Mesh Profile (MESH)

Bluetooth® Test Suite

- **Revision**: MESH.TS.1.0.1.1
- **Revision Date**: 2019-07-25
- **Group Prepared By**: Mesh Working Group
- **Feedback Email**: mesh-main@bluetooth.org
This document, regardless of its title or content, is not a Bluetooth Specification subject to the licenses granted by the Bluetooth SIG Inc. ("Bluetooth SIG") and its members under the Bluetooth Patent/Copyright License Agreement and Bluetooth Trademark License Agreement.

THIS DOCUMENT IS PROVIDED "AS IS" AND BLUETOOTH SIG, ITS MEMBERS, AND THEIR AFFILIATES MAKE NO REPRESENTATIONS OR WARRANTIES AND DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, THAT THE CONTENT OF THIS DOCUMENT IS FREE OF ERRORS.

TO THE EXTENT NOT PROHIBITED BY LAW, BLUETOOTH SIG, ITS MEMBERS, AND THEIR AFFILIATES DISCLAIM ALL LIABILITY ARISING OUT OF OR RELATING TO USE OF THIS DOCUMENT AND ANY INFORMATION CONTAINED IN THIS DOCUMENT, INCLUDING LOST REVENUE, PROFITS, DATA OR PROGRAMS, OR BUSINESS INTERRUPTION, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, AND EVEN IF BLUETOOTH SIG, ITS MEMBERS, OR THEIR AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document is proprietary to Bluetooth SIG. This document may contain or cover subject matter that is intellectual property of Bluetooth SIG and its members. The furnishing of this document does not grant any license to any intellectual property of Bluetooth SIG or its members.

This document is subject to change without notice.

Copyright © 2015–2019 by Bluetooth SIG, Inc. The Bluetooth word mark and logos are owned by Bluetooth SIG, Inc. Other third-party brands and names are the property of their respective owners.
Contents

1 Scope ................................................................................................................................. 13

2 References, Definitions, and Abbreviations ..................................................................... 14
  2.1 References ..................................................................................................................... 14

3 Test Suite Structure (TSS) .............................................................................................. 15
  3.1 Test Strategy .................................................................................................................. 15
  3.2 Test Groups .................................................................................................................... 15
    3.2.1 ADV Provisioning Bearer ......................................................................................... 15
    3.2.2 Mesh Provisioning Service ...................................................................................... 15
    3.2.3 Provisioning Procedure .......................................................................................... 15
    3.2.4 Beacons .................................................................................................................. 15
    3.2.5 Network Layer ........................................................................................................ 15
    3.2.6 Relay ....................................................................................................................... 15
    3.2.7 Transport Layers ...................................................................................................... 16
    3.2.8 IV Update ................................................................................................................ 16
    3.2.9 Key Refresh .............................................................................................................. 16
    3.2.10 Friendship ............................................................................................................. 16
    3.2.11 Access Layer .......................................................................................................... 16
    3.2.12 Proxy Protocol ....................................................................................................... 16
    3.2.13 Mesh Proxy Service ............................................................................................... 16
    3.2.14 Configuration Model ............................................................................................ 16
    3.2.15 Health Model ......................................................................................................... 16

4 Test Cases ........................................................................................................................ 17
  4.1 Introduction .................................................................................................................... 17
  4.1.1 Test Case Identification Conventions ....................................................................... 17
  4.1.2 Conformance ............................................................................................................. 19
  4.1.3 Pass/Fail Verdict Conventions .................................................................................. 19
  4.2 Setup Preambles .......................................................................................................... 20
    4.2.1 State: Node PB-ADV Link Established ................................................................. 20
    4.2.2 State: Provisioner PB-ADV Link Established ......................................................... 20
    4.2.3 State: Node PB-GATT Link Established ................................................................. 20
    4.2.4 State: Provisioner PB-GATT Link Established ......................................................... 21
    4.2.5 Establish an ATT Bearer connection .................................................................... 21
    4.2.6 Network and Application Security Credentials ..................................................... 21
  4.3 ADV Provisioning Bearer ............................................................................................ 21
    4.3.1 MESH/NODE/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close] .... 22
    4.3.2 MESH/NODE/PBADV/BV-02-C [Ignore Parallel Link Open Attempts] ................ 23
    4.3.3 MESH/NODE/PBADV/BV-03-C [Unprovisioned Device Receives and Sends PB-ADV PDU] .... 24
    4.3.4 MESH/NODE/PBADV/BV-04-C [Unprovisioned Device Receives and Sends PB-ADV PDU with Segmentation] .... 25
    4.3.5 MESH/PVNR/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close – Provisioner] .......... 26
    4.3.6 MESH/PVNR/PBADV/BV-02-C [Provisioner Sends and Receives PB-ADV PDU] .......... 27
    4.3.7 MESH/PVNR/PBADV/BV-03-C [Provisioner Sends and Receives PB-ADV PDU with Segmentation] .... 28
    4.3.8 MESH/PVNR/PBADV/BI-01-C [PB-ADV Link Establishment Timeout] .................. 29
    4.3.9 MESH/NODE/PBADV/BI-01-C [Ignore Link Open Attempts with Incorrect Device UUID] .......... 30
4.3.10 MESH/NODE/PADV/BV-02-C [Ignore Provisioning Bearer Control PDU with RFU Bearer Opcode Value] 31
4.3.11 MESH/NODE/PADV/BV-03-C [Receive Link Close Message with RFU Reason Value] .................. 32
4.3.12 MESH/NODE/PADV/BV-04-C [Ignore Connectable Advertisement using Mesh Provisioning AD Type] 32

4.4 Mesh Provisioning Service .............................................................................................................. 33
4.4.1 MESH/NODE/MPS/BV-01-C [Advertising Unprovisioned State for GATT Provisioning] ......................... 33
4.4.2 MESH/NODE/MPS/BV-02-C [PB-GATT Link Establishment and Link Close] ........................................... 34
4.4.3 MESH/NODE/MPS/BV-03-C [Expose Mesh Provisioning Service – Unprovisioned Device] ...................... 35
4.4.4 Mesh Provisioning Service Characteristic Discovery as Unprovisioned Device ................................. 36
4.4.4.1 MESH/NODE/MPS/BV-04-C [Exposé Mesh Provisioning Data Out Characteristic – Unprovisioned Device] 36
4.4.4.2 MESH/NODE/MPS/BV-05-C [Exposé Mesh Provisioning Data In Characteristic – Unprovisioned Device] 36
4.4.5 MESH/NODE/MPS/BV-06-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptors – Unprovisioned Device] ................ 37
4.4.6 MESH/NODE/MPS/BV-07-C [Enable Notifications for Mesh Provisioning Data Out – Unprovisioned Device] 37
4.4.7 MESH/NODE/MPS/BV-08-C [Receive Mesh Provisioning Data Out Notifications – Unprovisioned Device] 38
4.4.8 MESH/NODE/MPS/BV-09-C [Unprovisioned Device Sends and Receives Complete Provisioning PDUs] 39
4.4.9 MESH/NODE/MPS/BV-10-C [Unprovisioned Device Sends and Receives Segmented Provisioning PDU Message] ........................................................................................................... 40
4.4.10 MESH/PVNR/MPS/BV-01-C [Advertising Unprovisioned State for GATT Provisioning – Provisioner] ... 41
4.4.11 MESH/PVNR/MPS/BV-02-C [PB-GATT Link Establishment and Link Close – Provisioner] ...................... 42
4.4.12 Mesh Provisioning Service Characteristic Discovery by Provisioner .................................................. 43
4.4.12.1 MESH/PVNR/MPS/BV-03-C [Discover Mesh Provisioning Data Out Characteristic - Provisioner] ........... 43
4.4.12.2 MESH/PVNR/MPS/BV-04-C [Discover Mesh Provisioning Data In Characteristic] ...................... 43
4.4.13 MESH/PVNR/MPS/BV-05-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor - Provisioner] .......................................................... 44
4.4.14 MESH/PVNR/MPS/BV-06-C [Enable Notifications for Mesh Provisioning Data Out - Provisioner] ...... 44
4.4.15 MESH/PVNR/MPS/BV-08-C [Provisioner Sends and Receives Complete Provisioning PDUs] .................. 45
4.4.16 MESH/PVNR/MPS/BV-09-C [Provisioner Sends and Receives Segmented Message] ....................... 46
4.4.17 MESH/NODE/MPS/BV-01-C [Mesh Provisioning Data In Characteristic Only Supports Proxy PDU containing Provisioning PDU] ........................................................................................................... 47

