Message contains: - Service Instance ID of Offer Service Entry 1 is not set to extractedInstID2 7. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option Number 1 Transport Port to <extractedTransPort1> 8. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option Number 2 Transport Port to <extractedTransPort2> 9. TESTER: Set Destination Port Of client1 to extractedTransPort1 10. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to

	<METHOD-ID-1-SI-1> 11. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 12. DUT: Sends SOME/IP Response Message 13. TESTER: Set Destination Port Of client1 to extractedTransPort2 14. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-2-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 15. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 16. DUT: Sends SOME/IP Response Message 17. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 6. TESTER: Verify that received SOME/IP Notification Message contains: - Service Instance ID of Offer Service Entry 1 is not set to extractedInstID2 12. DUT: Sends SOME/IP Response Message 16. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.1.3 Multiple Service-Instances [TR_SOMEIP_00163] Page 35 (MUST)
Notes	

SOMEIPSRV_RPC_15: Update on change value

Synopsis	Update on change - send an update as soon as a "value" changes (e.g. door open)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 8. DUT: Sends SOME/IP Notification Message 9. DUT CONFIGURE: Trigger Event on <DIface-0> with the following informations - Event ID : <EVENT-ID-1-EG-ID-1> - Event Group ID : <EVENT-GROUP-ID-1-SI-1> - Service ID : <SERVICE-ID-1> 10. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on

	<DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-1-EG-ID-1> 11. DUT: Sends SOME/IP Notification Message 12. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 11. DUT: Sends SOME/IP Notification Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.4.1 Strategy for sending notifications [TR_SOMEIP_00176] Page 37 (MUST)
Notes	

SOMEIPSRV_RPC_16: Cyclic update of a value

Synopsis	Cyclic update - send an updated value in a fixed interval (e.g. every 300 ms)
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 1 to <extractedInstID1> 6. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE - Service ID in Subscribe Eventgroup Entry set to <SERVICE-ID-1> - EventGroup ID in Subscribe Eventgroup Entry set to <EVENT-GROUP-ID-1-SI-1> - Service Instance ID in Subscribe Eventgroup Entry set to extractedInstID1 7. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_SUBSCRIBE_ACK - SOME/IP Expected Service ID in Subscribe EventGroup Ack Entry set to <SERVICE-ID-1> - SOME/IP Expected Service Instance ID in Subscribe EventGroup Ack Entry set to extractedInstID1 - SOME/IP Expected EventGroup ID in Subscribe EventGroup Ack Entry set to <EVENT-GROUP-ID-1-SI-1> 8. DUT: Sends SOME/IP Notification Message 9. DUT CONFIGURE: Trigger Event on <DIface-0> with the following informations - Event ID : <EVENT-ID-2-EG-ID-1> - Event Group ID : <EVENT-GROUP-ID-1-SI-1> - Service ID : <SERVICE-ID-1> 10. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on

	<DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK - SOME/IP Expected Event ID in SOME/IP Header set to <EVENT-ID-2-EG-ID-1> 11. DUT: Sends 2 SOME/IP Notification Messages 12. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is greater than (<EVENT-ID-2-EG-ID-1-CYCLIC-TIME> - <ParamToleranceTimeMillisec>) micro second 13. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 8. DUT: Sends SOME/IP Notification Message 11. DUT: Sends 2 SOME/IP Notification Messages 12. TESTER: Verify that the time interval between reception of 1st SOME/IP Notification Message and 2nd SOME/IP Notification Message is greater than (<EVENT-ID-2-EG-ID-1-CYCLIC-TIME> - <ParamToleranceTimeMillisec>) micro second
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.4.1 Strategy for sending notifications [TR_SOMEIP_00176] Page 37 (MUST)
Notes	

SOMEIPSRV_RPC_17: Multiple instances use multiple TCP connections

Synopsis	When having more than one instance of a service a TCP connection per services instance is needed.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 2 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Expected Offer Services' Service ID set to <SERVICE-ID-1> - SOMEIP Expected Type of Option set to SOMEIP_OPTION_IPV4_ENDPOINT - SOMEIP Expected Number Of IPv4 Endpoint Option set to 2 - Number Of Expected Offer Service Entries set to 2 4. DUT: Sends SOME/IP Notification Message 5. TESTER: Extracts the content of Service Instance ID of Offer Service Entry 2 to <extractedInstID2> 6. TESTER: Verify that received SOME/IP Notification Message contains: - Service Instance ID of Offer Service Entry 1 is not set to extractedInstID2 7. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option Number 1 Transport Port to <extractedTransPort1> 8. TESTER: Extracts the content of SOMEIP IPv4 Endpoint Option Number 2 Transport Port to <extractedTransPort2> 9. TESTER: Set Destination Port Of client1 to extractedTransPort1 10. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1>

	- SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 11. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 12. DUT: Sends SOME/IP Response Message 13. TESTER: Set Destination Port Of client1 to extractedTransPort2 14. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <METHOD-ID-1-SI-1> 15. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <METHOD-ID-1-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_OK 16. DUT: Sends SOME/IP Response Message 17. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 6. TESTER: Verify that received SOME/IP Notification Message contains: - Service Instance ID of Offer Service Entry 1 is not set to extractedInstID2 12. DUT: Sends SOME/IP Response Message 16. DUT: Sends SOME/IP Response Message
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.1.2 TCP Binding [TR_SOMEIP_00150] Page 33 (MUST)
Notes	

SOMEIPSRV_RPC_18: In Response copy Message ID

Synopsis	For request/response methods the error message shall copy over the fields of the SOME/IP header (i.e. Message ID, Request ID, and Interface Version) but not the payload. In addition Message Type and Return Code have to be set to the appropriate values. NOTE: Checking For Message ID.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <UNKNOWN-METHOD-ID-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_UNKNOWN_METHOD 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - SOME/IP Expected Service ID is set to <SERVICE-ID-1> - SOME/IP Expected Method ID is set to <UNKNOWN-METHOD-ID-SI-1> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message

	8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - SOME/IP Expected Service ID is set to <SERVICE-ID-1> - SOME/IP Expected Method ID is set to <UNKNOWN-METHOD-ID-SI-1>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.6.2 Return Code [TR_SOMEIP_00190] Page 39 (MUST)
Notes	

SOMEIPSRV_RPC_19: In Response copy Request ID

Synopsis	For request/response methods the error message shall copy over the fields of the SOME/IP header (i.e. Message ID, Request ID, and Interface Version) but not the payload. In addition Message Type and Return Code have to be set to the appropriate values. NOTE: Checking For Request ID.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1> - SOME/IP Send Request Message Method ID set to <UNKNOWN-METHOD-ID-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <UNKNOWN-METHOD-ID-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_UNKNOWN_METHOD 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message

	8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - SOME/IP Packet Expected Header Request ID is set to <CLIENT1-CURR-REQUEST-ID> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - SOME/IP Packet Expected Header Request ID is set to <CLIENT1-CURR-REQUEST-ID>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.6.2 Return Code [TR_SOMEIP_00190] Page 39 (MUST)
Notes	

SOMEIPSRV_RPC_20: In Response copy Interface Version

Synopsis	For request/response methods the error message shall copy over the fields of the SOME/IP header (i.e. Message ID, Request ID, and Interface Version) but not the payload. In addition Message Type and Return Code have to be set to the appropriate values. NOTE: Checking For Interface Version.
Prerequisites	Check section prerequisites
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. DUT CONFIGURE: Start Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1> - Number Of Instances : 1 2. TESTER: <CLIENT-1> Sends SOME/IP Notification Message to DUT through <DIface-0> containing : - Entry Type set to SOMEIP_ENTRY_FIND_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 3. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - Entry Type set to SOMEIP_ENTRY_OFFER_SERVICE - Service ID in Entry Array set to <SERVICE-ID-1> 4. DUT: Sends SOME/IP Notification Message 5. TESTER: <CLIENT-1> Sends SOME/IP Request Message to DUT through <DIface-0> containing : - SOME/IP Packet Send Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Packet Send Header Interface Version set to <SERVICE-ID-1-INTF-VER-MAJ> - SOME/IP Send Request Message Service ID set to <SERVICE-ID-1>

	- SOME/IP Send Request Message Method ID set to <UNKNOWN-METHOD-ID-SI-1> 6. TESTER: <CLIENT-1> Listens (upto <ParamListenTime>) on <DIface-0> - SOME/IP Packet Expected Header Request ID set to <CLIENT1-CURR-REQUEST-ID> - SOME/IP Expected Service ID set to <SERVICE-ID-1> - SOME/IP Expected Method ID set to <UNKNOWN-METHOD-ID-SI-1> - SOME/IP Expected Return Code set to SOMEIP_RET_CODE_E_UNKNOWN_METHOD 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - Interface Version is set to <SERVICE-ID-1-INTF-VER-MAJ> 9. DUT CONFIGURE: Stop Service on <DIface-0> with the following informations - Service ID : <SERVICE-ID-1>
Pass Criteria	4. DUT: Sends SOME/IP Notification Message 7. DUT: Sends SOMEIP_MSG_TYPE_ERROR Message 8. TESTER: Verify that received SOMEIP_MSG_TYPE_ERROR Message contains: - Interface Version is set to <SERVICE-ID-1-INTF-VER-MAJ>
Reference	Example for a Serialization Protocol (SOME/IP) V1.1.0 R4.1 Rev 3 s6.3.6.2 Return Code [TR_SOMEIP_00190] Page 39 (MUST)
Notes	

5.1.6 Test Cases ETS

5.1.6.1.1 SOMEIP_ETS_01: Array_Length_longer_as_message_length_allows_it

Synopsys	Check that the DUT gives back an error message when the array length is longer then the SOMEIP length allows it.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message with Array Length longer than allowed by message length 2. TESTER: send SOME/IP Message using method echoUINT8 3. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00042 PRS_SOMEIP_00099 PRS_SOMEIP_00100
Notes	The test will also pass if the DUT ignores the request.

5.1.6.1.2 SOMEIP_ETS_02: Array_Length_too_long

Synopsys	Check that the DUT gives back an error message when the array length is longer than the actual array.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message with Array Length longer than the actual array 2. TESTER: send SOME/IP Message using method echoUINT8 3. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00902;PRS_SOMEIP_00099
Notes	The test will also pass if the DUT ignores the request.

5.1.6.1.3 SOMEIP_ETS_03: Array_Length_too_short_strips_Payload

Synopsys	Check that the DUT strips the payload of the array to the array length when the array length is shorter then the actual array.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message with Array Length shorter then the actual array 2. TESTER: send SOME/IP Message using method echoUINT8 3. DUT: returns method response Message with stripped Payload
Pass Criteria	DUT: returns method response Message with stripped Payload
Reference	PRS_SOMEIP_00099
Notes	

5.1.6.1.4 SOMEIP_ETS_04: Burst_Test

Synopsys	Check that the DUT can handle a burst of requests in a short time and process them to give back an answer to all of them.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create valid SOME/IP Message 2. TESTER: send burst of SOME/IP Message requests using method echoUINT8 3. DUT: returns method response Message response to each request
Pass Criteria	DUT: returns method response Message response to each request
Reference	PRS_SOMEIP_00065
Notes	

5.1.6.1.5 SOMEIP_ETS_05: checkByteOrder

Synopsys	Check byte order handling of parameters (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service

Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message with two parameters UINT8 and UINT16 2. TESTER: send SOME/IP Message using method checkByteOrder 3. DUT: returns method response message with valid UINT32 value = sum of the two passed Parameters (UINT8 + UINT16) in the Request
Pass Criteria	DUT: returns method response message with valid UINT32 value = sum of the two passed Parameters (UINT8 + UINT16) in the Request
Reference	PRS_SOMEIP_00191;PRS_SOMEIP_00065
Notes	

5.1.6.1.6 SOMEIP_ETS_07: echoBitfields

Synopsys	Check the bitfields and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoBitfields 3. DUT: returns method response message with bitfields in reversed order per bitfield compared to the reuquest
Pass Criteria	DUT: returns method response message with bitfields in reversed order per bitfield compared to the reuquest
Reference	PRS_SOMEIP_00191;PRS_SOMEIP_003001;PRS_SOMEIP_00300
Notes	

5.1.6.1.7 SOMEIP_ETS_08: echoCommonDatatypes

Synopsys	Check Common Datatypes (boolean, UINT8/16/32, INT8/16/32, float32) parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method

	echoCommonDatatypes 3. DUT: returns method response message with same data (values and order) as in the request
Pass Criteria	DUT: returns method response message with same content (values and order) as in the request
Reference	PRS_SOMEIP_00065
Notes	