4.5 Provisioning Procedure ...................................................................................................................... 48
4.5.1 Provisioning Procedure – Unprovisioned Device ............................................................................. 48
4.5.1.1 MESH/NODE/PROV/BV-01-C [Provisioning Procedure, No Public Key OOB, Output OOB Authentication] ......................................................................................................................... 50
4.5.1.2 MESH/NODE/PROV/BV-02-C [Provisioning Procedure, No Public Key OOB, Input OOB Authentication] .......................................................................................................................... 50
4.5.1.3 MESH/NODE/PROV/BV-03-C [Provisioning Procedure, No Public Key OOB, Static OOB Authentication] .......................................................................................................................... 50
4.5.1.4 MESH/NODE/PROV/BV-04-C [Provisioning Procedure, Public Key OOB, Output OOB Authentication] .......................................................................................................................... 50
4.5.1.5 MESH/NODE/PROV/BV-05-C [Provisioning Procedure, Public Key OOB, Input OOB Authentication] .......................................................................................................................... 50
4.5.1.6 MESH/NODE/PROV/BV-06-C [Provisioning Procedure, Public Key OOB, Static OOB Authentication] .......................................................................................................................... 51
4.5.1.7 MESH/NODE/PROV/BV-07-C [Provisioning Procedure, No Public Key OOB, No OOB Authentication] .......................................................................................................................... 51
4.5.1.8 MESH/NODE/PROV/BV-08-C [Provisioning Procedure, Public Key OOB, No OOB Authentication] 51
4.5.2 MESH/NODE/PROV/BV-09-C [Provisioning Procedure, Procedure Timeout] ........................................ 52
4.5.3 MESH/NODE/PROV/BV-10-C [Unprovisioned Device Sends Provisioning Failed Message on Provisioning Error] .................................................................................. 53
4.5.4 MESH/NODE/PROV/BV-11-C [Provisioned Node Updates Security Flags Based on the Provisioning Data PDU] .......................................................................................... 54
4.5.5 MESH/NODE/PROV/BV-12-C [Generate New Public Key on Each Provisioning Session] .................. 55
4.5.6 Provisioning Procedure – Provisioner .......................................................... 57
4.5.6.1 MESH/PVNR/PROV/BV-01-C [Provisioning Procedure as Provisioner, No Public Key OOB, Output OOB Authentication] ................................................................. 59
4.5.6.2 MESH/PVNR/PROV/BV-02-C [Provisioning Procedure as Provisioner, No Public Key OOB, Input OOB Authentication] ................................................................. 59
4.5.6.3 MESH/PVNR/PROV/BV-03-C [Provisioning Procedure as Provisioner, No Public Key OOB, Static OOB Authentication] ................................................................. 59
4.5.6.4 MESH/PVNR/PROV/BV-04-C [Provisioning Procedure as Provisioner, Public Key OOB, Output OOB Authentication] ................................................................. 59
4.5.6.5 MESH/PVNR/PROV/BV-05-C [Provisioning Procedure as Provisioner, Public Key OOB, Input OOB Authentication] ................................................................. 59
4.5.6.6 MESH/PVNR/PROV/BV-06-C [Provisioning Procedure as Provisioner, Public Key OOB, Static OOB Authentication] ................................................................. 59
4.5.6.7 MESH/PVNR/PROV/BV-07-C [Provisioning Procedure as Provisioner, No Public Key OOB, No OOB Authentication] ................................................................. 59
4.5.6.8 MESH/PVNR/PROV/BV-08-C [Provisioning Procedure as Provisioner, Public Key OOB, No OOB Authentication] ................................................................. 60
4.5.7 MESH/PVNR/PROV/BV-09-C [Provisioning Procedure, Procedure Timeout – Provisioner] .......... 61
4.5.8 MESH/PVNR/PROV/BV-10-C [Provisioner Disconnects Provisioning Bearer on Provisioning Protocol Failure from Unprovisioned Device] .................. 62
4.5.9 MESH/NODE/PROV/BI-01-C [Device Resets Attention Timer State when Provisioning Bearer Dropped] 63
4.5.10 MESH/NODE/PROV/BI-02-C [Reject Provisioning Start PDU with RFU values] ................. 63
4.5.11 MESH/NODE/PROV/BI-03-C [Reject Provisioning Start PDU with Inconsistent Values] ........ 64
4.5.12 MESH/NODE/PROV/BI-13-C [Reject Invalid Public Keys] ........................................ 66
4.5.13 MESH/PVNR/PROV/BI-14-C [Reject Invalid Public Keys] ........................................ 68
4.6 Secure Network Beacons ........................................................................ 69
4.6.1 MESH/NODE/BCN/SNB/BV-01-C [Advertising Secure Network Beacons] .......................... 69
4.7 Network Layer .................................................................................. 70
4.7.1 Transmit Mesh Messages over Advertising Bearer to Different Destination Address Types ..... 70
4.7.1.1 MESH/NODE/NET/BV-01-C [Send Mesh Message to Unicast Address] ....................... 71
4.7.1.2 MESH/NODE/NET/BV-02-C [Send Mesh Message to Virtual Address] ....................... 72
4.7.1.3 MESH/NODE/NET/BV-03-C [Send Mesh Message to Dynamically Assigned Group Address] .... 72
4.7.1.4 MESH/NODE/NET/BV-04-C [Send Mesh Message to All-proxies Address] ................... 72
4.7.1.5 MESH/NODE/NET/BV-05-C [Send Mesh Message to All-friends Address] ................... 72
4.7.1.6 MESH/NODE/NET/BV-06-C [Send Mesh Message to All-relays Address] ................... 72
4.7.1.7 MESH/NODE/NET/BV-07-C [Send Mesh Message to All-nodes Address] ................... 72
4.7.2 Receive Mesh Messages over Advertising Bearer Sent to Different Destination Address Types ..... 73
4.7.2.1 MESH/NODE/NET/BV-08-C [Receive Mesh Message Sent to Unicast Address] ............... 74
4.7.2.2 MESH/NODE/NET/BV-09-C [Receive Mesh Message Sent to Virtual Address] ............... 74
4.7.2.3 MESH/NODE/NET/BV-10-C [Receive Mesh Message Sent to Dynamic Group Address] ...... 74
4.7.2.4 MESH/NODE/NET/BV-11-C [Receive Mesh Message Sent to All-proxies Address] ........... 74
4.7.2.5 MESH/NODE/NET/BV-12-C [Receive Mesh Message Sent to All-friends Address] .......... 74
4.7.2.6 MESH/NODE/NET/BV-13-C [Receive Mesh Message Sent to All-relays Address] .......... 75
4.7.2.7 MESH/NODE/NET/BV-14-C [Receive Mesh Message Sent to All-nodes Address] .......... 75
4.7.3 MESH/NODE/NET/BI-01-C [Ignore Connectable Advertisements on ADV Bearer] ............... 75
4.10.4 MESH/NODE/NET/BI-02-C [Discard Messages with Invalid Addresses] ............................................................. 76
4.10.5 MESH/NODE/NET/BI-03-C [Receive Network Packet with Invalid NID Field] .............................................................. 77
4.10.6 MESH/NODE/NET/BI-04-C [Receive Network Packet with RFU Address Value] ............................................................. 78

4.8 Message Relaying .................................................................................................................................................. 79

4.8.1 MESH/NODE/REL/BI-01-C [Relay Network PDUs based on SRC, DST, TTL and Relay Retransmit] 79
4.8.2 MESH/NODE/REL/BI-02-C [Network Message Cache] ......................................................................................... 82
4.8.3 MESH/NODE/REL/BI-01-C [Ignore Invalid IV Index] ............................................................................................ 85
4.8.4 MESH/NODE/REL/BI-02-C [Ignore Invalid SRC] ................................................................................................. 86

4.9 Transport Layers .................................................................................................................................................. 86

4.9.1 MESH/NODE/TNPT/BI-01-C [Ignore Message with Unknown Application Key] ....................................................... 86
4.9.2 MESH/NODE/TNPT/BI-02-C [Ignore Message with Unknown Device Key] ............................................................... 87
4.9.3 MESH/NODE/TNPT/BI-03-C [Send an Unsegmented Access Message] ................................................................. 87
4.9.4 MESH/NODE/TNPT/BI-04-C [Send a Segmented Access Message – Unicast] ............................................................ 88
4.9.5 MESH/NODE/TNPT/BI-05-C [Send a Segmented Access Message – Group/Virtual Address] ............................. 90
4.9.6 MESH/NODE/TNPT/BI-06-C [Receive an Unsegmented Access Message] ............................................................... 90
4.9.7 MESH/NODE/TNPT/BI-07-C [Receive a Segmented Message – Unicast] ................................................................. 91
4.9.8 MESH/NODE/TNPT/BI-08-C [Receive a Segmented Message – Group/Virtual Address] ....................................... 93
4.9.9 MESH/NODE/TNPT/BI-09-C [Send Segmented Message to Unicast – Stop When Transaction Rejected] 93
4.9.10 MESH/NODE/TNPT/BI-10-C [Send Segmented Message - Resend Unacknowledged Segments After Segment Transmission Timer Expires] ................................................................. 94
4.9.14 MESH/NODE/TNPT/BI-01-C [Ignore Invalid RFU Bits in Segment Acknowledgement Message] ...... 102
4.9.15 MESH/NODE/TNPT/BI-02-C [Ignore Message When Segments Received with Old SeqAuth] .................. 102

4.10 IV Update ............................................................................................................................................................ 103

4.10.1 MESH/NODE/IU/BI-01-C [IV Update Initiated by Receiving Secure Network Beacons] ............................. 103
4.10.2 MESH/NODE/IU/BI-02-C [IV Update Initiated by Sending Secure Network Beacons] ................................. 106
4.10.3 MESH/NODE/IU/BI-03-C [IV Update when Operating in Normal Operation for a Normal IV Index] 109
4.10.4 MESH/NODE/IU/BI-04-C [Stay in IV Update In Progress Operation When Waiting On Segmented Message Transport PDU Acknowledgement] ........................................................................ 110
4.10.5 MESH/NODE/IU/BI-05-C [IV Update for a Node Not on Primary Subnet] ......................................................... 112
4.10.6 MESH/NODE/IU/BI-01-C [Ignore IV Update Flag When Operating In Normal Operation For Less Than the Minimum Time] ................................................................. 113
4.10.7 MESH/NODE/IU/BI-02-C [Ignore IV Update Flag When Operating In Normal Operation For an Abnormal IV Index] ........................................................................................................ 114
4.10.8 MESH/NODE/IU/BI-04-C [Ignore IV Update Flag When Operating In Normal Operation For Message From Other Subnets] ........................................................................................................ 115
4.10.9 MESH/NODE/IU/BI-05-C [Ignore IV Update Flag When in Normal Operation state within 96 Hours of Exiting IV Update In Progress State] .......................................................... 117
4.10.10 MESH/NODE/IU/BI-06-C [IV Update – Invalid IV Index value] ............................................................................ 118

4.11 Key Refresh .......................................................................................................................................................... 119

4.11.1 MESH/NODE/KR/BI-01-C [Key Refresh Procedure as Node via Configuration Client Messages] ... 119
4.11.2 MESH/NODE/KR/BI-02-C [Key Refresh Procedure as Node via Secure Network Beacons] .............. 122
4.11.3 MESH/NODE/KR/BI-03-C [Key Refresh Procedure Skipping Phase 2] ......................................................... 124
4.11.4 MESH/CFGCL/KR/BI-01-C [Key Refresh as Configuration Client via Configuration Messages] .... 125
4.11.5 MESH/CFGCL/KR/BI-02-C [Key Refresh as Configuration Client via Secure Network Beacons] ... 128
4.11.6 MESH/CFGCL/KR/BI-03-C [Key Refresh with Blacklisting via Configuration Messages] ............. 130
4.11.7 MESH/CFGCL/KR/BI-04-C [Key Refresh with Blacklisting via Secure Network Beacons] ........... 131
4.11.8 MESH/NODE/KR/BI-01-C [Receive Config Key Refresh Phase Set Messages with Valid and Invalid Transition Values] ................................................................. 132
4.11.9 MESH/NODE/KR/BI-02-C [Ignore New Keys During Key Refresh Procedure] ............................................. 135
4.11.10 MESH/NODE/KR/BI-03-C [Receive Config Key Refresh Phase Get with Invalid NetKeyIndex] .................................................. 137
4.11.11 MESH/NODE/KR/BI-04-C [Receive Messages with Old Keys During Configuration Client Key Refresh] 137
4.11.12 MESH/NODE/KR/BI-05-C [Receive Messages with Old Keys During Secure Network Beacon Key Refresh] 139

4.12 Friendship................................................................................................................................. 140

4.12.1 Friend Node ............................................................................................................................ 140

4.12.1.2 MESH/NODE/FRND/FN/BV-02-C [Friendship Messaging – Friend Node] ................................. 142
4.12.1.3 MESH/NODE/FRND/FN/BV-03-C [Segment Acknowledgement – Friend Node] ....................... 144


4.12.1.9 MESH/NODE/FRND/FN/BV-09-C [Friendship – Friend Node Cached Cleared on Poll Timeout]. 155
4.12.1.10 MESH/NODE/FRND/FN/BV-10-C [Friendship – Friend Node Config Friend Set Received] .... 157


4.12.2 Low Power Node ..................................................................................................................... 185

4.12.2.1 MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node] ........ 185
4.12.2.3 MESH/NODE/FRND/LPN/BV-03-C [Friend IV Update – Low Power Node] ......................... 189
4.12.2.4 MESH/NODE/FRND/LPN/BV-04-C [Friendship – Send Friend Subscription List Add Message] . 191
4.12.2.7 MESH/NODE/FRND/LPN/BV-07-C [Friend Key Refresh – Low Power Node] ......................... 194
4.12.2.8 MESH/NODE/FRND/LPN/BV-08-C [Low Power Node Send Friend Clear Message] .......................... 197
4.12.2.9 MESH/NODE/FRND/LPN/BI-01-C [Friendship Establishment Mode – Low Power Node Friend Poll
Retries] ........................................................................................................ 198

4.13 Proxy Protocol .......................................................................................... 202
4.13.2 MESH/SR/PROX/BV-02-C [Complete Message Network PDU Relay to Proxy Client] ..................... 204
4.13.3 MESH/SR/PROX/BV-03-C [Segmented Message Network PDU Relay to Proxy Client] ..................... 205
4.13.4 MESH/SR/PROX/BV-04-C [Send a Complete Message Network PDU to Node from Proxy Client] .. 206
4.13.5 MESH/SR/PROX/BV-05-C [Send a Segmented Message Network PDU to Node from Proxy Client] 207
4.13.6 MESH/SR/PROX/BV-06-C [Filter White List Addresses] ................................................................. 209
4.13.7 MESH/SR/PROX/BV-07-C [Receive Black List Filter Setup] .............................................................. 210
4.13.8 MESH/SR/PROX/BV-08-C [Filter Black List Addresses] ................................................................. 212
4.13.9 MESH/SR/PROX/BV-09-C [Remove Address from White List] ......................................................... 213
4.13.10 MESH/SR/PROX/BV-10-C [Remove Address from Black List] ....................................................... 214
4.13.11 MESH/SR/PROX/BV-11-C [Send Secure Network Beacon When Proxy Connection Established] ... 215
4.13.12 MESH/SR/PROX/BV-12-C [Send Secure Network Beacon When IV Index Updated] .................... 215
4.13.13 MESH/SR/PROX/BV-13-C [Send Secure Network Beacon with Previous IV Index Value] ............ 216
4.13.14 MESH/SR/PROX/BV-14-C [IV Update Initiated by Proxy Client] ..................................................... 218
4.13.15 MESH/SR/PROX/B-01-C [Ignore Invalid Message Type] .............................................................. 219
4.13.16 MESH/CL/PROX/BV-01-C [Receive Complete Message from Proxy Server] ................................. 220
4.13.17 MESH/CL/PROX/BV-02-C [Receive Segmented Message from Proxy Server] ............................ 220
4.13.18 MESH/CL/PROX/BV-03-C [Send Complete Message to Proxy Server] ........................................ 221
4.13.19 MESH/CL/PROX/BV-04-C [Send Segmented Message to Proxy Server] ........................................ 222
4.13.20 MESH/CL/PROX/BV-05-C [Send Set Filter Type Proxy Configuration Message] .......................... 223
4.13.21 MESH/CL/PROX/BV-06-C [Send Add Addresses to Filter Proxy Configuration Message] ............ 224
4.13.22 MESH/CL/PROX/BV-07-C [Send Remove Addresses from Filter Proxy Configuration Message] .... 225
4.13.23 MESH/CL/PROX/BV-08-C [Receive Secure Network Beacon When Proxy Connection Established] ... 226
4.13.24 MESH/CL/PROX/BV-09-C [Send Secure Network Beacon When IV Index Updated] .................... 226
4.13.25 MESH/CL/PROX/BV-10-C [Send Secure Network Beacon with Previous IV Index Value] .......... 227
4.13.26 MESH/CL/PROX/B-01-C [Ignore Invalid Message Type] .............................................................. 228

4.14 Mesh Proxy Service ..................................................................................... 228
4.14.9 MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] ........................................................... 234