5.1.6.1.8 SOMEIP_ETS_09: echoENUM

Synopsys	Check Enum parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoENUM 3. DUT: returns method response message with same data (values and order) as in the request
Pass Criteria	DUT: returns method response message with same content (values and order) as in the request
Reference	PRS_SOMEIP_00191
Notes	

5.1.6.1.9 SOMEIP_ETS_12: echoExtDatatypes16BitLength_StructureOfStrings

Synopsys	Check echoExtendedDatatypes16BitLengthAndTypeField parameters and their order (sending and receiving). The method returns the passed parameter with a 16Bit Length and Type field.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method echoExtendedDatatypes16BitLengthAndTypeField 2. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request

Reference	PRS_SOMEIP_00079
Notes	

5.1.6.1.10 SOMEIP_ETS_13: echoExtDatatypes16BitLength_StructureOfStringUnion

Synopsys	Check echoExtendedDatatypes16BitLengthAndTypeField parameters and their order (sending and receiving). The method returns the passed parameter with a 16Bit Length and Type field.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method echoExtendedDatatypes16BitLengthAndTypeField 2. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00079
Notes	

5.1.6.1.11 SOMEIP_ETS_14: echoExtDatatypes8BitLength_StructureOfStrings

Synopsys	Check echoExtendedDatatypes8BitLengthAndTypeField parameters and their order (sending and receiving). The method returns the passed parameter with a 8Bit Length and Type field.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method echoExtendedDatatypes8BitLengthAndTypeField 2. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00079
Notes	

5.1.6.1.12 SOMEIP_ETS_15: echoExtDatatypes8BitLength_StructureOfStringUnion

Synopsys	Check echoExtendedDatatypes8BitLengthAndTypeField parameters and their order (sending and receiving). The method returns the passed parameter with a 8Bit Length and Type field.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method echoExtendedDatatypes8BitLengthAndTypeField 2. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00191
Notes	

5.1.6.1.13 SOMEIP_ETS_19: echoFLOAT64

Synopsys	Check echoFLOAT64 parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	3. TESTER: create SOME/IP Message 4. TESTER: send SOME/IP Message using method echoFLOAT64 5. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00065
Notes	

5.1.6.1.14 SOMEIP_ETS_21: echoINT8

Synopsys	Check echoINT8 parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service

Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoINT8 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00065
Notes	

5.1.6.1.15 SOMEIP_ETS_22: echoStaticUINT8Array_One_Dimensional

Synopsys	Check echoStaticUINT8Array_One-Dimensional parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoStaticUINT8Array 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00099;PRS_SOMEIP_00100
Notes	

5.1.6.1.16 SOMEIP_ETS_25: echoTYPEDEF

Synopsys	Check echoTypedef parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message

	2. TESTER: send SOME/IP Message using method echoTypedef 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00065
Notes	

5.1.6.1.17 SOMEIP_ETS_27: echoUINT8

Synopsys	Check echoUINT8 parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00052;PRS_SOMEIP_00053;PRS_SOMEIP_00058
Notes	

5.1.6.1.18 SOMEIP_ETS_28: echoUINT8Array

Synopsys	Check echoUINT8Array parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8Array 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00375;PRS_SOMEIP_00377;

	PRS_SOMEIP_00114;PRS_SOMEIP_00052;PRS_SOMEIP_00053
Notes	

5.1.6.1.19 SOMEIP_ETS_29: echoUINT8Array16Bitlength

Synopsys	Check echoUINT8Array16BitLength parameters and that the length field is 16 Bit long and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8Array16Bitlength 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00375;PRS_SOMEIP_00377; PRS_SOMEIP_00052;PRS_SOMEIP_00107
Notes	

5.1.6.1.20 SOMEIP_ETS_30: echoUINT8Array2Dim

Synopsys	Check echoUINT8Array2Dim parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8Array2Dim 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00101;PRS_SOMEIP_00102; PRS_SOMEIP_00377;PRS_SOMEIP_00114;PRS_SOMEIP_00052
Notes	

5.1.6.1.21 SOMEIP_ETS_31: echoUINT8Array8Bitlength

Synopsys	Check echoUINT8Array8BitLength parameters and that the length field is 8 Bit long and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8Array8Bitlength 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00099;PRS_SOMEIP_00100;PRS_SOMEIP_00052
Notes	

5.1.6.1.22 SOMEIP_ETS_32: echoUINT8ArrayMinSize

Synopsys	Check echoUINT8ArrayMinSize parameters and their order (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8ArrayMinSize 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00377;PRS_SOMEIP_00114
Notes	

5.1.6.1.23 SOMEIP_ETS_33: echoUINT8ArrayMinSize_too_short

Synopsys	Check that the DUT can handle an echoUINT8ArrayMinSize which is shorter than 3 elements of the array.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message with a too short array using method echoUINT8ArrayMinSize 3. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00377;PRS_SOMEIP_00114
Notes	

5.1.6.1.24 SOMEIP_ETS_34: echoUINT8E2E

Synopsys	The method returns the transferred UINT8 value back to the invoker. The method is E2E protected.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8E2E 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00065
Notes	

5.1.6.1.25 SOMEIP_ETS_35: echoUINT8RELIABLE

Synopsys	Check echoUINT8RELIABLE parameters and their order and check for handling a tcp connection. (sending and receiving)

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8RELIABLE 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00065;PRS_SOMEIP_00708
Notes	

5.1.6.1.26 SOMEIP_ETS_36: echoUINT8RELIABLE_Check_Magic_Cookies

Synopsys	Check that the DUT sends out magic cookie messages by sending a lot of SOMEIP messages over TCP
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUINT8RELIABLE 3. DUT: returns method response message with same value as in the request 4. DUT: sends Magic cookies
Pass Criteria	DUT: returns method response message with same value as in the request DUT: sends Magic cookies
Reference	PRS_SOMEIP_00154;PRS_SOMEIP_00161
Notes	

5.1.6.1.27 SOMEIP_ETS_37: echoUINT8RELIABLE_client_closes_TCP_connection_automatically

Synopsys	Client builds up a connection and sends a reset interface message - DUT sends a FIN,ACK when the service is stopped - Client shall close tcp connection automatically according to FIN,ACK from DUT -> tcp connection successful

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send a reset interface message 2. DUT: sends a FIN,ACK when the service is stopped
Pass Criteria	DUT: sends a FIN,ACK when the service is stopped
Reference	PRS_SOMEIP_00708;PRS_SOMEIP_00711
Notes	

5.1.6.1.28 SOMEIP_ETS_38: echoUNION

Synopsys	Check echoUNION parameters and their order. (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUNION 3. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00119;PRS_SOMEIP_00126
Notes	

5.1.6.1.29 SOMEIP_ETS_39: echoUTF16DYNAMIC

Synopsys	Check echoUNION parameters and their order. (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message using method echoUTF16DYNAMIC 3. DUT: returns method response message with same value

	as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00093;PRS_SOMEIP_00372; PRS_SOMEIP_00084;PRS_SOMEIP_00085; PRS_SOMEIP_00087;PRS_SOMEIP_00091; PRS_SOMEIP_00092;PRS_SOMEIP_00094; PRS_SOMEIP_00095
Notes	

5.1.6.1.30 SOMEIP_ETS_40: echoUTF16DYNAMIC_length_too_long_for_String

Synopsys	Tester sending out a length which is longer than the actual string and DUT should give back an Error message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method echoUTF16DYNAMIC with too big length 2. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00372
Notes	

5.1.6.1.31 SOMEIP_ETS_41: echoUTF16DYNAMIC_length_too_short_for_malformed_String

Synopsys	According to the SOME/IP specification for UTF-16 strings with dynamic length, devices receiving this type of strings shall strip the total amount of bytes received to fit the exact length specified int the string's length field. This test case will request the DUT to reply a echoUTF16DYNAMIC() as defined in the Testability Service, carrying a dynamic UTF-16 string whose specified length forces the DUT to reduce the received string to only 2 bytes , resulting in an invalid payload consisting only of an incomplete BOM.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: send SOME/IP Message using method echoUTF16DYNAMIC with i. Source MAC set to the tester's MAC address ii. destination MAC set to the DUT's MAC address iii. Source Port set to 30492 iv. destination port set to the UDP endpoint option provided by the DUT's OfferService for the Testability Service. v. Source IP set to the tester's IP address vi. destination IP set to the UDP endpoint option provided by the DUT's OfferService for the Testability Service. vii. Service-ID set to 0x0101 viii. Instance-Id set to 0x01 ix. Method-ID set to 0x0016 x. Payload: 0x00 00 00 80 FE FF 00 48 00 00 2. DUT: returns method response message with same value as in the request 3. TESTER: repeat step 1 but changing the payload of the echoUTF16Dynamic request to: xi. 0x00 00 00 02 FE FF 00 48 00 00 4. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns method response message with same value as in the request DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00093;PRS_SOMEIP_00372; PRS_SOMEIP_00084;PRS_SOMEIP_00085; PRS_SOMEIP_00087;PRS_SOMEIP_00091; PRS_SOMEIP_00092;PRS_SOMEIP_00094; PRS_SOMEIP_00095
Notes	

5.1.6.1.32 SOMEIP_ETS_42: echoUTF16DYNAMIC_length_too_short_for_String

Synopsys	Check that the DUT rejects with a malformed message an echoUTF16DYNAMIC

	string which has a length that is shorter than the actual string.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: create SOME/IP Message 2. TESTER: send SOME/IP Message with a length which is shorter than the actual string using method echoUTF16DYNAMIC 3. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00372
Notes	

5.1.6.1.33 SOMEIP_ETS_43: echoUTF16DYNAMIC_odd_number_before_termination

Synopsys	Check that the DUT can handle an echoUTF16DYNAMIC string which has an odd number by sending a byte more before the termination and the DUT should give back an error message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message which has an odd number by sending one additional byte before the termination using method echoUTF16DYNAMIC 2. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00372;PRS_SOMEIP_00085; PRS_SOMEIP_00086;PRS_SOMEIP_00092
Notes	

5.1.6.1.34 SOMEIP_ETS_44: echoUTF16DYNAMIC_with_odd_number_after_termination

Synopsys	Check that the DUT can handle an echoUTF16DYNAMIC string which has an odd number by sending a byte more after the termination and the DUT should strip this byte away.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message which has an odd number by sending one additional byte after the termination using method echoUTF16DYNAMIC 2. DUT: returns method response message with odd number after the termination and corrected values
Pass Criteria	DUT: returns method response message with odd number after the termination and corrected values
Reference	PRS_SOMEIP_00372;PRS_SOMEIP_00085; PRS_SOMEIP_00086;PRS_SOMEIP_00092
Notes	

5.1.6.1.35 SOMEIP_ETS_45: echoUTF16DYNAMIC_wrong_BOM

Synopsys	Check that the DUT can handle an echoUTF16DYNAMIC string which has a wrong BOM and the DUT should give back an error.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message with a wrong BOM using method echoUTF16DYNAMIC 2. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00087
Notes	

5.1.6.1.36 SOMEIP_ETS_46: echoUTF16FIXED

Synopsys	Check echoUTF16FIXED parameters and their order. (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method

	echoUTF16FIXED 2. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request
Reference	PRS_SOMEIP_00373;PRS_SOMEIP_00084; PRS_SOMEIP_00085;PRS_SOMEIP_00087
Notes	

5.1.6.1.37 SOMEIP_ETS_47: echoUTF16FIXED_with_odd_number

Synopsys	Check that the DUT can handle an echoUTF16FIXED string which has an odd number by sending a byte more after the termination and the DUT should strip this byte away.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	3. TESTER: send SOME/IP Message which has an odd number by sending one additional byte after the termination using method echoUTF16FIXED 1. DUT: returns method response message with odd number after the termination and without the additional byte
Pass Criteria	DUT: returns method response message with odd number after the termination and without the additional byte
Reference	PRS_SOMEIP_00085;PRS_SOMEIP_00086
Notes	

5.1.6.1.38 SOMEIP_ETS_48: echoUTF8DYNAMIC

Synopsys	Check echoUTF8DYNAMIC parameters and their order. (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method echoUTF8DYNAMIC 2. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the

	request
Reference	PRS_SOMEIP_00093;PRS_SOMEIP_00372; PRS_SOMEIP_00087;PRS_SOMEIP_00091; PRS_SOMEIP_00092;PRS_SOMEIP_00094; PRS_SOMEIP_00095
Notes	