4.15 Configuration Model .................................................................................. 239
4.15.1 Secure Network Beacon Procedures ............................................................... 239
4.15.1.1 MESH/NODE/CFG/SNBP/BV-01-C [Respond to Config Beacon Set/Get Messages] ........................................ 239
4.15.1.2 MESH/NODE/CFG/SNBP/B1-01-C [Ignore invalid Config Beacon Set/Get Messages] ........................................ 241
4.15.1.3 MESH/CFGCL/CFG/SNBP/BV-01-C [Send Config Beacon Set/Get Messages] .................................................. 242
4.15.2 Composition Data Procedures ........................................................................ 243
4.15.2.1 MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] .................................. 244
4.15.2.2 MESH/CFGCL/CFG/COMP/BV-01-C [Send Config Composition Data Messages] .................................................. 245
4.15.3 Default TTL Procedure ..................................................................................... 246
4.15.3.1 MESH/NODE/CFG/DTTL/BV-01-C [Respond to Config Default TTL Set/Get Messages] .................................... 246
4.15.3.2 MESH/NODE/CFG/DTTL/B1-01-C [Respond to Config Default TTL Set Message (On Secondary Element or With Invalid TTL)] ........................................................................ 248
4.15.3.3 MESH/CFGCL/CFG/DTTL/BV-01-C [Send Config Default TTL Set/Get Messages] ............................................. 250
4.15.4 GATT Proxy Procedures ................................................................................... 251
4.15.4.1 MESH/NODE/CFG/GPXY/BV-01-C [Respond to Config GATT Proxy Set/Get Messages] ................................... 251
4.15.4.2 MESH/NODE/CFG/GPXY/BV-02-C [Network ID Advertising on Multiple Subnets] ............................................. 253
4.15.4.3 MESH/NODE/CFG/GPXY/B1-01-C [Respond to Config GATT Proxy Set Messages Invalid Parameters] .................. 254
4.15.4.4 MESH/CFGCL/CFG/GPXY/BV-01-C [Send Config GATT Proxy Set/Get Messages] ............................................. 255
4.15.5 Friend Procedures ........................................................................................... 256
4.15.5.1 MESH/NODE/CFG/CFGF/BV-01-C [Respond to Config Friend Get/Set Messages] ............................................ 256
4.15.5.2 MESH/NODE/CFG/CFGF/B1-01-C [Respond to Config Friend Set Messages Invalid Parameters] ....................... 259
4.15.5.3 MESH/CFGCL/CFG/CFGF/BV-01-C [Send Config Friend Get/Set Messages] .................................................. 260
4.15.6 Relay Procedures ............................................................................................ 261
4.15.6.1 MESH/NODE/CFG/CFGR/BV-01-C [Respond to Config Relay Get/Set Messages] ............................................. 261
4.15.6.2 MESH/NODE/CFG/CFGR/B1-01-C [Respond to Config Relay Set Messages Invalid Parameters] ....................... 263
4.15.6.3 MESH/CFGCL/CFG/CFGR/BV-01-C [Send Config Relay Get/Set Messages] .................................................. 264
4.15.7 Model Publication Procedures ......................................................................... 265
4.15.7.1 MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages] ........................................... 265
4.15.7.2 MESH/NODE/CFG/MP/BV-02-C [Disabling Publishing – AppKey/Config NetKey Deleted] .............................. 269
4.15.7.3 MESH/NODE/CFG/MP/B1-01-C [Respond to Config Model Publication Get/Set/Virtual Address Set Messages (Invalid ElementAddress)] .......................................................... 272
4.15.7.4 MESH/NODE/CFG/MP/B1-02-C [Respond to Config Model Publication Get/Set/Virtual Address Set Message (Invalid Model Identifier)] ................................................................. 273
4.15.7.5 MESH/CFGCL/CFG/MP/BV-03-C [Send Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages] ........................................... 275
4.15.7.6 MESH/CFGCL/CFG/MP/BV-01-C [Send Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages] ........................................... 276
4.15.8 Subscription List Procedures ......................................................................... 278
4.15.8.1 MESH/NODE/CFG/SL/BV-01-C [Respond to Config Model Subscription Messages] ........................................... 278
4.15.8.2 MESH/NODE/CFG/SL/BV-02-C [Respond to Config Model Subscription Virtual Address Messages] ......................... 281
4.15.8.3 MESH/NODE/CFG/SL/BV-03-C [Respond to Config Vendor Model Subscription Get Message] .......................... 284
4.15.8.4 MESH/NODE/CFG/SL/B1-01-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid ElementAddress)] .......................................................... 286
4.15.8.5 MESH/NODE/CFG/SL/B1-02-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid Model Identifier)] .......................................................... 287
4.15.8.6 MESH/NODE/CFG/SL/B1-03-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid ElementAddress)] .................................................. 289
4.15.8.7 MESH/NODE/CFG/SL/B1-04-C [Respond to Config SIG Model Subscription Get Message (Invalid ElementAddress)] ........................................................................ 290
4.15.8.8 MESH/NODE/CFG/SL/BI-05-C [Respond to Config SIG Model Subscription Get Message (Invalid Model Identifier)].................................................. 291
4.15.8.9 MESH/NODE/CFG/SL/BI-06-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid Model Identifier)].................................................. 292
4.15.8.10 MESH/NODE/CFG/SL/BI-07-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid ElementAddress)].......................... 293
4.15.8.11 MESH/NODE/CFG/SL/BI-08-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid Model Identifier)].......................... 295
4.15.8.12 MESH/NODE/CFG/SL/BI-09-C [Respond to Config Vendor Model Subscription Get Message (Invalid ElementAddress)].................................................. 296
4.15.8.13 MESH/NODE/CFG/SL/BI-10-C [Respond to Config Vendor Model Subscription Get Message (Invalid Model Identifier)].................................................. 297
4.15.8.14 MESH/CFGCL/CFG/SL/BV-01-C [Send Config Model Subscription Messages] .................................................. 298
4.15.8.15 MESH/CFGCL/CFG/SL/BV-02-C [Send Config Model Subscription Virtual Address Messages]. 301
4.15.9 NetKey List Procedures .................................................. 303
4.15.9.1 MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages].............. 303
4.15.9.2 MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message].......................... 305
4.15.9.3 MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message].......................... 306
4.15.9.4 MESH/NODE/CFG/NKL/BV-04-C [Receive Config NetKey Delete Message for Last NetKey].... 307
4.15.9.5 MESH/NODE/CFG/NKL/BI-01-C [Respond to Config NetKey Update Message (Invalid NetKeyIndex)].................................................. 308
4.15.9.6 MESH/NODE/CFG/NKL/BI-02-C [Respond to Config NetKey Add Message (NetKeyIndex Already Stored)].................................................. 309
4.15.9.7 MESH/NODE/CFG/NKL/BI-03-C [Respond to Config NetKey Add Message (Insufficient Resources)].................................................. 310
4.15.9.8 MESH/CFGCL/CFG/NKL/BV-01-C [Send Config NetKey Add/Get Messages].................................................. 311
4.15.9.9 MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages].......................... 312
4.15.9.10 MESH/CFGCL/CFG/NKL/BV-03-C [Send Config NetKey Delete Messages].......................... 313
4.15.10 AppKey List Procedures .................................................. 314
4.15.10.1 MESH/NODE/CFG/AKL/BV-01-C [Respond to Config AppKey Add/Get Message].................................................. 314
4.15.10.2 MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message].......................... 316
4.15.10.3 MESH/NODE/CFG/AKL/BV-03-C [Respond to Config AppKey Delete Message].......................... 318
4.15.10.4 MESH/NODE/CFG/AKL/BI-01-C [Respond to Config AppKey Add Message (Invalid NetKeyIndex)] 319
4.15.10.5 MESH/NODE/CFG/AKL/BI-02-C [Respond to Config AppKey Update Message (Invalid AppKey Index and NetKeyIndex)].................................................. 320
4.15.10.6 MESH/NODE/CFG/AKL/BI-03-C [Respond to Config AppKey Get Message with Invalid NetKey Index] 322
4.15.10.7 MESH/NODE/CFG/AKL/BI-04-C [Respond to Config AppKey Add Message (NetKeyIndex and AppKeyIndex Already Stored)].................................................. 323
4.15.10.8 MESH/CFGCL/CFG/AKL/BV-01-C [Send Config AppKey Add Messages].................................................. 325
4.15.10.9 MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages].......................... 325
4.15.10.10 MESH/CFGCL/CFG/AKL/BV-03-C [Send Config AppKey Delete Messages].......................... 326
4.15.10.11 MESH/CFGCL/CFG/AKL/BV-04-C [Send Config AppKey Get Messages].................................................. 327
4.15.11 Model to AppKey List Procedures .................................................. 327
4.15.11.1 MESH/NODE/CFG/MAKL/BV-01-C [Respond to Config Model App Bind and SIG/Config Vendor Model App Get Messages].................................................. 327
4.15.11.2 MESH/NODE/CFG/MAKL/BV-02-C [Respond to Config Model App Unbind and SIG/Config Vendor Model App Get Messages].................................................. 330
4.15.11.3 MESH/NODE/CFG/MAKL/BI-01-C [Respond to Config Model App Bind Message with Invalid Values] 332
4.15.11.4 MESH/NODE/CFG/MAKL/BI-02-C [Respond to Config Model App Unbind Message with Invalid Values] 334
4.15.11.5 MESH/NOE/CFG/MAKL/BI-03-C [Respond to Config SIG Model App Get Message with Invalid Values] 336
4.15.11.6 MESH/NOE/CFG/MAKL/BI-04-C [Respond to Config Vendor Model App Get Message with Invalid Values] ................................................................. 337
4.15.11.7 MESH/CFGCL/CFG/MAKL/BV-01-C [Send Config Model App Bind Messages] ................................................................. 339
4.15.11.8 MESH/CFGCL/CFG/MAKL/BV-02-C [Send Config Model App Unbind Messages] ................................................................. 340
4.15.11.9 MESH/CFGCL/CFG/MAKL/BV-03-C [Send Config SIG Model App Get Messages] ................................................................. 341
4.15.11.10 MESH/CFGCL/CFG/MAKL/BV-04-C [Send Config Vendor Model App Get Messages] ................................................................. 342
4.15.12 Node Identity Procedures................................................................. 343
4.15.12.1 MESH/NOE/CFG/NID/BV-01-C [Receive Config Node Identity Get/Set Messages] ................................................................. 343
4.15.12.2 MESH/NOE/CFG/NID/BV-02-C [Node Identity Advertising on Multiple Subnets] ................................................................. 345
4.15.12.3 MESH/NOE/CFG/NID/BV-03-C [Node Identity Config NetKey Deleted] ................................................................. 347
4.15.12.5 MESH/NOE/CFG/NID/BI-03-C [Receive Config Node Identity Set Message – Change to Not Supported] 350
4.15.12.6 MESH/CFGCL/CFG/NID/BV-01-C [Send Config Node Identity Get/Set Messages] ................................................................. 351
4.15.13 Reset Procedures............................................................................. 352
4.15.13.1 MESH/NOE/CFG/RST/BV-01-C [Receive Config Node Reset Message] ................................................................. 353
4.15.13.2 MESH/CFGCL/CFG/RST/BV-01-C [Send Config Node Reset Messages] ................................................................. 354
4.15.14 Heartbeat Publication Procedures................................................................. 355
4.15.14.1 MESH/NOE/CFG/HBP/BV-01-C [Receive Config Heartbeat Publication Set/Get messages] ................................................................. 355
4.15.14.2 MESH/NOE/CFG/HBP/BV-02-C [Periodic publishing of Heartbeat messages] ................................................................. 356
4.15.14.3 MESH/NOE/CFG/HBP/BV-03-C [Triggered publishing of Heartbeat messages – Relay State changes] 359
4.15.14.4 MESH/NOE/CFG/HBP/BV-04-C [Triggered publishing of Heartbeat messages - Friend state changes] 361
4.15.14.5 MESH/NOE/CFG/HBP/BV-05-C [Triggered publishing of Heartbeat messages - Low Power State changes] 364
4.15.14.6 MESH/NOE/CFG/HBP/BV-06-C [Disabling publishing of Heartbeat messages – Config NetKey Deleted] 367
4.15.14.7 MESH/NOE/CFG/HBP/BV-07-C [Triggered publishing of Heartbeat messages – Proxy state changes] 369
4.15.14.9 MESH/NOE/CFG/HBP/BI-02-C [Respond to Config Heartbeat Publication Get – Wrongly Encrypted by AppKey] ................................................................. 375
4.15.14.10 MESH/CFGCL/CFG/HBP/BV-01-C [Send Config Heartbeat Publication Set/Get Messages] ................................................................. 376
4.15.15 Heartbeat Subscription Procedures................................................................. 378
4.15.15.1 MESH/NOE/CFG/HBS/BV-01-C [Receive Config Heartbeat Subscription Set/Get message] ................................................................. 378
4.15.15.2 MESH/NOE/CFG/HBS/BV-02-C [Receive Heartbeat messages – Controlled by PeriodLog State] 380
4.15.15.3 MESH/NOE/CFG/HBS/BV-03-C [Not receiving Heartbeat messages – Unassigned or Mismatched SRC field] ................................................................. 383
4.15.15.4 MESH/NOE/CFG/HBS/BV-04-C [Receive Heartbeat messages – CountLog and Hops] ................................................................. 386
4.15.15.5 MESH/NOE/CFG/HBS/BV-05-C [Receive Heartbeat messages – Friendship] ................................................................. 389
4.15.15.6 MESH/NOE/CFG/HBS/BI-01-C [Ignore Config Heartbeat Subscription Set Message – Invalid Source Address or Destination Address] ................................................................. 391
4.15.15.7 MESH/CFGCL/CFG/HBS/BV-01-C [Send Config Heartbeat Subscription Set/Get Messages] ................................................................. 394
4.15.16 Network Transmit Procedures................................................................. 396
4.15.16.1 MESH/NOE/CFG/NTX/BV-01-C [Respond to Config Network Transmit Get/Set Messages] ................................................................. 396
4.15.16.2 MESH/CFGCL/CFG/NTX/BV-01-C [Send Config Network Transmit Get/Set Messages] ................................................................. 396
4.15.17 Low Power Node PollTimeout Procedures................................................................. 397
4.15.17.1  MESH/NODE/CFG/LPNPT/BI-01-C [Ignore Invalid Config Low Power Node PollTimeout Get Messages] 397
4.15.17.2  MESH/CFGCL/CFG/LPNPT/BV-01-C [Send Config Low Power Node PollTimeout Get Messages] 398
4.16  Health Model ............................................................................................................................................................................. 399
  4.16.1  Current Fault State .................................................................................................................................................................. 399
  4.16.1.1  MESH/SR/HM/CFS/BV-01-C [Send Health Current Status Messages - No Faults] .................................................. 399
  4.16.1.2  MESH/SR/HM/CFS/BV-02-C [Send Health Current Status Messages - With Faults] .................................................. 400
  4.16.2  Registered Fault State ............................................................................................................................................................. 401
  4.16.2.1  MESH/SR/HM/RFS/BV-01-C [Respond to Health Fault Get] ....................................................................................... 402
  4.16.2.2  MESH/SR/HM/RFS/BV-02-C [Respond to Health Fault Clear and Health Fault Clear Unacknowledged Messages] ......................................................................................................................... 403
  4.16.2.3  MESH/SR/HM/RFS/BV-03-C [Respond to Health Fault Test and Health Fault Test Unacknowledged Messages] .......................................................................................................................... 404
  4.16.2.4  MESH/SR/HM/RFS/BI-01-C [Ignore Invalid Health Fault Test message] ................................................................. 406
  4.16.2.5  MESH/SR/HM/RFS/BI-02-C [Ignore Invalid Health Fault Test Unacknowledged message] .................................. 407
  4.16.2.6  MESH/CL/HM/RFS/BV-01-C [Send Health Messages] ............................................................................................... 409
  4.16.3  Health Period State ............................................................................................................................................................... 411
  4.16.3.1  MESH/SR/HM/HPS/BV-01-C [Respond to Health Period Messages] ........................................................................ 411
  4.16.3.2  MESH/CL/HM/HPS/BV-01-C [Send Health Period Messages] .................................................................................. 412
  4.16.4  Attention Timer State ............................................................................................................................................................. 414
  4.16.4.1  MESH/SR/HM/ATS/BV-01-C [Respond to Attention Messages] .................................................................................. 414
  4.16.4.2  MESH/CL/HM/ATS/BV-01-C [Send Attention Messages] ......................................................................................... 415

5  Test Case Mapping ........................................................................................................................................................................... 417

6  Revision History and Contributors ..................................................................................................................................................... 427
1 Scope

This Bluetooth document contains the Test Suite Structure (TSS) and Test Cases (TC) to test the Mesh Profile specification.

The objective of this test suite is to provide a basis for interoperability for Bluetooth devices giving a high probability of air interface interoperability between different manufacturers' Bluetooth devices.
2 References, Definitions, and Abbreviations

2.1 References
This Bluetooth document incorporates, by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter.

[1] Bluetooth Core Specification 4.0 or later
[3] Mesh Profile Specification 1.0 or later
[5] GATT Test Suite, GATT.TS
3 Test Suite Structure (TSS)

3.1 Test Strategy

The test objectives are to verify Mesh Profile functionality that enables interoperability between Mesh nodes and to ensure nodes can be provisioned and operate as part of a Mesh network. The testing approach is to cover mandatory and optional requirements in the specification and to match these to the support of the IUT as described in the ICS Proforma.

The test equipment shall provide an implementation of the Radio, Bluetooth Controller and the parts of the Host needed to perform the test cases defined in the Mesh Profile Test Suite. For some test cases, it is necessary to stimulate the IUT from an Upper Tester. In practice, this could be implemented as a special test interface, an MMI, or another interface supported by the IUT. For some test cases, more than one Lower Tester will be required to execute the test procedure.

This test suite contains Valid Behavior (BV) tests complemented with Invalid Behavior (BI) tests where required. The test coverage was determined by cataloging specification requirements, accessing these requirements for testability, and mapping these testable requirements into test categories and test purposes.

This specification uses features common in many GATT-based specifications although it includes an advertising-based protocol that will introduce behavior per the data contained in the advertising packet.

3.2 Test Groups

The following test groups have been defined.

3.2.1 ADV Provisioning Bearer

Tests for the ADV Provisioning Bearer. These include the testing of Link Establishment and PB-ADV PDUs.

3.2.2 Mesh Provisioning Service

Tests for the Mesh Provisioning Service. These include the testing of GATT-based functionality and the testing of messages requiring segmentation and reassembly.

3.2.3 Provisioning Procedure

Tests for bearer agnostic testing of the provisioning procedure. These tests are intended to be executed for each bearer supported by the IUT.

3.2.4 Beacons

Tests covering the formatting and behavior of beacons, including advertising of Secure Network beacons.

3.2.5 Network Layer

Tests for the Network Layer. These include basic tests, message caching, invalid behavior tests, and future-proofing (RFU values) tests.

3.2.6 Relay

Tests for the Relay feature. These include basic Relay functionality, Network Message Cache, invalid behavior tests, and future-proofing (RFU values) tests.
3.2.7 Transport Layers
Tests for the Upper Transport Layer and Lower Transport Layer. These include basic tests, message replay protection, Segmentation and Reassembly, invalid behavior tests, and future-proofing (RFU values) tests.

3.2.8 IV Update
Tests for the IV Update procedure. These include tests for both initiating IV Update and participating in the IV Update procedure via Secure Network beacons. IV Update Test Mode is used to circumvent timer limits.

3.2.9 Key Refresh
Tests for the Key Refresh procedure. These include tests for initiating the Key Refresh procedure as a Configuration Client by both Secure Network beacons and Configuration messages and tests for a Node participating in Key Refresh by both Secure Network beacons and Configuration messages.

3.2.10 Friendship
Tests for the Friend feature. These include tests for both Friend nodes and Low Power nodes – including Friend Establishment, Friend Messaging (including caching), and Friend Management.

3.2.11 Access Layer
Tests for the Access Layer. These tests include basic tests, invalid behavior tests, and future-proofing (RFU values) tests.

3.2.12 Proxy Protocol
Tests for the proxy protocol. These tests include filters, advertising, and Proxy PDUs exchanged between Proxy Clients and Proxy Servers.

3.2.13 Mesh Proxy Service
Tests for the Mesh Proxy Service. These include the testing of advertising of Mesh Proxy Service, and GATT-based functionality including characteristic discovery and notifications.

3.2.14 Configuration Model
Tests for the Configuration Server Model and Configuration Client Model. These include the testing of messages exchanged between Configuration Servers and Configuration Clients. The following procedure groups are included: Secure Network Beacon, Composition Data, Default TTL, GATT Proxy, Friend, Relay, Model Publication, Subscription List, NetKey List, AppKey List, Model to AppKey List, Node Identity, Reset, Heartbeat Publication, and Heartbeat Subscription.

3.2.15 Health Model
Tests for the Health Server Model and Health Client Model. These include the testing of messages exchanged between a Health Server and Health Client during the following state instances: Current Fault State, Registered Fault State, Health Period State, and Attention Timer State.
## 4 Test Cases

### 4.1 Introduction

#### 4.1.1 Test Case Identification Conventions

Test cases shall be assigned unique identifiers per the conventions in [2]. The convention used here is

\[
\text{<spec abbreviation>/<IUT role>/<class>/<feat>/<func>/<subfunc>/<cap>/<xx>-<nn>-<y>}. 
\]

Bolded ID parts shall appear in the order prescribed. Non-bolded ID parts (if applicable) shall appear between the bolded parts. The order of the non-bolded parts may vary from test suite to test suite, but shall be consistent within each individual test suite.