5.1.6.1.39 SOMEIP_ETS_49: echoUTF8DYNAMIC_length_too_long_for_String

Synopsys	Check that the DUT can handle an echoUTF8DYNAMIC string which has a length that is longer than the actual string and the DUT should give back an error.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message which has a length that is longer than the actual string using method echoUTF8DYNAMIC 2. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00372
Notes	

5.1.6.1.40 SOMEIP_ETS_50: echoUTF8DYNAMIC_length_too_short_for_malformed_String

Synopsys	According to the SOME/IP specification for UTF-8 strings with dynamic length, ECUs receiving this type of strings shall strip the total amount of bytes received to fit the exact length specified in the string's length field. This test case will request the DUT to reply an echoUTF8DYNAMIC() as defined in the Testability Service, carrying a dynamic UTF-8 string whose specified length forces the DUT to reduce the received string to only 2 bytes , resulting in an invalid payload consisting only of an incomplete BOM.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method echoUTF16DYNAMIC with xii. Source MAC set to the tester's MAC address

	xiii. destination MAC set to the DUT's MAC address xiv. Source Port set to 30492 xv. destination port set to the UDP endpoint option provided by the DUT's OfferService for the Testability Service. xvi. Source IP set to the tester's IP address xvii. destination IP set to the UDP endpoint option provided by the DUT's OfferService for the Testability Service. xviii. Service-ID set to 0x0101 xix. Instance-Id set to 0x01 xx. Method-ID set to 0x0015 xxi. Payload: 0x00 00 00 80 EF BB BF 48 00 2. DUT: returns method response message with same value as in the request 3. TESTER: repeat step 1 but changing the payload of the echoUTF16Dynamic request to: xxii. 0x00 00 00 02 EF BB BF 48 00 4. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns method response message with same value as in the request DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00093;PRS_SOMEIP_00372; PRS_SOMEIP_00087;PRS_SOMEIP_00091; PRS_SOMEIP_00092;PRS_SOMEIP_00094; PRS_SOMEIP_00095
Notes	

5.1.6.1.41 SOMEIP_ETS_51: echoUTF8DYNAMIC_length_too_short_for_String

Synopsys	Check that the DUT can handle an echoUTF8DYNAMIC string which has a length that is shorter than the actual string.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1

Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message which has a length that is shorter than the actual string using method echoUTF8DYNAMIC 2. DUT: returns method response message with data string stripped to the length field value
Pass Criteria	DUT: returns method response message with data string stripped to the length field value
Reference	PRS_SOMEIP_00372
Notes	

5.1.6.1.42 SOMEIP_ETS_52: echoUTF8DYNAMIC_wrong_BOM

Synopsys	Check that the DUT can handle an echoUTF8DYNAMIC string which has a wrong BOM and the DUT should give back an error.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message with wrong BOM using method echoUTF8DYNAMIC 2. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00372;PRS_SOMEIP_00087; PRS_SOMEIP_00091;PRS_SOMEIP_00092; PRS_SOMEIP_00094;PRS_SOMEIP_00095
Notes	

5.1.6.1.43 SOMEIP_ETS_53: echoUTF8FIXED

Synopsys	Check echoUTF8FIXED parameters and their order. (sending and receiving)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message using method echoUTF8FIXED 2. DUT: returns method response message with same value as in the request
Pass Criteria	DUT: returns method response message with same value as in the request

Reference	PRS_SOMEIP_00373;PRS_SOMEIP_00087
Notes	

5.1.6.1.44 SOMEIP_ETS_54: Length_equals_0_Test

Synopsys	Check that the DUT gives back an error message when the SOMEIP length equals zero.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send SOME/IP Message` with length zero using `method` echoUINT8 2. `DUT: returns Error Message MALFORMED_MESSAGE`
Pass Criteria	`DUT: returns Error Message MALFORMED_MESSAGE`
Reference	PRS_SOMEIP_00042
Notes	

5.1.6.1.45 SOMEIP_ETS_55: Length_smaller_than_8_Test

Synopsys	Check that the DUT gives back an error message when the SOMEIP length is less than 8 bytes.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send SOME/IP Message` with length less than 8 bytes using `method` echoUINT8 2. `DUT: returns Error Message MALFORMED_MESSAGE`
Pass Criteria	`DUT: returns Error Message MALFORMED_MESSAGE`
Reference	PRS_SOMEIP_00042
Notes	

5.1.6.1.46 SOMEIP_ETS_58: Length_way_too_long

Synopsys	Tester sends a much longer SOME/IP length than the current payload with a few requests and DUT should give back an Error message or directly ignore it.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOMEIP messages with a much longer SOME/IP length than the current payload 2. DUT: returns Error Message MALFORMED_MESSAGE or ignores the Message
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE or ignores the Message
Reference	PRS_SOMEIP_00902;PRS_SOMEIP_00191
Notes	

5.1.6.1.47 SOMEIP_ETS_59: ResetInterface_wrong_Fire_and_forget_package_get_No_Error_back

Synopsys	Check that the DUT doesn't answer or send an error message to a wrong Fire&Forget message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: use method TestFieldUINT8 with Field Getter method 2. DUT: returns the requested field value 3. TESTER: store the returned value and increment it 4. TESTER: use method TestFieldUINT8 with Setter method and set the incremented value 5. DUT: return the field value which was set by the Tester 6. TESTER: trigger a reset of the interface by using method resetInterface 7. TESTER: use method TestFieldUINT8 with Field Getter method to request the field value again 8. DUT: returns the incremented value which was set by the Tester in step 4
Pass Criteria	DUT: returns the incremented value which was set by the Tester in step 4
Reference	PRS_SOMEIP_00701;PRS_SOMEIP_00170; PRS_SOMEIP_00171;PRS_SOMEIP_00189
Notes	

5.1.6.1.48 SOMEIP_ETS_60: SD_Discover_Port_and_IP

Synopsys	The DUT has to answer the Testers Multicast FindService with a Unicast OfferService, stating all IPs and Ports needed to fulfill all sort of possible Comunications with it.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send` Multicast FindService 2. `DUT: returns` Unicast OfferService, stating all IPs and Ports needed to fulfill all sort of possible Comunications with it.
Pass Criteria	`DUT: returns` Unicast OfferService, stating all IPs and Ports needed to fulfill all sort of possible Comunications with it.
Reference	PRS_SOMEIPSD_00310;PRS_SOMEIPSD_00357; PRS_SOMEIPSD_00358;PRS_SOMEIPSD_00361; PRS_SOMEIPSD_00362
Notes	

5.1.6.1.49 SOMEIP_ETS_61: Sending_two_SOMEIP_Messages_in_a_row

Synopsys	Sending three SOMEIP messages in one UDP package. DUT has to reply on all three SOMEIP messages and send the correct response. Check the handling when more SOMEIP messages are in one transport protocol frame.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	3. `TESTER: send two SOMEIP messages in one UDP frame e.g. using the methods echoUINT8 and echoENUM in combination` 4. `DUT: returns the method results of both requests in one message or separate messages`
Pass Criteria	`DUT: returns the method results of both requests in one message or separate messages`
Reference	PRS_SOMEIPSD_00310;PRS_SOMEIPSD_00357; PRS_SOMEIPSD_00358;PRS_SOMEIPSD_00361;

	PRS_SOMEIPSD_00362
Notes	

5.1.6.1.50 SOMEIP_ETS_63: String_UTF16FIXED_too_long_strips_to_64_Byte

Synopsys	Check that the DUT strips a UTF16FIXED string which is too long to 64 Byte. Check string length handling.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send a echoUTF16FIXED message with unrepairable payload. 2. DUT: returns Error Message MALFORMED_MESSAGE 3. TESTER: send a echoUTF16FIXED message with repairable payload 4. DUT: answers with a correctly repaired message
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE DUT: answers with a correctly repaired message
Reference	PRS_SOMEIP_00373
Notes	

5.1.6.1.51 SOMEIP_ETS_64: String_UTF16FIXED_too_short

Synopsys	Check that the DUT which has a UTF16FIXED string which is shorter than 64 Byte gives back an error message
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	5. TESTER: send SOME/IP Message with string which is shorter than 64 Byte using method echoUTF16FIXED 6. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00373
Notes	

5.1.6.1.52 SOMEIP_ETS_65: String_UTF8FIXED_too_long_strips_to_64_Byte

Synopsys	Check that the DUT strips a UTF8FIXED string which is too large (larger than 64 Byte)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send SOME/IP Message with` string which is lager than 64 Byte `using method` echoUTF16FIXED 2. `DUT: returns method response message with payload stripped to 64 Byte`
Pass Criteria	`DUT: returns method response message with payload stripped to 64 Byte`
Reference	PRS_SOMEIP_00373
Notes	

5.1.6.1.53 SOMEIP_ETS_66: String_UTF8FIXED_too_short

Synopsys	Check that the DUT which has a UTF8FIXED string which is shorter than 64 Byte gives back an error message
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send SOME/IP Message with` string which is shorter than 64 Byte `using method` UTF8FIXED 2. `DUT: returns Error Message MALFORMED_MESSAGE`
Pass Criteria	`DUT: returns Error Message MALFORMED_MESSAGE`
Reference	PRS_SOMEIP_00373
Notes	

5.1.6.1.54 SOMEIP_ETS_67: UINT8Array_with_Length_0_strips_Payload

Synopsys	Check that the DUT strips the array to no payload when the array length equals zero.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message with array length equal to zero using method UINT8Array 2. DUT: returns method response message with payload length zero (no payload)
Pass Criteria	DUT: returns method response message with payload length zero (no payload)
Reference	PRS_SOMEIP_00375;PRS_SOMEIP_00377;PRS_SOMEIP_00114
Notes	

5.1.6.1.55 SOMEIP_ETS_68: Unaligned_SOMEIP_Messages_overTCP

Synopsys	Sending three SOMEIP messages in one TCP package and one SOMEIP message is unaligned. DUT has to reply to all three SOMEIP messages and send the correct response.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send three SOME/IP Messages in one TCP package with one unaligned SOMEIP message using method echoUINT8array 2. DUT: returns method response messages to all three requests
Pass Criteria	DUT: returns method response messages to all three requests
Reference	PRS_SOMEIP_00142;PRS_SOMEIP_00569
Notes	

5.1.6.1.56 SOMEIP_ETS_69: Unaligned_SOMEIP_Messages_overUDP

Synopsys	Sending three SOMEIP messages in one UDP package and one SOMEIP message is unaligned. DUT hast to reply on all three SOMEIP messages and send the correct response.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service

Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send three SOME/IP Messages in one UDP package with one unaligned SOMEIP message using method echoUINT8array 2. DUT: returns method response messages to all three requests
Pass Criteria	DUT: returns method response messages to all three requests
Reference	PRS_SOMEIP_00142;PRS_SOMEIP_00569
Notes	

5.1.6.1.57 SOMEIP_ETS_70: Union_Length_longer_as_mesage_length_allows_it

Synopsys	Check that the DUT gives back an error message when the union length is longer then the SOMEIP length allows it.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message where the union length is longer then the SOMEIP length allows it using method echoUNION 2. DUT: returns Error Message MALFORMED_MESSAGE
Pass Criteria	DUT: returns Error Message MALFORMED_MESSAGE
Reference	PRS_SOMEIP_00119;PRS_SOMEIP_00126
Notes	

5.1.6.1.58 SOMEIP_ETS_71: Union_Length_too_long

Synopsys	Check that the DUT gives back an error message when the union length is longer than the

	actual union.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send SOME/IP Message` where the union length is longer than the actual union using method echoUNION 2. `DUT: returns Error Message MALFORMED_MESSAGE`
Pass Criteria	`DUT: returns Error Message MALFORMED_MESSAGE`
Reference	PRS_SOMEIP_00119;PRS_SOMEIP_00126
Notes	

5.1.6.1.59 SOMEIP_ETS_72: Union_Length_too_short

Synopsys	Tester sending out a request where the union length is shorter than the actual message length allows it and the DUT should give back an Error message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send SOME/IP Message` where the union length is is shorter than the actual message length allows it using method echoUNION 2. `DUT: returns Error Message MALFORMED_MESSAGE`
Pass Criteria	`DUT: returns Error Message MALFORMED_MESSAGE`
Reference	PRS_SOMEIP_00119;PRS_SOMEIP_00126
Notes	

5.1.6.1.60 SOMEIP_ETS_73: Union_with_wrong_type_field_for_union_member_Padding

Synopsys	Check that the DUT gives back the correct Union for the type field (with or without padding) when the union has the wrong type field for the payload.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send SOME/IP Message` where the union has the wrong type field for the payload using method echoUNION 2. `DUT: returns method response messages with the correct union type and adjusted padding`
Pass Criteria	`DUT: returns method response messages with the correct union type and adjusted padding`
Reference	PRS_SOMEIP_00119;PRS_SOMEIP_00126
Notes	

5.1.6.1.61 SOMEIP_ETS_74: Wrong_Interface_Version

Synopsys	Check that the DUT gives back an error message or ignores it when a wrong interface version arrives in a request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. `TESTER: send SOME/IP Message` with wrong Interface Version using method echoUINT8 2. `DUT:` returns Error Message WRONG_INTERFACE_VERSION or ignores the request
Pass Criteria	`DUT:` returns Error Message WRONG_INTERFACE_VERSION or ignores the request

Reference	PRS_SOMEIP_00053;PRS_SOMEIP_00058; PRS_SOMEIP_00190;PRS_SOMEIP_00191
Notes	

5.1.6.1.62 SOMEIP_ETS_75: Wrong_Message_Type

Synopsys	Check that the DUT gives back an error message or ignores it when a wrong Message Type arrives in a request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SOME/IP Message with wrong Message Type using method echoUINT8 2. DUT: returns Error Message WRONG_MESSAGE_TYPE or ignores the request
Pass Criteria	DUT: returns Error Message WRONG_MESSAGE_TYPE or ignores the request
Reference	PRS_SOMEIP_00055;PRS_SOMEIP_00701
Notes	

5.1.6.1.63 SOMEIP_ETS_76: Wrong_Method_ID

Synopsys	Check that the DUT gives back an error message or ignores it when a wrong Method ID arrives in a request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: send SOME/IP Message with wrong Method ID using method echoUINT8 2. DUT: returns Error Message UNKNOWN_METHOD or ignores the request
Pass Criteria	DUT: returns Error Message UNKNOWN_METHOD or ignores the request
Reference	PRS_SOMEIP_00058;PRS_SOMEIP_00190; PRS_SOMEIP_00191;PRS_SOMEIP_00055
Notes	

5.1.6.1.64 SOMEIP_ETS_77: Wrong_Service_ID

Synopsys	Check that the DUT gives back an error message when a wrong Service ID arrives in a request or the DUT shall ignore the request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	3. TESTER: send SOME/IP Message with wrong Service ID using method echoUINT8 4. DUT: returns Error Message UNKNOWN_SERVICE or ignores the request
Pass Criteria	DUT: returns Error Message UNKNOWN_SERVICE or ignores the request
Reference	PRS_SOMEIP_00190;PRS_SOMEIP_00191
Notes	

5.1.6.1.65 SOMEIP_ETS_78: Wrong_SOMEIP_Protocol_Version

Synopsys	Check that the DUT gives back an error message when a wrong Protocol Version arrives in a request or the DUT shall ignore the request.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	5. TESTER: send SOME/IP Message with wrong Protocol Version using method echoU echoUINT8NION 6. DUT: returns Error Message WRONG_PROTOCOL_VERSIONor ignores the request
Pass Criteria	DUT: returns Error Message WRONG_PROTOCOL_VERSIONor ignores the request
Reference	PRS_SOMEIP_00052;PRS_SOMEIP_00190;PRS_SOMEIP_00191
Notes	

5.1.6.1.66 SOMEIP_ETS_81: ClientServiceActivate_Server_reboot

Synopsys	Check if the DUT detected a reboot of a server it reacts properly by re-establishing the communication.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUT's ETS client Service mode by method clientServiceActivate 2. TESTER: send OfferService to offer the Tester's ETS with a TCP Endpoint option 3. DUT: establishes a TCP-connection with the tester based on the offered end point options 4. DUT: subscribes to the event group clientServiceSubscribeEventgroup (0x0032). The end point option of this subscription shall match the parameters of the TCP-connection (IP-address and port). 5. TESTER: send SubscribeEventgroupAck to the DUT 6. TESTER: simulate a reboot by sending OfferService entries in SOME/IP-SD message with lower Session-ID as before

	7. DUT: renews its TCP-connection by closing the old one and opening a new one
Pass Criteria	DUT: establishes a TCP-connection with the tester based on the offered end point options DUT: subscribes to the event group clientServiceSubscribeEventgroup (0x0032). The end point option of this subscription shall match the parameters of the TCP-connection (IP-address and port). DUT: renews its TCP-connection by closing the old one and opening a new one
Reference	PRS_SOMEIPSD_00385
Notes	No ICMP (port unreachable) message shall be sent from the DUT.

5.1.6.1.67 SOMEIP_ETS_82: ClientServiceActivate_Server_reboot_2

Synopsys	Check if the DUT detected a reboot of a server it reacts properly by using the newly assigned UDP-Port.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUT's ETS client Service mode by method clientServiceActivate 2. TESTER: send OfferService to offer the Tester's ETS with a UDP Endpoint option 3. DUT: subscribes to the event group clientServiceSubscribeEventgroup (0x0032) 4. TESTER: send SubscribeEventgroupAck to the DUT 5. TESTER: send a SOME/IP Event to the DUT 6. TESTER: simulate a reboot by sending OfferService entries in SOME/IP-SD message with lower Session-ID as before 7. TESTER: send OfferService to offer the Tester's ETS again with another UDP-port in the endpoint option 8. DUT: subscribes again to the event group clientServiceSubscribeEventgroup (0x0032) 9. TESTER: send SubscribeEventgroupAck to the DUT 10. TESTER: send a SOME/IP Event to the DUT from the new UDP-port

Pass Criteria	DUT: subscribes to the event group clientServiceSubscribeEventgroup (0x0032) DUT: subscribes again to the event group clientServiceSubscribeEventgroup (0x0032)
Reference	PRS_SOMEIPSD_00385
Notes	No ICMP (port unreachable) message shall be sent from the DUT.

5.1.6.1.68 SOMEIP_ETS_84: ClientServiceDeactivate

Synopsys	The DUT is expected to stop the Client Mode, halting all reactions to clientService related Requests.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUT's ETS client Service mode by method clientServiceActivate with a timeout value of 1 Second 2. DUT: starts the client Start-up Phase 3. TESTER: trigger the DUT to subscribe to the Tester's Eventgroup using method clientServiceSubscribeEventgroup 4. TESTER: send OfferService message with the eventgroup parameterized in step 3 5. DUT: sends SubscribeEventgroup 6. TESTER: deactivate the DUT's ETS client Service mode by method clientServiceDeactivate 7. DUT: sends StopSubscribeEventgroup to cancel the subscription of step 5
Pass Criteria	DUT: starts the client Start-up Phase DUT: sends SubscribeEventgroup DUT: sends StopSubscribeEventgroup to cacnel the subscription of step 5
Reference	PRS_SOMEIPSD_00386;PRS_SOMEIP_00710; PRS_SOMEIP_00708;PRS_SOMEIPSD_00351
Notes	

5.1.6.1.69 SOMEIP_ETS_85: Eventgroup_EventsAndFieldsAll_2

Synopsys	The DUT has to send out SubscribeEventgroupAck and all initial Fields regarding the Eventgroup 0x02.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message for Eventgroup 0x02 2. DUT: sends SubscribeEventgroupAck 3. DUT: sends all initial Events to the same IP and Port the Tester indicated in the SubscribeEventgroup Message: • Interface Version Method-ID: 0x8005 • TestfieldUINT8 Method-ID: 0x8006 • TestfieldUINT8Array Method-ID: 0x8007 • TestfieldUINT8E2E Method-ID: 0x8009.
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends all initial Events to the same IP and Port the Tester indicated in the SubscribeEventgroup Message
Reference	PRS_SOMEIPSD_00386;PRS_SOMEIPSD_00387; PRS_SOMEIPSD_00391
Notes	

5.1.6.1.70 SOMEIP_ETS_86: Eventgroup_EventsAndFieldsAll_2_TCP

Synopsys	The DUT has to send out SubscribeEventgroupAck and all initial Fields regarding the Eventgroup 0x02.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: send SubscribeEventgroup Message for Eventgroup 0x02 2. DUT: sends SubscribeEventgroupAck 3. DUT: sends all initial Events to the same IP and Port the Tester indicated in the SubscribeEventgroup Message: • Interface Version Method-ID: 0x8005 • TestfieldUINT8 Method-ID: 0x8006 • TestfieldUINT8Array Method-ID: 0x8007 • TestfieldUINT8E2E Method-ID: 0x8009.
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends all initial Events to the same IP and Port the Tester indicated in the SubscribeEventgroup Message
Reference	PRS_SOMEIPSD_00386 PRS_SOMEIPSD_00387;PRS_SOMEIPSD_00391; PRS_SOMEIPSD_00391
Notes	

5.1.6.1.71 SOMEIP_ETS_87: Eventgroup_EventsAndFieldsUnreliable_5

Synopsys	The DUT has to send out SubscribeEventgroupAck and all initial Fields regarding the Eventgroup 0x05.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message for Eventgroup 0x05 2. DUT: sends SubscribeEventgroupAck 3. DUT: sends all initial Events to the same IP and Port the Tester indicated in the SubscribeEventgroup Message: • Interface Version Method-ID: 0x8005 • TestfieldUINT8 Method-ID: 0x8006 • TestfieldUINT8Array Method-ID: 0x8007 • TestfieldUINT8E2E Method-ID: 0x8009.
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends all initial Events to the same IP and Port the Tester indicated in the

	SubscribeEventgroup Message
Reference	PRS_SOMEIPSD_00386;PRS_SOMEIPSD_00387;PRS_SOMEIPSD_00391
Notes	

5.1.6.1.72 SOMEIP_ETS_88: SD_Answer_multiple_subscribes_together

Synopsys	The DUT has to accept each SubscribeEventgroup Entry it receives.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send a message with multiple SubscribeEventgroup Entries for the Eventgroups: • 0x02 • 0x05 • 0x06 2. DUT: sends SubscribeEventgroupAck for each entry
Pass Criteria	DUT: sends SubscribeEventgroupAck for each entry
Reference	PRS_SOMEIPSD_00263;SIP_SD_65;SIP_SD_836
Notes	

5.1.6.1.73 SOMEIP_ETS_89: SD_Calling_same_ports_before_and_after_suspendInterface

Synopsys	A certain amount of method calls are made using specific source ports and all against the same SOME/IP port offered by the DUT. Once the tester has finished one request, it will perform a new request against the DUT's SOME/IP port and therefore the DUT must be able to resume listening and dispatching requests on that port as soon as a previous request has finished.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: First TesFieldUINT8 Getter and Setter 2. DUT: responds normally 3. TESTER: request Interface Version 4. DUT: tells his Interface Version through another Port 5. TESTER: Second TestFieldUINT8 Getter and Setter 6. DUT: Similar to the first Try but using different Ports. 7. TESTER: First TestFieldUINT8Array 8. DUT: The DUT has to answer the Getter and Setter on the given Port. 9. TESTER: SuspendInterface 10. DUT: suspends the Interface for the given ammount of Time and sends a StopOfferService Message. 11. TESTER: Third TestFieldUINT8 12. DUT: has to answer again using the same Ports, thus this must still be available. 13. TESTER: Second call to InterfaceVersion 14. DUT: has to be able to answer to the same Ports it did before the SuspendInterface. 15. TESTER: Fourth TestFieldUINT8 16. DUT: The same Ports must be open. 17. TESTER: Second TestFieldUINT8Array 18. DUT: must have available the same Ports it had for the first call.
Pass Criteria	
Reference	PRS_SOMEIPSD_00356;PRS_SOMEIPSD_00364; PRS_SOMEIPSD_00357
Notes	

5.1.6.1.74 SOMEIP_ETS_91: SD_Check_OfferService_Request_ID_incrementation

Synopsys	The DUT's regular OfferService Messages have to increment the Session-ID from one iteration to the next one.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1

Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: observe the DUTs OfferService Messages and analyze the Session-IDs 2. DUT: increments the Session-IDs of its OfferService Messages
Pass Criteria	DUT: increments the Session-IDs of its OfferService Messages
Reference	PRS_SOMEIPSD_00157; PRS_SOMEIPSD_00355; PRS_SOMEIPSD_00154
Notes	

5.1.6.1.75 SOMEIP_ETS_92: SD_Check_Reaction_to_a_Subscribe_with_ttl_0

Synopsys	The DUT shall not respond to a unicast SubscribeEventgroup Request with TTL = 0 (i.e. StopSubscribeEventgroup)
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	3. TESTER: `send a unicast SubscribeEventgroup message with a TTL set to 0` 4. DUT: does not react to the request
Pass Criteria	DUT: does not react to the request
Reference	PRS_SOMEIPSD_00386;PRS_SOMEIPSD_00387;PRS_SOMEIPSD_00391
Notes	