<table>
<thead>
<tr>
<th>Identifier Abbreviation</th>
<th>Spec Identifier &lt;spec abbreviation&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESH</td>
<td>Mesh Profile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier Abbreviation</th>
<th>Role Identifier &lt;IUT role&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL</td>
<td>Client role</td>
</tr>
<tr>
<td>NODE</td>
<td>Node role</td>
</tr>
<tr>
<td>SR</td>
<td>Server role</td>
</tr>
<tr>
<td>PVNR</td>
<td>Provisioner role</td>
</tr>
<tr>
<td>CFGCL</td>
<td>Configuration Client role</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identifier Abbreviation</th>
<th>Feature Identifier &lt;feat&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PBADV</td>
<td>PB-ADV Provisioning</td>
</tr>
<tr>
<td>MPS</td>
<td>Mesh Provisioning Service</td>
</tr>
<tr>
<td>PROV</td>
<td>Provisioning Procedure</td>
</tr>
<tr>
<td>BCN</td>
<td>Beacons</td>
</tr>
<tr>
<td>NET</td>
<td>Network Layer</td>
</tr>
<tr>
<td>RLY</td>
<td>Relay</td>
</tr>
<tr>
<td>TNPT</td>
<td>Transport Layers</td>
</tr>
<tr>
<td>IVU</td>
<td>IV Update</td>
</tr>
<tr>
<td>KR</td>
<td>Key Refresh</td>
</tr>
<tr>
<td>FRND</td>
<td>Friendship</td>
</tr>
<tr>
<td>ACC</td>
<td>Access Layer</td>
</tr>
<tr>
<td>Identifier Abbreviation</td>
<td>Capability Identifier &lt;cap&gt;</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>FN</td>
<td>Friendship - Friend Node Procedures</td>
</tr>
<tr>
<td>LPN</td>
<td>Friendship - Low Power Node Procedures</td>
</tr>
<tr>
<td>SNBP</td>
<td>Configuration Model – Secure Network Beacon Procedure</td>
</tr>
<tr>
<td>COMP</td>
<td>Configuration Model – Composition Data Procedure</td>
</tr>
<tr>
<td>DTTL</td>
<td>Configuration Model – Default TTL Procedure</td>
</tr>
<tr>
<td>GPXY</td>
<td>Configuration Model - GATT Proxy Procedures</td>
</tr>
<tr>
<td>CFGF</td>
<td>Configuration Model – Friend Procedure</td>
</tr>
<tr>
<td>CFGR</td>
<td>Configuration Model – Relay Procedure</td>
</tr>
<tr>
<td>MP</td>
<td>Configuration Model – Model Publication Procedures</td>
</tr>
<tr>
<td>SL</td>
<td>Configuration Model – Subscription List Procedures</td>
</tr>
<tr>
<td>NKL</td>
<td>Configuration Model – NetKey List Procedures</td>
</tr>
<tr>
<td>AKL</td>
<td>Configuration Model – AppKey List Procedures</td>
</tr>
<tr>
<td>MAKL</td>
<td>Configuration Model – Model to AppKey List Procedures</td>
</tr>
<tr>
<td>NID</td>
<td>Configuration Model – Node Identity Procedures</td>
</tr>
<tr>
<td>RST</td>
<td>Configuration Model – Reset Procedures</td>
</tr>
<tr>
<td>KRP</td>
<td>Configuration Model – Key Refresh Procedures</td>
</tr>
<tr>
<td>HBP</td>
<td>Configuration Model – Heartbeat Publication Procedures</td>
</tr>
<tr>
<td>HBS</td>
<td>Configuration Model – Heartbeat Publication Procedures</td>
</tr>
<tr>
<td>NTX</td>
<td>Configuration Model – Network Transmit Procedures</td>
</tr>
<tr>
<td>LPNPT</td>
<td>Configuration Model – Low Power Node PollTimeout Procedures</td>
</tr>
<tr>
<td>CFS</td>
<td>Health Model – Current Fault State</td>
</tr>
<tr>
<td>RFS</td>
<td>Health Model – Registered Fault State</td>
</tr>
</tbody>
</table>
4.1.2 Conformance

When conformance is claimed, all capabilities indicated as mandatory for this Specification shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory capabilities and optional and conditional capabilities for which support is indicated are subject to verification as part of the Bluetooth Qualification Program.

The Bluetooth Qualification Program may employ tests to verify implementation robustness. The level of implementation robustness that is verified varies from one Specification to another and may be revised for cause based on interoperability issues found in the market.

Such tests may verify:

- That claimed capabilities may be used in any order and any number of repetitions that is not excluded by the Specification, OR
- That capability enabled by the implementations are sustained over durations expected by the use case, OR
- That the implementation gracefully handles any quantity of data expected by the use case, OR
- That in cases where more than one valid interpretation of the Specification exists, the implementation complies with at least one interpretation and gracefully handles other interpretations OR
- That the implementation is immune to attempted security exploits.

A single execution of each of the required tests is required in order to constitute a pass verdict. However, it is noted that in order to provide a foundation for interoperability, it is necessary that a qualified implementation consistently and repeatedly passes any of the applicable tests.

In any case, where a member finds an issue with the Test Plan Generator, the Test Case as described in the Test Suite, or with the Test System utilized, the Member is required to notify the responsible party via an errata request such that the issue may be addressed.

4.1.3 Pass/Fail Verdict Conventions

Each test case has an Expected Outcome section, which outlines all the detailed pass criteria conditions that shall be met by the IUT to merit a Pass Verdict.

The convention in this test suite is that, unless there is a specific set of fail condition outlined in the test case, the IUT fails the test case as soon one of the pass criteria conditions cannot be met. If this occurs the outcome of the test shall be the Fail Verdict.
4.2 Setup Preambles

4.2.1 State: Node PB-ADV Link Established

1. The IUT is induced to send Unprovisioned Device beacons with the Beacon Type set to 0x00 (Unprovisioned Device), the Device UUID value set to the IUT’s Device UUID, and the OOB Information field has the bit set to a non-RFU value supported by the manufacturer.
2. The Lower Tester scans for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the IUT.
3. The Lower Tester begins to periodically advertise PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents, the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU containing the Link Open message has the Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the IUT.
4. The Lower Tester receives a Link Ack message from the IUT before the link establishment timeout. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning». The PB-ADV PDU containing the Link Ack message has a BearerOpcode value of 0x01, the GPCF field is set to 0b11, the Generic Provisioning Payload is zero length, the Link ID is set to the value sent in the Link Open message, the Transaction Number value is set to 0x00, and the Link Ack message has no parameters.

4.2.2 State: Provisioner PB-ADV Link Established

1. The IUT is induced to periodically advertise a PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning». The PB-ADV PDU containing the Link Open message has a Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the Lower Tester as supplied in the IXIT [6].
2. The Lower Tester detects the PB-ADV PDU advertised by the IUT.
3. The Lower Tester periodically advertises a Link Ack message containing the newly allocated Link ID sent in step 1. The Lower Tester continues to advertise for at least 5 seconds to ensure the message is received by the IUT.

4.2.3 State: Node PB-GATT Link Established

1. IUT is induced to advertise the Mesh Provisioning Service in connectable advertising PDUs, interleaved with Unprovisioned Device beacons.
2. The Lower Tester scans for connectable advertisements from the IUT until it receives a service advertisement containing the Mesh Provisioning Service and with a Service Data field containing the IUT’s Device UUID.
3. Execute the steps in setup preamble 4.2.5.
5. The Lower Tester sends an ATT_Write_Request to the IUT with the handle of the Mesh Provisioning Data Out characteristic and the value set to 0x0001 to enable notifications.
4.2.4 State: Provisioner PB-GATT Link Established

1. Execute the steps in setup preamble 4.2.5.
2. The IUT is induced to discover the handles for the Mesh Provisioning Data In characteristic and the Mesh Provisioning Data Out characteristic exposed by the Mesh Provisioning Service on the Lower Tester by executing the procedures in GATT.TS [5] Discover All Characteristics of a Service – by Server, GATT/SR/GAD/BV-04-C.
3. The IUT is induced to send an ATT_Write_Request to the Lower Tester with the handle of the Mesh Provisioning Data Out characteristic and the value set to 0x0001 to enable notifications.

4.2.5 Establish an ATT Bearer connection

1. Establish an LE transport connection between the IUT and the Lower Tester.
2. Establish an L2CAP channel 0x0004 between the IUT and the Lower Tester over that LE transport.

4.2.6 Network and Application Security Credentials

Some test cases require, as initial condition, that the IUT has been provisioned and that it shares a common network key and an IV Index with the Lower Tester. This data is denoted as network security credentials.

Some test cases require the IUT and the Lower Tester to also share application security credentials, which consist of an application key defined in the IXIT file.

4.3 ADV Provisioning Bearer

The test group objective is to verify behavior specific to the PB-ADV provisioning bearer. These include the testing of Link Establishment and PB-ADV PDUs. These test cases also serve as common procedures for the general provisioning procedures when PB-ADV is the selected bearer.
4.3.1 MESH/NODE/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close]

- Test Purpose
  Verify that an unprovisioned device IUT can establish a PB-ADV provisioning bearer link. Verify the IUT sends correctly formatted PB-ADV packets.