5.1.6.1.76 SOMEIP_ETS_93: SD_Check_Reboot_Detection_separate_multicast_and_unicast

Synopsys	The DUT has to detect that the Client performed a reboot when sending Multicast and

	Unicast.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	Test description:A: Multicast - Test 1. TESTER: `send FindService (0x0101) with Session-ID: 0x0004` 2. DUT: can answer with a unicast or multicast offer service message 3. TESTER: subscribes to the Eventgroup 0x5 of the offered ETS by the DUT with Session-ID: 0x0004. 4. DUT: shall send a SubscribeEventgroupAck to the tester. After the Acknowledge the DUT shall immediately send an initial event (Notify of Field InterfaceVersion: 0x8005) due to the new subscription 5. TESTER: send a second FindService (0x0101) with Session-ID: 0x0005. (1 second after step 1) 6. DUT: can answer with a unicast or multicast offer service message 7. TESTER: subscribe to the Eventgroup 0x5 of the offered ETS by the DUT with Session-ID: 0x0005. 8. DUT: shall send a SubscribeEventgroupAck to the tester. After the second subscription the DUT is not allowed to send the initial event again 9. TESTER: send a third FindService (0x0101) with the Session-ID 0x0001 (Client reboot simulation) 10. DUT: can answer with a unicast or multicast offer service message 11. TESTER: subscribe to the Eventgroup 0x5 of the offered ETS by the DuT with Session-ID: 0x0001. 12. DUT: shall send a SubscribeEventgroupAck to the tester. It is expected that the DUT sends the initial event again, which means that it has detected the client reboot B: Unicast - Test 13. TESTER: subscribe to the Eventgroup 0x5 of the offered ETS by the DuT with Session-ID: 0x0004. 14. DUT: shall send a SubscribeEventgroupAck to the tester. After the Acknowledge the DUT shall immediately send an initial event (Notify of Field InterfaceVersion: 0x8005) due to the new subscription. 15. TESTER: sends second Subscribe to the Eventgroup 0x5 with a higher Session-ID (0x0005). 16. DUT: shall send a SubscribeEventgroupAck to the tester. After the second subscription the DUT is not allowed to send the initial event again.

	17. TESTER: send a third Subscribe to the Eventgroup 0x5 with the Session-ID 0x0001 (Client reboot simulation). 18. DUT: shall send a SubscribeEventgroupAck to the tester. It is expected that the DUT sends the initial event again, which means that it has detected the client reboot. C: Unicast and Multicast - Robustness test 19. Finally, in order to see if the DUT is able to keep the reboot detection separated for multicast (M) and unicast (U) Messages, both types of Messages are cross sent. There is no real reboot since the Session-IDs for each kind of them are always incremented but they are lower for unicast than for multicast. If the DUT is not able to keep them separated, it would believe that a reboot happened and the Test would fail.
Pass Criteria	See expected DUT reactions above
Reference	
Notes	

5.1.6.1.77 SOMEIP_ETS_94: SD_Check_Reboot_Detection_Server_Side

Synopsys	The DUT has to detect a reboot of the Tester.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	20. TESTER: `send a SubscribeEventgroup message for Eventgroup 0x02` 21. DUT: sends `SubscribeEventgroupAck` 22. TESTER: trigger the DUT to send Events by using the method TriggerEventUint8. 23. DUT: sends TestEventUint8 24. TESTER: send FindService message with a SessionID of 0x05 25. DUT: continues sending TestEventUint8 26. TESTER: send FindService message with a SessionID of 0x05 (to

	simulated a reboot) 27. DUT: stops sending TestEventUint8
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends TestEventUint8 DUT: continues sending TestEventUint8 DUT: stops sending TestEventUint8
Reference	PRS_SOMEIPSD_00157
Notes	

5.1.6.1.78 SOMEIP_ETS_95: SD_Check_subscribe_eventgroup_ttl_expired

Synopsys	The DUT has to detect that the Tester's Subscription has expired (ttl = 3) and must not respond to the Trigger that the Tester invokes after the TTL expired.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: `send a SubscribeEventgroup message for Eventgroup 0x02 with a ttl value of 3 seconds` 2. DUT: sends `SubscribeEventgroupAck` 3. TESTER: wait for at least 3 seconds (until the subscription has expired) 4. TESTER: trigger the DUT to send Events by using the method TriggerEventUint8. 5. DUT: does not send any TestEventUint8
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: does not send any TestEventUint8
Reference	PRS_SOMEIPSD_00386
Notes	

5.1.6.1.79 SOMEIP_ETS_96: SD_Check_TCP_Connection_before_SubscribeEventgroup

Synopsys	The DUT has to deny a SubscribeEventgroup request for Eventgroup 0x02 with a

	SubscribeEventgroupNAck in case the SubscribeEventgroup request includes a TCP Endpoint Option but a TCP connection has not been established in advance.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: `send a SubscribeEventgroup message for Eventgroup 0x02 with a` a TCP Endpoint Option (but the corresponding TCP connection is not established) 2. DUT: sends `SubscribeEventgroupNAck`
Pass Criteria	DUT: sends `SubscribeEventgroupNAck`
Reference	PRS_SOMEIPSD_00362
Notes	

5.1.6.1.80 SOMEIP_ETS_97: SD_Client_restarts_tcp_connection

Synopsys	On the DUT Client Mode is started and a SubscribeEventgroup attempt is triggered, therefor the DUT will try to establish a TCP Connection which will be refused by the Tester.The DUT has to react to this refusal by trying to re-engage the TCP Connection with a second TCP SYN Message as soon as it receives another OfferService Message.The Tester will react to this accepting the second TCP SYN and expects that this leads the DUT to finally subscribe. After all this, the client Mode is deactivated in order not to interfere with other TestCases.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate and send SubscribeEventgroup 2. DUT: trys to establish a TCP Connection 3. TESTER: refuse the TCP Connection 4. DUT: has to react to this refusal by trying to re-engage the TCP Connection

	with a second TCP SYN Message as soon as it receives another OfferService Message 5. TESTER: accept the second TCP SYN
Pass Criteria	DUT: trys to establish a TCP Connection DUT: has to react to this refusal by trying to re-engage the TCP Connection with a second TCP SYN Message as soon as it receives another OfferService Message
Reference	PRS_SOMEIPSD_00527
Notes	

5.1.6.1.81 SOMEIP_ETS_98: SD_ClientService_subscribe_without_method_call

Synopsys	The Tester activates the clientService Mode on the DUT and starts sending OfferService Messages. The DUT shall not try to subscribe with each OfferService it receives since the clientServiceSubscribeEventgroup Message that is needed for the DUT to can subscribe is not being sent by the tester. At the End of the TestCase, the ClientService Mode is deactivated to prevent it from interfering with other TestCases.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate and start sending OfferService Messages 2. DUT: shall not try to subscribe with each OfferService it receives
Pass Criteria	DUT: shall not try to subscribe with each OfferService it receives
Reference	PRS_SOMEIPSD_00386
Notes	

5.1.6.1.82 SOMEIP_ETS_99: SD_ClientServiceActivate

Synopsys	The DUT is triggered to enter the Client Mode without stopping its already running Server Mode.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate 2. DUT: starts the client Start-Up procedure and stays ready to behave as a client while it keeps running also the Server Mode 3. TESTER: deactivate the DUTs ETS client Service mode by the method clientServiceDeactivate
Pass Criteria	DUT: starts the client Start-Up procedure and stays ready to behave as a client while it keeps running also the Server Mode
Reference	PRS_SOMEIPSD_00350;PRS_SOMEIPSD_00351
Notes	

5.1.6.1.83 SOMEIP_ETS_100: SD_ClientServiceActivate_no_FindServices_in_Main_Phase

Synopsys	The clientService Mode shall only send FindService Messages during the Start-Up Phase, not in the Main Phase
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate 2. DUT: starts sending FindService messages during the Start Phase and stops sending FindService messages when entering the Main Phase 3. TESTER: deactivate the DUTs ETS client Service mode by the method

	clientServiceDeactivate
Pass Criteria	DUT: starts sending FindService messages during the Start Phase and stops sending FindService messages when entering the Main Phase
Reference	PRS_SOMEIPSD_00351
Notes	

5.1.6.1.84 SOMEIP_ETS_101: SD_ClientServiceActivate_send_StopOfferService

Synopsys	Client Mode is activated on the DUT and when the DUT starts sending FindService messages the Tester sends StopOfferService. The DUT is expected to understand that the Service and Instance-ID that it is looking for is no longer available. It should stop sending FindService Messages for this Service and Instance-ID. Find-ALL FindService Messages are still allowed.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate 2. DUT: starts sending FindService messages 3. TESTER: send StopOfferService as soon as the first FindService message from the DUT arrives at the Tester 4. DUT: stops sending FindService messages 5. TESTER: deactivate the DUTs ETS client Service mode by the method clientServiceDeactivate
Pass Criteria	DUT: starts sending FindService messages DUT: stops sending FindService messages
Reference	PRS_SOMEIPSD_00351;PRS_SOMEIPSD_00363
Notes	

5.1.6.1.85 SOMEIP_ETS_103: SD_ClientServiceGetLastValueOfEventTCP

Synopsys	The Tester activates the clietnService Mode on the DUT, checks it's Behaviour and triggers it to subscribe to it. After that, it sends it a UINT8 TCP Event, to the IP and Port the DUT

	indicated with its SubscribeEventgroup Message, valued as 0x08. This Value has to be returned by the DUT once the Tester triggers the clientServiceGetLastValueOfEventTCP Method, after this, the clientService Mode is deactivated on the DUT.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate 2. DUT: starts sending FindService messages 3. TESTER: Send OfferService Messages 4. DUT: sends SubscribeEventgroup Message 5. TESTER: send a UINT8 TCP Event, to the IP and Port the DUT indicated with its SubscribeEventgroup Message, valued as 0x08 6. TESTER: trigger the clientServiceGetLastValueOfEventTCP Method 7. DUT: return the value of step 5 8. TESTER: deactivate the DUTs ETS client Service mode by the method clientServiceDeactivate
Pass Criteria	DUT: starts sending FindService messages DUT: sends SubscribeEventgroup Message DUT: return the value of step 5
Reference	PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00362
Notes	

5.1.6.1.86 SOMEIP_ETS_104: SD_ClientServiceGetLastValueOfEventUDPMulticast

Synopsys	The Tester activates the clientService Mode on the DUT, checks it's Behaviour and triggers it to subscribe to it. After that, it sends it a UINT8 multicast Event, to the IP and Port the DUT indicated with the Tester's SubscribeEventgroupAck Message, valued as 0x08.This Value has to be returned by the DUT once the Tester triggers the clientServiceGetLastValueOfEventUDPMulticast Method, after this, the clientService Mode is deactivated on the DUT.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service

Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate 2. DUT: starts sending FindService messages 3. TESTER: Send OfferService Messages 4. DUT: sends SubscribeEventgroup Message 5. TESTER: send a UINT8 multicast Event, to the IP and Port the DUT indicated with its SubscribeEventgroup Message, valued as 0x08 6. TESTER: trigger the clientServiceGetLastValueOfEventUDPMulticast Method 7. DUT: return the value of step 5 8. TESTER: deactivate the DUTs ETS client Service mode by the method clientServiceDeactivate
Pass Criteria	DUT: starts sending FindService messages DUT: sends SubscribeEventgroup Message DUT: return the value of step 5
Reference	PRS_SOMEIPSD_00360;PRS_SOMEIPSD_00361
Notes	

5.1.6.1.87 SOMEIP_ETS_105: SD_ClientServiceGetLastValueOfEventUDPUnicast

Synopsys	The Tester activates the clietnService Mode on the DUT, checks it's Behaviour and triggers it to subscribe to it. After that, it sends it a UINT8 UDP unicast Event, to the IP and Port the DUT indicated with his SubscribeEventgroup Message, valued as 0x08. This Value has to be returned by the DUT once the Tester triggers the clientServiceGetLastValueOfEventUDPUnicast Method, after this, the clientService Mode is deactivated on the DUT.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate

Procedure	2. DUT: starts sending FindService messages 3. TESTER: Send OfferService Messages 4. DUT: sends SubscribeEventgroup Message 5. TESTER: send a UINT8 UDP Event, to the IP and Port the DUT indicated with its SubscribeEventgroup Message, valued as 0x08 6. TESTER: trigger the clientServiceGetLastValueOfEventUDPUnicast Method 7. DUT: return the value of step 5 8. TESTER: deactivate the DUTs ETS client Service mode by the method clientServiceDeactivate
Pass Criteria	DUT: starts sending FindService messages DUT: sends SubscribeEventgroup Message DUT: return the value of step 5
Reference	PRS_SOMEIPSD_00360;PRS_SOMEIPSD_00361; PRS_SOMEIPSD_00380
Notes	

5.1.6.1.88 SOMEIP_ETS_106: SD_ClientServiceSubscribeEventgroup

Synopsys	Client Mode is activated on the DUT and it has to subscribe to the Tester.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: activate the DUTs ETS client Service mode by the method clientServiceActivate 2. DUT: starts sending FindService messages 3. TESTER: Send clientServiceSubscribeEventgroup Message 4. TESTER: send OfferService Messages offering the ETS service 5. DUT: sends SubscribeEventgroup Message to subscribe to the ETS service 6. TESTER: deactivate the DUTs ETS client Service mode by the method clientServiceDeactivate
Pass Criteria	DUT: starts sending FindService messages DUT: sends SubscribeEventgroup Message to subscribe to the ETS service
Reference	PRS_SOMEIPSD_00385;PRS_SOMEIPSD_00386; PRS_SOMEIPSD_00268;PRS_SOMEIPSD_00380

Notes	

5.1.6.1.89 SOMEIP_ETS_107: SD_Consider_Entries_Order

Synopsys	The Tester initially subscribes to the DUT and then sends a Message containing two Entries: A SubscribeEventgroup with TTL 0 and normal SubscribeEventgroup. The Testcases awaits the DUT first to delete the Tester from the List of subscribed Entities and then an SubscribeEventgroupAck (in this order) for the SubscribeEventgroup, this one should be followed by the initial Fields for the subscribed Eventgroup (0x05).
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message to subscribe to Eventgroup 0x05 2. DUT: sends SubscribeEventgroupAck and the initial Events 3. TESTER: Send SD Message containing two Entries: • StopSubscribeEventgroup Entry (TTL = 0) for Eventgroup 0x05 • SubscribeEventgroup Entry for Eventgroup 0x05 (to re-subscribe) 4. DUT: sends SubscribeEventgroupAck and the initial Events again (corresponding to the re-subscription)
Pass Criteria	DUT: sends SubscribeEventgroupAck and the initial Events DUT: sends SubscribeEventgroupAck and the initial Events again (corresponding to the re-subscription)
Reference	PRS_SOMEIPSD_00263
Notes	

5.1.6.1.90 SOMEIP_ETS_108: SD_Deregister_from_Eventgroup

Synopsys	The Tester subscribes and unsubscribes. Right after having received the Initial Events corresponding to the SubscribeEventgroup the Tester tries to trigger the TestEventUINT8 and expects the DUT to ignore the Trigger since the initial Subscription is not active anymore.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message to subscribe to Eventgroup 0x05 2. DUT: sends SubscribeEventgroupAck and the initial Events 3. TESTER: Send StopSubscribeEventgroup message for Eventgroup 0x05 4. TESTER: by using the method TestEventUINT8 trigger the DUT to send a corresponding Event 5. DUT: does not send the Event as the initial subscription is not active anymore
Pass Criteria	DUT: sends SubscribeEventgroupAck and the initial Events DUT: does not send the Event as the initial subscription is not active anymore
Reference	PRS_SOMEIPSD_00388;PRS_SOMEIPSD_00389;PRS_SOMEIPSD_00386
Notes	

5.1.6.1.91 SOMEIP_ETS_109: SD_Do_not_specify_a_port

Synopsys	The Tester sends a SubscribeEventgroup Message without indicating a Port in the Endpoint Options. This should lead the DUT to reject the whole Message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message without specifying a port in the Endpoint option (port number is 0) 2. DUT: sends SubscribeEventgroupNack to reject the subscription attempt
Pass Criteria	DUT: sends SubscribeEventgroupNack to reject the subscription attempt
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00393

Notes	

5.1.6.1.92 SOMEIP_ETS_110: SD_Do_not_specify_IPv4_Adress

Synopsys	The Tester sends a SubscribeEventgroup Message which does not indicate IPv4 Addresses in the Endpoint Options (since this is not possible without malforming the message, the transmitted IP is 32.0.0.0), this should lead the DUT to reject the Message and answer with a SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message without specifying a valid IP address in the Endpoint option 2. DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Pass Criteria	DUT: sends SubscribeEventgroupNack to reject the subscription attempt
Reference	PRS_SOMEIPSD_00306;PRS_SOMEIPSD_00307; PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.93 SOMEIP_ETS_111: SD_Empty_Entries_Array

Synopsys	The Tester sends a SubscribeEventgroup Message which states a total length of Zero for the Entries Array, though it should not be interpreted and completely ignored.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: send SubscribeEventgroup Message with Entries Array length zero 2. DUT: ignores the SubscribeEventgroup message (shows no reaction)
Pass Criteria	DUT: ignores the SubscribeEventgroup message (shows no reaction)
Reference	PRS_SOMEIPSD_00540
Notes	

5.1.6.1.94 SOMEIP_ETS_112: SD_Empty_Option

Synopsys	The Tester sends SubscribeEventgroup with IPv4Option length = 0. The DUT shall respond SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message for DefaultEventgroup with options array length = 0 2. DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Reference	PRS_SOMEIPSD_00307
Notes	

5.1.6.1.95 SOMEIP_ETS_113: SD_Empty_Options_Array

Synopsys	The Tester sends SubscribeEventgroup with IPv4Option length = 0. The DUT shall respond SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1

Test Input Parameters	Check section general Input Parameters
Test Procedure	3. TESTER: send SubscribeEventgroup message for DefaultEventgroup with options array length = 0 4. DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Reference	PRS_SOMEIPSD_00265;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.96 SOMEIP_ETS_114: SD_Entries_Length_wrong_combined

Synopsys	The Tester sends SubscribeEventgroup with two correct Entries but a shortened Entries Array length. The DUT shall respond SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message for DefaultEventgroup with two correct Entries but a shortened Entries Array length 2. DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Reference	PRS_SOMEIPSD_00265;PRS_SOMEIPSD_00264;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.97 SOMEIP_ETS_115: SD_Entry_references_more_options_than_exist

Synopsys	The Tester sends SubscribeEventgroup with more option references than existing options. The DUT shall respond SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service

Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message for DefaultEventgroup with more option references than existing options 2. DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Reference	PRS_SOMEIPSD_00393;PRS_SOMEIPSD_00566
Notes	

5.1.6.1.98 SOMEIP_ETS_116: SD_Entry_references_non_existing_option_type

Synopsys	The Tester sends SubscribeEventgroup with unknown Option type (0x77). The DUT shall respond SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message for DefaultEventgroup with unknown Option type (0x77). 2. DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription attempt
Reference	PRS_SOMEIPSD_00305;PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.99 SOMEIP_ETS_117: SD_Entry_references_options_of_same_kind

Synopsys	The Tester sends SubscribeEventgroup with two options of the same type. The DUT shall respond SubscribeEventgroupNAck or ignore the request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message for DefaultEventgroup with two options of the same type 2. DUT: may send SubscribeEventgroupNAck to reject the subscription attempt or may ignore the request
Pass Criteria	DUT: may send SubscribeEventgroupNAck to reject the subscription attempt or may ignore the request
Reference	PRS_SOMEIPSD_00393
Notes	

5.1.6.1.100 SOMEIP_ETS_118: SD_Ignore_Options_in_FindService

Synopsys	The Tester sends 10 FindService messages with IPv4 Endpoint Option. The DUT shall ignore the options and respond with at least one unicast OfferService message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send 10 FindService messages with IPv4 Endpoint Option in 100ms intervals 2. DUT: sends at least one unicast OfferService message
Pass Criteria	DUT: sends at least one unicast OfferService message
Reference	PRS_SOMEIPSD_00268;SIP_SD_877;SIP_SD_878
Notes	

5.1.6.1.101 SOMEIP_ETS_119: SD_Indicate_wrong_l4proto_param

Synopsys	The SubscribeEventgroup Message sent by the Tester refers to an IPv4Endpoint Option with a wrong l4proto Parameter (neither UDP or TCP), since this way not all needed Endpoint Options are present in a correct way the Message has to be rejected with a SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message referencing an IPv4Endpoint Option with a wrong l4proto Parameter 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends at least one unicast OfferService message
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.102 SOMEIP_ETS_120: SD_Initial_Events_after_Subscribe_with_alternate_IPs

Synopsys	The DUT has to answer the Testers SubscribeEvengroup Message with a SubscribeEventgroupAck and immediatelly after send out the corresponding Initial Fields for the Subscribed Eventgroup to the destination stated in the SubscribeEventgroup Endpoint Option.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message referencing an IPv4Endpoint Option which differs from the Testers IP and Port. 2. DUT: sends SubscribeEventgroupAck to the DUT's IP address and Port 3. DUT: sends the initial fields to the IP address and Port given in the Endpoint

	option of the request
Pass Criteria	DUT: sends SubscribeEventgroupAck to the DUT's IP address and Port DUT: sends the initial fields to the IP address and Port given in the Endpoint option of the request
Reference	PRS_SOMEIPSD_00386;PRS_SOMEIPSD_00387; PRS_SOMEIPSD_00391;PRS_SOMEIPSD_00391
Notes	

5.1.6.1.103 SOMEIP_ETS_121: SD_Initial_Events_after_SubscribeEventgroup

Synopsys	The DUT has to answer the Testers SubscribeEvengroup Message with a SubscribeEventgroupAck and immediatelly after send out the corresponding Initial Fields for the Subscribed Eventgroup
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message for Eventgroup 0x05. 2. DUT: sends SubscribeEventgroupAck 3. DUT: sends the initial fields
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends the initial fields
Reference	PRS_SOMEIPSD_00386;PRS_SOMEIPSD_00391;PRS_SOMEIPSD_00391; PRS_SOMEIPSD_00310;PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00360; PRS_SOMEIPSD_00361;PRS_SOMEIPSD_00362
Notes	

5.1.6.1.104 SOMEIP_ETS_122: SD_Interface_Version

Synopsys	The DUT has to answer indicating its Interface Version in the correct format.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: trigger the Interface Version Getter method 2. DUT: returns the method response
Pass Criteria	DUT: returns the method response
Reference	PRS_SOMEIPSD_00357; PRS_SOMEIPSD_00360;PRS_SOMEIPSD_00361
Notes	

5.1.6.1.105 SOMEIP_ETS_123: SD_Length_of_Entry_Array_longer_than_message_allows

Synopsys	The Tester sends a SubscribeEventgroup Message whos Entry Array Length exceeds the Message total Length and so the Message should be unrecognizable for the DUT or at least too damaged to be answered with a SubscribeEventgroupAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message with an Entry Array length exceeding the total Length of the message 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00265;PRS_SOMEIPSD_00153;PRS_SOMEIPSD_00270
Notes	

5.1.6.1.106 SOMEIP_ETS_124: SD_Length_of_Entry_Array_too_long

Synopsys	The Message's Entry Array Length does not exceed the Message total Length but goes far beyond its normal Limit and so the Message turns defect and must be rejected by the DUT.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	3. TESTER: send SubscribeEventgroup message with an Entry Array length exceeding the total Length of the message by at least 20 Bytes 4. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00265;PRS_SOMEIPSD_00393; PRS_SOMEIPSD_00270;PRS_SOMEIPSD_00264; PRS_SOMEIPSD_00566
Notes	

5.1.6.1.107 SOMEIP_ETS_125: SD_Length_of_Entry_Array_too_short

Synopsys	The Tester sends a SubscribeEventgroup Message with an Entry Array Length shorter than it should be based on the Length indicated by each Option.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup message with an Entry Array Length less than the sum of the Lengths indicated by each Option 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00265;PRS_SOMEIPSD_00270
Notes	

5.1.6.1.108 SOMEIP_ETS_127: SD_Multicast_FindService

Synopsys	The DUT receives 10 Multicast FindService Messages every 100ms asking for a valid Service/Instance-ID (depending on the DUT). The DUT has to answer using a Unicast OfferService for this Service/Instance-ID.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send 10 Multicast FindService messages in an interval of 100ms requesting a valid Service/Instance-ID (depending on the DUT). 2. DUT: responses with at least one unicast OfferService message
Pass Criteria	DUT: responses with at least one unicast OfferService message
Reference	PRS_SOMEIPSD_00305;PRS_SOMEIPSD_00306; PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00261
Notes	