- Reference
  [3] Section 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

- Test Procedure
  1. The Upper Tester orders the IUT to send Unprovisioned Device beacons with the Beacon Type set to 0x00 (Unprovisioned Device), the Device UUID value set to the IUT’s Device UUID, and the OOB Information field has the bit set to a non-RFU value supported by the manufacturer.
  2. The Lower Tester scans for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the IUT.
  3. The Lower Tester begins to periodically advertise PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents, the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU containing the Link Open message has the Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the IUT.
  4. The Lower Tester receives a Link Ack message from the IUT before the link establishment timeout. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning». The PB-ADV PDU containing the Link Ack message has a BearerOpcode value of 0x01, the GPCF field is set to 0b11, the Generic Provisioning Payload is zero length, the Link ID is set to the value sent in the Link Open message, the Transaction Number value is set to 0x00, and the Link Ack message has no parameters.
  5. The Lower Tester begins to periodically advertise a Link Close message with a Reason value of 0x00 to terminate the link. The Lower Tester sends this message until the IUT reports that the PB-ADV link has been closed.

- Expected Outcome
  Pass verdict
  The IUT broadcasts an Unprovisioned Device beacon that is formatted correctly and contains valid values.

  The IUT ignores the value of the Transaction Number field in the Generic Provisioning PDU containing the Link Open message.
The IUT responds to the Link Open message with a Link Ack message and establishes a PB-ADV link.

The IUT receives a Link Close message from the Lower Tester and reports that the PB-ADV link is closed.

All Generic Provisioning PDUs are sent using the Mesh Provisioning AD Type («Mesh Provisioning»).

All bits of the Transaction Number field are set to 0 in each PB-ADV PDU sent by the IUT. All PB-ADV PDUs are formatted correctly with the correct values and are non-connectable.

4.3.2 MESH/NODE/PBADV/BV-02-C [Ignore Parallel Link Open Attempts]

- **Test Purpose**
  
  Verify that an unprovisioned device IUT ignores multiple parallel Link Open attempts and only sends a Link Ack on the first received Link Open attempt and on subsequent Link Open messages with the same Link ID. Verify that after Link Close the IUT will then respond again to future Link Open attempts. Verify that the IUT accepts new Link Open attempts after 60 seconds have passed without receiving any provisioning PDUs.

- **Reference**
  
  [3] Section 3.3.1, 5.2.1, 5.3.1.4.1, 5.3.2

- **Initial Condition**
  
  - The IUT is an unprovisioned device.
  
  - The Lower Tester is a Provisioner.
  
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

- **Test Procedure**
  
  1. Execute the procedure in Section 4.2.1 to establish a link on the PB-ADV bearer.
  2. The Lower Tester immediately begins to periodically advertise a new Link Open message containing a different (2nd) random Link ID and the Device UUID of the IUT.
  3. The Lower Tester periodically scans for a Link Ack message containing the 2nd allocated Link ID sent in step 2. If the Lower Tester receives a Link Ack containing the 2nd allocated link ID, the test case ends in a fail verdict.
  4. The Lower Tester resends the Link Open message with the 1st Link ID and expects the IUT to respond with a Link Ack.
  5. The Lower Tester begins to periodically advertise a PB-ADV PDU with the 1st Link ID containing a Link Close message to terminate the link. The Link Close message field Info is set to 0b000010 and Control set to 0b11. The Lower Tester continues to advertise for at least 30 seconds to ensure the message is received by the IUT.
  6. Execute the procedure in Section 4.2.1 to establish a link on the PB-ADV bearer with a new random Link ID.
  7. The Lower Tester begins to periodically advertise a Link Close message using the 3rd Link ID to terminate the link. The Lower Tester continues to advertise for at least 5 seconds to ensure the message is received by the IUT.
  8. The Lower Tester sends a Link Open message with a new (4th) Link ID and expects the IUT to respond with a Link Ack.
9. The Lower Tester waits for 65 seconds, then it sends a Link Open message with a new (5th) Link ID and expects the IUT to respond with a Link Ack.
10. The Lower Tester advertises a Link Close message with the 5th Link ID, for at least 5 seconds.

• Expected Outcome

   **Pass verdict**

The IUT broadcasts an Unprovisioned Device beacon that is formatted correctly and contains valid values.

The IUT sends a Link Ack message to the Lower Tester after receiving a Link Open message and establishes a link over PB-ADV. The Link Ack message is properly formatted.

The IUT ignores the Link Open message(s) sent by the Lower Tester while the 1st link is still established and does not send a Link Ack message containing the 2nd allocated Link ID.

The IUT responds to the same Link Open message that has opened the link (containing the same Link ID).

The IUT successfully closes the 1st link and responds to the Link Open sent by the Lower Tester containing the 3rd allocated Link ID. The Link Ack message is properly formatted with the expected Info and Control values.

The IUT accepts a Link Open message with a different Link ID after 60 seconds have passed since sending the Link Ack.

**4.3.3 MESH/NODE/PBADV/BV-03-C [Unprovisioned Device Receives and Sends PB-ADV PDU]**

• **Test Purpose**

Verify that an unprovisioned device IUT can send and receive PB-ADV PDUs that do not require segmentation. Verify the formatting and values of the Mesh Provisioning AD Type, any PB-ADV PDUs contained in this AD Type, and any Generic Provisioning PDUs contained within a PB-ADV PDU.

• **Reference**

[3] Section 3.3.1, 4.3.1.4, 5.2.1, 5.3.3, 5.4.1

• **Initial Condition**

- The IUT is an unprovisioned device.
- The Lower Tester is a Provisioner.
- The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].
- The IUT is in State: Node PB-ADV Link Established after execution of setup preamble 4.2.1.
• Test Procedure

1. The Lower Tester sends a Provisioning PDU containing a Provisioning Invite. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU contains the Link ID set to the value used in link establishment and the Transaction Number is set to 0, and the Generic Provisioning PDU containing the Provisioning Invite message. The Generic Provisioning PDU contains two RFU bits set to 0, the Type field set to 0x00 ("Provisioning Invite"), and the Parameters field contains the Attention Duration field set to a value greater than 0.

2. The IUT receives the Provisioning PDU and sends a Provisioning PDU containing a Provisioning Capabilities PDU in response. The Provisioning PDU contains a 2-bit RFU field with all bits set to 0, a 6-bit Type field set to 0x01 ("Provisioning Capabilities"), and the Parameters field set to any valid Provisioning Capabilities field values.

• Expected Outcome

Pass verdict

The IUT receives a Provisioning PDU containing a Provisioning Invite PDU.

The IUT sends a Provisioning PDU containing a Provisioning Capabilities PDU.

All Mesh Provisioning AD Type messages are non-connectable.

4.3.4 MESH/NODE/PBADV/BV-04-C [Unprovisioned Device Receives and Sends PB-ADV PDU with Segmentation]

• Test Purpose

Verify that an unprovisioned device IUT can send and receive PB-ADV PDUs that require segmentation. Verify the formatting and values of the Mesh Provisioning AD Type, any PB-ADV PDUs contained in this AD Type, and any Generic Provisioning PDUs contained within any PB-ADV PDUs. Verify that segmented messages are transmitted in the correct order and Transaction Acknowledgement PDUs are sent when required.

• Reference

[3] Section 3.1.2, 3.3.1, 4.3.1.4, 5.2.1, 5.3.1.2, 5.3.1.3, 5.3.3, 5.4.1

• Initial Condition

- The IUT is an unprovisioned device.
- The Lower Tester is a Provisioner.
- The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].
- The IUT is in State: Node PB-ADV Link Established after execution of setup preamble 4.2.1.

• Test Procedure

1. The Lower Tester sends a Provisioning PDU containing a Provisioning Public Key PDU in multiple segments. The Generic Provisioning PDU contains two RFU bits set to 0, the Type field set to 0x03 ("Provisioning Public Key"), and the Parameters field set to validate the Provisioning Public Key field values. The first segment of the message is a Transaction Start PDU with SegN set to the last segment number, GPCF set to 0b00, TotalLength set to the number of octets in the
Provisioning PDU, and FCS set to the value calculated by the IUT. Each subsequent segment is transmitted as a Transaction Continuation PDU with the SegmentIndex set to the correct sequential value and the GPCF set to 0b10.

2. Upon receiving the PDU with the SegmentIndex value of SegN from the Transaction Start PDU, the IUT reassembles the Provisioning PDU. The IUT calculates the FCS for the reassembled Provisioning PDU.

3. The IUT compares the calculated FCS value with the FCS field value from the Transaction Start PDU. If the values match, the IUT sends a Transaction Acknowledgement PDU after a random delay between 20 ms and 50 ms with the RFU field bits set to 0, and the GPCF field set to 0b01. If the values do not match, the test case ends in failure.

4. The IUT sends a Provisioning PDU containing a Provisioning Public Key PDU in response. The Provisioning PDU contains a 2-bit RFU field with all bits set to 0, a 6-bit Type field set to 0x03 ("Provisioning Public Key"), and the Parameters field set to valid Provisioning Public Key field values.

5. The Lower Tester receives the Provisioning PDU, reassembles the message, and responds by sending a Transaction Acknowledgment PDU if FCS validation is successful.

• Expected Outcome

Pass verdict

The IUT sends a Transaction Acknowledgement PDU with the GPCF field set to 0b01.

The IUT receives a Provisioning PDU containing a Provisioning Public Key PDU in multiple segments, reassembles the message, and validates the FCS value. The IUT ignores any RFU bits that are set to 1.