5.1.6.1.109 SOMEIP_ETS_128: SD_Multicast_FindService_Major_Minor_Version_set_to_all

Synopsys	The DUT receives both 10 multicast FindService Requests with Major Version and later with a Minor Version set to 0xFF (all of them in 100ms intervals) and is expected to answer each one of them with at least one unicast OfferService Message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send 10 Multicast FindService messages in an interval of 100ms with Major Version set to 0xFF (Minor Version set to Default value) 2. DUT: responses with at least one unicast OfferService message 3. TESTER: send 10 Multicast FindService messages in an interval of 100ms with Minor Version set to 0xFFFFFFFF (Major Version set to Default value) 4. DUT: responses with at least one unicast OfferService message

Pass Criteria	DUT: responses with at least one unicast OfferService message DUT: responses with at least one unicast OfferService message
Reference	PRS_SOMEIPSD_00268;PRS_SOMEIPSD_00305; PRS_SOMEIPSD_00306;PRS_SOMEIPSD_00307; PRS_SOMEIPSD_00351;PRS_SOMEIPSD_00351
Notes	

5.1.6.1.110 SOMEIP_ETS_130: SD_Multicast_FindService_with_unicast_Flag_to_0

Synopsys	The DUT receives a multicast FindService Request with its Unicast Flag set to 0 and is expected to ignore this flag and to answer with a unicast OfferService Message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send a Multicast FindService message with Unicast Flag set to 0 2. DUT: sends a unicast OfferService Message
Pass Criteria	DUT: sends a unicast OfferService Message
Reference	PRS_SOMEIPSD_00268;PRS_SOMEIPSD_00305; PRS_SOMEIPSD_00306;PRS_SOMEIPSD_00307
Notes	

5.1.6.1.111 SOMEIP_ETS_134: SD_Option_Length_ends_past_Options_Array_Var_A

Synopsys	The Tester sends a SubscribeEventgroup Message whose Option's Length surpasses the total Length indicated for the options Array and so the Message must be answered with an SubscribeEventgroupNAck or not answered at all. In Variant A of this testCase, the total length indicated in the SOME/IP Header is slightly shortened from 60 to 48 Bytes in order to cut the Options Array and make this one look shorter than it really is. The Length of the Option Array is cut in the same order as in the SOME/IP Header (lossing also 12 Bytes, from 24 to 12), only the Option's individual Length fields are correct.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with an actual Option Length greater than the indicated options Array length. 2. TESTER: sends SubscribeEventgroupNAck to reject the subscription request or ignores the request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request or ignores the request
Reference	PRS_SOMEIPSD_00274;PRS_SOMEIPSD_00393;PRS_SOMEIP_00042
Notes	

5.1.6.1.112 SOMEIP_ETS_135: SD_Option_Length_ends_past_Options_Array_Var_B

Synopsys	The Tester sends a SubscribeEventgroup Message whose Options has a Length which surpasses the Length indicated for the Options Array and so the Message must be answered with an SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with an actual Option Length greater than the indicated options Array length. 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request or ignores the request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request or ignores the request
Reference	PRS_SOMEIPSD_00274;PRS_SOMEIPSD_00393;PRS_SOMEIP_00042
Notes	

5.1.6.1.113 SOMEIP_ETS_136: SD_Option_Length_shorter_GT_0_as_specified_for_type

Synopsys	The Tester sends a SubscribeEventgroup Message with a UDP Option Length less than specified for the Type (4 Bytes instead of 9) .The Message must be answered with an SubscribeEventgroupNAck or be fully ignored.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with UDP Option Length less than specified for the Type (4 Bytes instead of 9) 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request or ignores the request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request or ignores the request
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.114 SOMEIP_ETS_137: SD_Option_shorter_with_unaligned_next_option

Synopsys	The Tester sends a SubscribeEventgroup Message which first Option has a Length of 14 Bytes and the second one of 4 Bytes, both are wrong sized but the total expected Length for the Options Array is appearantly kept since the missing bytes are deleted from the UDP option and added as dummy bytes at the end of the TCP option. The total SOME/IP length is kept at 60 and the missing bytes from the options array are replenished with dummy bytes. The DUT has to answer this Message with a SubscribeEventgroupNAck, since this change in the Option Sizes disrupts them and should turn them useless.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	1. TESTER: send SubscribeEventgroup Message with • TCP endpoint option length = 14 bytes (append dummy bytes) • UDP endpoint option length = 4 bytes (cut to 4 bytes) 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00393; PRS_SOMEIP_00042;PRS_SOMEIPSD_00265PRS_SOMEIPSD_00274
Notes	

5.1.6.1.115 SOMEIP_ETS_138: SD_Options_Array_longer_than_message_allows

Synopsys	The Tester sends a SubscribeEventgroup Message which stated Options Array Length is longer than the Message itself (Option Array Length is 0x28 instead of 0x18). The DUT shall return SubscribeEventgroupAck. Due to AUTOSAR compatibility purposes it is also allowed to ignore the message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • Options Array Length > actual length 2. DUT: sends SubscribeEventgroupAck message or ignore the reuqest
Pass Criteria	DUT: sends SubscribeEventgroupAck message or ignore the reuqest
Reference	PRS_SOMEIPSD_00390
Notes	

5.1.6.1.116 SOMEIP_ETS_139: SD_Options_Array_too_short

Synopsys	The Tester sends a SubscribeEventgroup Message which Options Array Length is shorter than required (2 Bytes instead of the original 24). The Message must be answered with

	SubscribeEventgroupNAck since the required Options for this Entry are not accesible anymore.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • Options Array Length < actual length 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00265;PRS_SOMEIPSD_00270;PRS_SOMEIPSD_00566
Notes	

5.1.6.1.117 SOMEIP_ETS_140: SD_Request_non_existing_EventgroupID

Synopsys	The Tester sends a SubscribeEventgroup Message requesting a non existing Eventgroup-ID. The Message must be answered with SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • non existing Eventgroup-ID 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00394;PRS_SOMEIPSD_00393;PRS_SOMEIPSD_00566
Notes	

5.1.6.1.118 SOMEIP_ETS_141: SD_Request_non_existing_InstanceID

Synopsys	The Tester sends a SubscribeEventgroup Message requesting a non existing Instance-ID. The Message must be answered with an SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • non existing Instance-ID 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00394;PRS_SOMEIPSD_00393;PRS_SOMEIPSD_00566
Notes	

5.1.6.1.119 SOMEIP_ETS_142: SD_Request_non_existing_Major_Version

Synopsys	The Tester sends a SubscribeEventgroup Message requesting a non existing Major Version. The Message must be answered with an SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • non existing Major Version 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00394;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.120 SOMEIP_ETS_143: SD_Request_non_existing_ServiceID

Synopsys	The Tester sends a SubscribeEventgroup Message requesting a non existing ServiceID. The Message must be answered with an SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • non existing ServiceID 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00386;PRS_SOMEIPSD_00394;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.121 SOMEIP_ETS_144: SD_Reserved_Field_Endpoint_Option_set

Synopsys	The Tester sends a SubscribeEventgroup Message with the reserved Fields of the Endpoint Options set. The Message must be answered with SubscribeEventgroupAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • reserved Fields of the Endpoint Options set 2. DUT: ignores the reserved fields and sends SubscribeEventgroupAck
Pass Criteria	DUT: ignores the reserved fields and sends SubscribeEventgroupAck
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00391
Notes	

5.1.6.1.122 SOMEIP_ETS_146: SD_ResetInterface

Synopsys	The Tester gets the TestFieldUINT8 Value hold by the DUT and stores it, later it sets a new, different, one. For both Requests the Tester expects an answer which is checked against the SOME/IP Specifications. Once this Preparations are taken, the Tester triggers the DUT to reset and after waiting 3 Seconds for the Reset to complete, it asks again for the TestFieldUINT8 Value, expecting it to be at least different from the one he set before the Reset.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: get the TestFieldUINT8 Value hold by the DUT and store it 2. DUT: returns the value 3. TESTER: set a new different value 4. DUT: returns the value 5. TESTER: trigger the DUT to reset 6. TESTER: ask again for the TestFieldUINT8 Value 7. DUT: returns the value which shall be at least different from the one he set before the Reset
Pass Criteria	DUT: returns the value DUT: returns the value DUT: returns the value which shall be at least different from the one he set before the Reset
Reference	PRS_SOMEIPSD_00356;PRS_SOMEIP_00170
Notes	

5.1.6.1.123 SOMEIP_ETS_147: SD_Send_triggerEventUINT8_Eventgroup_2

Synopsys	The Tester subscribes to an Eventgroup and triggers the DUT to send TestEventUINT8 events afterwards. The DUT shall acknowledge the subscription and send the field events to the IP address und port indicated by the Endpoint Option in the subscription request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters

Test Procedure	8. TESTER: send SubscribeEventgroup Message for Eventgroup 0x02 9. DUT: sends SubscribeEventgroupAck 10. TESTER: trigger the method triggerEventUINT8 11. DUT: sends the field TestEventUINT8 to the IP and port indicated in the Endpoint Option of the request
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends the field TestEventUINT8 to the IP and port indicated in the Endpoint Option of the request
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00310; PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00360; PRS_SOMEIPSD_00361
Notes	

5.1.6.1.124 SOMEIP_ETS_148: SD_Send_triggerEventUINT8Array_Eventgroup_2

Synopsys	The Tester subscribes to an Eventgroup and triggers the DUT to send TestEventUINT8Array events afterwards. The DUT shall acknowledge the subscription and send the field events to the IP address und port indicated by the Endpoint Option in the subscription request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message for Eventgroup 0x02 2. DUT: sends SubscribeEventgroupAck 3. TESTER: trigger the method triggerEventUINT8Array 4. DUT: sends the field TestEventUINT8Array to the IP and port indicated in the Endpoint Option of the request
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends the field TestEventUINT8Array to the IP and port indicated in the Endpoint Option of the request
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00310;PRS_SOMEIPSD_00380
Notes	

5.1.6.1.125 SOMEIP_ETS_149: SD_Send_triggerEventUINT8E2E_Eventgroup_2

Synopsys	The Tester subscribes to an Eventgroup and triggers the DUT to send TestEventUINT8E2E events afterwards. The DUT shall acknowledge the subscription and send the field events to the IP address und port indicated by the Endpoint Option in the subscription request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message for Eventgroup 0x02 2. DUT: sends SubscribeEventgroupAck 3. TESTER: trigger the method triggerEventUINT8E2E 4. DUT: sends the field TestEventUINT8E2E to the IP and port indicated in the Endpoint Option of the request
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends the field TestEventUINT8E2E to the IP and port indicated in the Endpoint Option of the request
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00310;PRS_SOMEIPSD_00380
Notes	

5.1.6.1.126 SOMEIP_ETS_150: SD_Send_triggerEventUINT8Multicast_Eventgroup_6

Synopsys	The Tester subscribes to an Eventgroup and triggers the DUT to send TestEventUINT8Multicast events afterwards. The DUT shall acknowledge the subscription and send the field events to the IP address und port indicated by the Endpoint Option in the subscription request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message for Eventgroup 0x06 2. DUT: sends SubscribeEventgroupAck 3. TESTER: trigger the method trigger triggerUINT8Multicast 4. DUT: sends the field TestEventUINT8Multicast to the IP and port indicated in the Endpoint Option of the request

Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends the field TestEventUINT8Multicast to the IP and port indicated in the Endpoint Option of the request
Reference	TBD
Notes	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00310; PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00323; PRS_SOMEIPSD_00324;PRS_SOMEIPSD_00325; PRS_SOMEIPSD_00326;PRS_SOMEIPSD_0039

5.1.6.1.127 SOMEIP_ETS_151: SD_Send_triggerEventUINT8Reliable_Eventgroup_2

Synopsys	The Tester subscribes to an Eventgroup and triggers the DUT to send TestEventUINT8Reliable events afterwards. The DUT shall acknowledge the subscription and send the field events to the IP address und port indicated by the Endpoint Option in the subscription request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message for Eventgroup 0x02 2. DUT: sends SubscribeEventgroupAck 3. TESTER: trigger the method triggerEventUINT8Reliable 4. DUT: sends the field TestEventUINT8Reliable to the IP and port indicated in the Endpoint Option of the request
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends the field TestEventUINT8Reliable to the IP and port indicated in the Endpoint Option of the request
Reference	PRS_SOMEIPSD_00307;PRS_SOMEIPSD_00310; PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00362
Notes	

5.1.6.1.128 SOMEIP_ETS_152: SD_Session_ID_is_one_after_wrapping

Synopsys	The Tester sends lots of FindService Messages in order to observe the session id incrementation and wrap-around of the OfferService responses. After wrap-around the session id has to start from 1.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send lots of FindService Messages 2. DUT: responses with OfferService messages and starts with session id 1 after wrap-around
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: responses with OfferService messages and starts with session id 1 after wrap-around
Reference	PRS_SOMEIPSD_00159
Notes	

5.1.6.1.129 SOMEIP_ETS_153: SD_SOMEIP_Length_shorter_as_expected

Synopsys	The Tester subscribes to an Eventgroup with wrong parameters in the request: SOME/IP Length is less than the actual length. The DUT shall reject the request with SubscribeEventgroupNAck or may ignore the request.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • SOME/IP Length is less than the actual length 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request or ignores the request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request or ignores the request
Reference	PRS_SOMEIP_00042;PRS_SOMEIPSD_00393;PRS_SOMEIPSD_00566
Notes	

5.1.6.1.130 SOMEIP_ETS_154: SD_Specify_an_unexisting_IPv4_Address

Synopsys	The Tester subscribes to an Eventgroup with wrong parameters in the request: Invalid IPv4 Address in the EndpointOption. The DUT shall reject the request with SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • Invalid IPv4 Address in the EndpointOption 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00306;PRS_SOMEIPSD_00307; PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.131 SOMEIP_ETS_155: SD_Subscribe_after_StopSubscribe

Synopsys	The Tester subscribes to an Eventgroup, unsubscribes und subscribes again. The DUT shall confirm the first subscription, accept the unsubscribe message and react correctly to the re-subscription.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message 2. DUT: sends SubscribeEventgroupAck 3. TESTER: send StopSubscribeEventgroup Message 4. DUT: sends SubscribeEventgroupAck and sends all releated initial fields
Pass Criteria	DUT: sends SubscribeEventgroupAck DUT: sends SubscribeEventgroupAck and sends all releated initial fields
Reference	PRS_SOMEIPSD_00263;PRS_SOMEIPSD_00386

Notes	

5.1.6.1.132 SOMEIP_ETS_162: SD_SubscribeEventgroup_with_unallowed_option_ip

Synopsys	The Tester subscribes to an Eventgroup with wrong parameters in the request: Invalid IPv4 Address in the EndpointOption (IP address = IP address of the DUT). The DUT shall reject the request with SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send SubscribeEventgroup Message with • Invalid IPv4 Address in the EndpointOption 2. DUT: sends SubscribeEventgroupNAck to reject the subscription request
Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00306;PRS_SOMEIPSD_00307; PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00393; PRS_SOMEIPSD_00566
Notes	

5.1.6.1.133 SOMEIP_ETS_164: SD_SubscribeEventgroup_with_unallowed_option_ip_2

Synopsys	The Tester subscribes to an Eventgroup with wrong parameters in the request: Invalid IPv4 Address in the EndpointOption (IP address = 111.111.111.111). The DUT shall reject the request with SubscribeEventgroupNAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	3. TESTER: send SubscribeEventgroup Message with • Invalid IPv4 Address in the EndpointOption 4. DUT: sends SubscribeEventgroupNAck to reject the subscription request

Pass Criteria	DUT: sends SubscribeEventgroupNAck to reject the subscription request
Reference	PRS_SOMEIPSD_00306;PRS_SOMEIPSD_00307; PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00393;PRS_SOMEIPSD_00566
Notes	

5.1.6.1.134 SOMEIP_ETS_163: SD_SuspendInterface

Synopsys	The Tester retrieves the TestFieldUINT8 Value the DUT has before the Suspension and sets a new one, for both Requests it expects a valid Answer. After that, the SuspendInterface Request is sent to the DUT and two conditions are expected: - First, during the suspension time, which must least as long as the suspendInterface Message stated in his Options, the DUT should not send out any kind of Message. It should notify this by sending out a StopOfferService Message. - Second, after the Suspension finished, the Getter is called again and the Value returned must not be the same as set by the Tester before the Suspension since after the suspension Time, the Interface is expected not to reset.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: trigger the DUT to send the field TestFieldUINT8 by using the method Getter 2. DUT: returns the field TestFieldUINT8 3. TESTER: set a new value for the field TestFieldUINT8 by using the method Setter 4. DUT: returns the field TestFieldUINT8 with the value set by the Tester in step 3 5. TESTER: send the SuspendInterface Request 6. DUT: should not send out any kind of Message during the suspension time. It should notify this by sending out a StopOfferService Message. 7. TESTER: after the Suspension finished call Getter again 8. DUT: returns the value. Value returned must not be the same as set by the Tester before the Suspension
Pass Criteria	DUT: returns the field TestFieldUINT8 DUT: returns the field TestFieldUINT8 with the value set by the Tester in step 3 DUT: should not send out any kind of Message during the suspension time. It should notify this by sending out a StopOfferService Message. DUT: returns the value. Value returned must not be the same as set by the Tester before the Suspension

Reference	PRS_SOMEIPSD_00356;PRS_SOMEIPSD_00364;SIP_SD_811;PRS_SOMEIPSD_00363
Notes	

5.1.6.1.135 SOMEIP_ETS_166: SD_TestFieldUINT8

Synopsys	The Tester triggers the DUT to send the field TestFieldUINT8 by using the methods Getter and Setter. The DUT shall respond correctly.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: trigger the DUT to send the field TestFieldUINT8 by using the method Getter 2. DUT: returns the field TestFieldUINT8 3. TESTER: set a new value for the field TestFieldUINT8 by using the method Setter 4. DUT: returns the field TestFieldUINT8 with the value set by the Tester in step 3
Pass Criteria	DUT: returns the field TestFieldUINT8 DUT: returns the field TestFieldUINT8 with the value set by the Tester in step 3
Reference	PRS_SOMEIPSD_00357;PRS_SOMEIPSD_00360; PRS_SOMEIPSD_00361;PRS_SOMEIP_00180
Notes	

5.1.6.1.136 SOMEIP_ETS_167: SD_TestFieldUINT8Array

Synopsys	The Tester triggers the DUT to send the field TestFieldUINT8Array by using the methods Getter and Setter. The DUT shall respond correctly.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: trigger the DUT to send the field TestFieldUINT8Array by using the method Getter

	2. DUT: returns the field TestFieldUINT8Array 3. TESTER: set a new value for the field TestFieldUINT8Array by using the method Setter 4. DUT: returns the field TestFieldUINT8Array with the value set by the Tester in step 3
Pass Criteria	DUT: returns the field TestFieldUINT8Array DUT: returns the field TestFieldUINT8Array with the value set by the Tester in step 3
Reference	PRS_SOMEIPSD_00357
Notes	

5.1.6.1.137 SOMEIP_ETS_168: SD_TestFieldUINT8Reliable

Synopsys	The Tester triggers the DUT to send the field TestFieldUINT8Reliable by using the methods Getter and Setter. The DUT shall respond correctly.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: trigger the DUT to send the field TestFieldUINT8Reliable by using the method Getter 2. DUT: returns the field TestFieldUINT8Reliable 3. TESTER: set a new value for the field TestFieldUINT8Reliable by using the method Setter 4. DUT: returns the field TestFieldUINT8Reliable with the value set by the Tester in step 3
Pass Criteria	DUT: returns the field TestFieldUINT8Reliable DUT: returns the field TestFieldUINT8Reliable with the value set by the Tester in step 3
Reference	PRS_SOMEIPSD_00362
Notes	

5.1.6.1.138 SOMEIP_ETS_171: SD_ Unicast_FindService

Synopsys	The Tester requests a service the DUT offers by sending several unicast FindService messages. The DUT shall respond with at least one unicast OfferService message.

Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send several unicast FindService messages to request a service the DUT offers 2. DUT: sends at least one unicast OfferService message
Pass Criteria	DUT: sends at least one unicast OfferService message
Reference	PRS_SOMEIPSD_00268;PRS_SOMEIPSD_00305; PRS_SOMEIPSD_00306;PRS_SOMEIPSD_00307
Notes	

5.1.6.1.139 SOMEIP_ETS_172: SOMEIP_ETS_173: SD_Unicast_SubscribeEventgroup

Synopsys	The Tester sends a unicast subscription request and expects the DUT to send a unicast subscription acknowledgement in response.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send unicast subscribeEventgroup indicating the Option settings: • index1stOptions = 0 • index2ndOptions = 1 • numOfOptions1 = 1 • numOfOptions2 = 1 2. DUT: sends unicast subscribeEventgroupAck 3. TESTER: send subscribeEventgroup indicating a deviating Endpoint Option configuration: • index1stOptions = 0 • index2ndOptions = 0 • numOfOptions1 = 2 • numOfOptions2 = 0 4. DUT: sends unicast subscribeEventgroupAck
Pass Criteria	DUT: sends unicast subscribeEventgroupAck

	DUT: sends unicast subscribeEventgroupAck
Reference	PRS_SOMEIPSD_00386; PRS_SOMEIPSD_00387;PRS_SOMEIPSD_00391
Notes	

5.1.6.1.140 SOMEIP_ETS_174: SD_Unknown_Option_type

Synopsys	The Tester sends a SubscribeEventgroup Message referencing an unknown Option type, this turns useless one of the two Endpoint Options needed by the DUT (since the unknown one has to be ignored) and so the Message must be answered with a SubscribeEventgroupNAck Message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send subscribeEventgroup with an unknown Option type 2. DUT: sends subscribeEventgroupNAck
Pass Criteria	DUT: sends subscribeEventgroupNAck
Reference	PRS_SOMEIPSD_00273;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.141 SOMEIP_ETS_175: SD_ Unreferenced_option

Synopsys	The Tester sends a SubscribeEventgroup Message containing all needed Endpoint Options including a not needed Configuration Endpoint Option This last Option is not referenced and so the Message must be answered by the DUT with an SubscribeEventgroupAck.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input	Check section general Input Parameters

Parameters	
Test Procedure	1. TESTER: send subscribeEventgroup containing all needed Endpoint Options including a not needed Configuration Endpoint Option 2. DUT: sends subscribeEventgroupAck
Pass Criteria	DUT: sends subscribeEventgroupAck
Reference	PRS_SOMEIPSD_00337;PRS_SOMEIPSD_00387;PRS_SOMEIPSD_00393
Notes	

5.1.6.1.142 SOMEIP_ETS_176: SD_Unused_data_after_Options_Array

Synopsys	The DUT receives a unicast SubscribeEventgroup Request with unused Payload Data right after the Options Array (and which is included in the SOME/IP Length Field) and is expected to answer with a SubscribeEventgroupAck Message. The extra payload Data is 0x30303a3031. After that a second SubscribeEventgroup is sent which also includes the extra bytes at the end but does not count them to the messages's total length.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send subscribeEventgroup with unused Payload Data right after the Options Array (and which is included in the SOME/IP Length Field) 2. DUT: sends subscribeEventgroupAck 3. TESTER: send subscribeEventgroup with unused Payload Data right after the Options Array but not counted to the messages's total length 4. DUT: sends subscribeEventgroupAck
Pass Criteria	DUT: sends subscribeEventgroupAck DUT: sends subscribeEventgroupAck
Reference	PRS_SOMEIPSD_00153PRS_SOMEIPSD_00273
Notes	

5.1.6.1.143 SOMEIP_ETS_177: SD_ Unused_data_after_Options_Array_wrong_length

Synopsys	The DUT receives a unicast SubscribeEventgroup Request with unused Payload Data, whos Length is not included in the SOME/IP Length Field, at the End and is expected to answer with an SubscribeEventgroupAck Message.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send subscribeEventgroup Request with unused Payload Data, whos Length is not included in the SOME/IP Length Field, at the end 2. DUT: sends subscribeEventgroupAck
Pass Criteria	DUT: sends subscribeEventgroupAck
Reference	PRS_SOMEIPSD_00153PRS_SOMEIPSD_00273
Notes	

5.1.6.1.144 SOMEIP_ETS_178: Subscribe_using_wrong_SOMEIP_MessageID

Synopsys	The Tester sends a SubscribeEventgroup Message whose SOME/IP header uses a wrong Message-ID for Service Discovery, this should lead the DUT to reject the Message since it cannot be interpreted as a valid SOME/IP-SD.
Prerequisites	DUT ETS is running and offering the Enhanced Testability Service
Test setup	Topology 1
Test Input Parameters	Check section general Input Parameters
Test Procedure	1. TESTER: send subscribeEventgroup Message whose SOME/IP header uses a wrong Message-ID for Service Discovery 2. DUT: sends subscribeEventgroupNAck
Pass Criteria	DUT: sends subscribeEventgroupNAck
Reference	PRS_SOMEIPSD_00306;PRS_SOMEIPSD_00307; PRS_SOMEIPSD_00380;PRS_SOMEIPSD_00393

Notes	