The IUT sends a Provisioning PDU containing a Provisioning Public Key PDU. The initial segment (Segment 0) is sent using a Transaction Start PDU. The subsequent segments are sent using Transaction Continuation PDUs. Each Provisioning PDU is the length of the negotiated full MTU for the bearer (23 bytes), except the final segment which may be smaller. The number of segments received by the IUT matches the SegN value from the Transaction Start PDU.

All Mesh Provisioning AD Type messages are non-connectable.

Each Provisioning PDU is the length of the negotiated full MTU (23 bytes) for the bearer, except the final segment which may be smaller.

4.3.5 MESH/PVNR/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close – Provisioner]

• Test Purpose

Verify that a Provisioner IUT can establish a PB-ADV provisioning bearer link with an unprovisioned device. Verify the IUT sends correctly formatted PB-ADV packets.

• Reference

[3] Section 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

• Initial Condition

- The IUT is a Provisioner.
- The Lower Tester emulates an unprovisioned device.
- The IUT has been triggered to provision the Lower Tester.
- The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].

**Test Procedure**

1. The Lower Tester sends Unprovisioned Device beacons with the Beacon Type set to 0x00 (Unprovisioned Device), the Device UUID value set to the IUT’s Device UUID, and the OOB Information field has the bit set to a non-RFU value supported by the manufacturer.
2. The IUT is induced to scan for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the Lower Tester.
3. The IUT is induced to periodically advertise a PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning». The PB-ADV PDU containing the Link Open message has a Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the Lower Tester as supplied in the IXIT [6].
4. The Lower Tester detects the PB-ADV PDU advertised by the IUT.
5. The Lower Tester periodically advertises a Link Ack message containing the newly allocated Link ID sent in Step 1. The Lower Tester continues to advertise for at least 5 seconds to ensure the message is received by the IUT.
6. The IUT is induced to send a Link Close message with the Reason field set to 0x02 to terminate the link. The message is sent at least three times to ensure the message is received by the Lower Tester.

**Expected Outcome**

*Pass verdict*

The IUT and the Lower Tester establish a link on the PB-ADV bearer.

The IUT advertises a PB-ADV PDU containing a Link Open message. The GPCF field is set to 0b11.

All bits of the Transaction Number field are set to 0 in each PB-ADV PDU transmitted and sent by the IUT. All PB-ADV PDUs are formatted correctly with the correct values.

The IUT sends a properly formatted Link Close message with the Reason field set to 0x02.

**4.3.6 MESH/PVNR/PBADV/BV-02-C [Provisioner Sends and Receives PB-ADV PDU]**

**Test Purpose**

Verify that a Provisioner IUT can send and receive PB-ADV PDUs that do not require segmentation. Verify the formatting and values of the Mesh Provisioning AD Type, any PB-ADV PDUs contained in this AD Type, and any Generic Provisioning PDUs contained within a PB-ADV PDU.

**Reference**

[3] Section 3.1.2, 3.3.1, 4.3.1.4, 5.2.1, 5.3.3, 5.4.1, 5.4.2
• Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].
  - The IUT is in State: Provisioner PB-ADV Link Established after execution of setup preamble 4.2.2.

• Test Procedure
  1. The IUT is induced to send a Provisioning PDU containing a Provisioning Invite PDU. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents, the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU contains the Link ID set to the value used in link establishment and the Transaction Number is set to 0, and the Generic Provisioning PDU containing the Provisioning Invite message. The Generic Provisioning PDU contains two RFU bits set to 0, the Type field set to 0x00 (“Provisioning Invite”), and the Parameters field contains the Attention Duration field set to a value greater than 0.
  2. The Lower Tester receives the Provisioning PDU and sends a Provisioning PDU containing a Provisioning Capabilities PDU in response. The Provisioning PDU contains a 2-bit RFU field with all bits set to 0, a 6-bit Type field set to 0x01 (“Provisioning Capabilities”), and the Parameters field set to valid Provisioning Capabilities field values.

• Expected Outcome
  Pass verdict

  The IUT sends the Provisioning PDU containing the Provisioning Invite PDU.
  The IUT receives the Provisioning PDU containing the Provisioning Capabilities PDU.

  All Mesh Provisioning AD Type messages are non-connectable.

4.3.7 MESH/PVNR/PBADV/BV-03-C [Provisioner Sends and Receives PB-ADV PDU with Segmentation]

• Test Purpose
  Verify that a Provisioner IUT can send and receive PB-ADV PDUs that require segmentation. Verify the formatting and values of the Mesh Provisioning AD Type, any PB-ADV PDUs contained in this AD Type, and any Generic Provisioning PDUs contained within any PB-ADV PDUs. Verify that segmented messages are transmitted in the correct order and Transaction Acknowledgement PDUs are sent when required.

• Reference
  [3] Section 3.3.1, 5.2.1, 5.3.1, 5.3.3, 5.4.1, 5.4.2

• Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].
- The IUT is in State: Provisioner PB-ADV Link Established after execution of setup preamble 4.2.2.

• Test Procedure

1. The Upper Tester sends a command to the IUT to send a Provisioning PDU containing a Provisioning Public Key PDU in multiple segments. The Generic Provisioning PDU contains two RFU bits set to 0, the Type field set to 0x03 ("Provisioning Public Key"), and the Parameters field set to valid Provisioning Public Key field values. The first segment of the message is a Transaction Start PDU with SegN set to the last segment number, GPCF set to 0b00, TotalLength set to the number of octets in the Provisioning PDU, and FCS set to the value calculated by the IUT.

2. The Lower Tester receives the Provisioning PDU, reassembles the message, and responds by sending a Transaction Acknowledgment PDU if FCS validation is successful.

3. The Lower Tester sends a Provisioning PDU containing a Provisioning Public Key PDU to the IUT. The Provisioning PDU contains a 2-bit RFU field with all bits set to 0, a 6-bit Type field set to 0x03 ("Provisioning Public Key"), and the Parameters field set to valid Provisioning Public Key field values. The first segment of the message is a Transaction Start PDU with SegN set to the last segment number, GPCF set to 0b00, TotalLength set to the number of octets in the Provisioning PDU, and FCS set to the value calculated by the IUT. Each subsequent segment is transmitted as a Transaction Continuation PDU with the SegmentIndex set to the correct sequential value and the GPCF set to 0b10.

4. Upon receiving the Transaction Continuation PDU with the SegmentIndex value equal to the value of SegN in the Transaction Start PDU, the IUT reassembles the Provisioning PDU. The IUT calculates the FCS for the Provisioning PDU once all segments are received.

5. The IUT compares the calculated FCS to the value of the FCS field value from the Transaction Start PDU. If the values match, the IUT sends a Transaction Acknowledgement PDU after a random delay between 20 ms and 50 ms with all bits of the RFU field set to 0 and the GPCF field set to 0b01. If the values do not match, the test case ends in failure.

• Expected Outcome

Pass verdict

The IUT sends the Provisioning PDU containing the Provisioning Public Key PDU. The initial segment (Segment 0) is sent using a Transaction Start PDU. The subsequent segments are sent using Transaction Continuation PDUs. Each Provisioning PDU is the length of the negotiated full MTU for the bearer (23 bytes), except the final segment which may be smaller. All segments are sent in sequence.

The IUT receives the Provisioning PDU containing the Provisioning Capabilities PDU in multiple segments, reassembles the message, and validates the FCS value.

The IUT sends a Transaction Acknowledgement PDU with the RFU field bits set to 0 and the GPCF field set to 0b01 after calculating the FCS.

All Mesh Provisioning AD Type messages are non-connectable.

4.3.8 MESH/PVNR/PBADV/BI-01-C [PB-ADV Link Establishment Timeout]

• Test Purpose

Verify that a Provisioner IUT properly handles a timeout while establishing a PB-ADV link.
• **Reference**
  
  [3] Section 3.3.1, 5.2.1, 5.2.4, 5.3.3, 5.4.4

• **Initial Condition**
  
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The IUT has been triggered to provision the Lower Tester.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].

• **Test Procedure**

  1. The IUT is induced to periodically advertise a PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU contains a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU field contains the Provisioning Bearer Control PDU containing the Link Open message with the BearerOpcode set to 0x00, the GPCF field to 0b11, and the Parameters set to the Device UUID of the Lower Tester.
  2. The Lower Tester detects the PB-ADV PDU advertised by the IUT.
  3. The Lower Tester does not send a Link Ack message to the IUT within the timeout period (at least 60 seconds).
  4. The Lower Tester does not receive a Provisioning Failure PDU from the IUT after the timeout period is exceeded.

• **Expected Outcome**

  **Pass verdict**

  The IUT transmits a PB-ADV PDU containing a Link Open message. All fields are correctly formatted and contain valid values.

  The IUT does not send a Provisioning Failure PDU after the timeout is exceeded without receiving a Link Ack message from the Lower Tester.

4.3.9 **MESH/NODE/PBADV/BI-01-C [Ignore Link Open Attempts with Incorrect Device UUID]**

• **Test Purpose**

  Verify that an unprovisioned device IUT ignores Link Open attempts containing the incorrect Device UUID and only sends a Link Ack when the Device UUID matches.

• **Reference**

  [3] Section 3.3.1, 5.2.1. 5.3.1.4

• **Initial Condition**

  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The IUT is in the unprovisioned state.
- The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

• Test Procedure

1. The Upper Tester sends a command to the IUT to send Unprovisioned Device beacons with the following field values: the Beacon Type set to 0x00 (Unprovisioned Device), the Device UUID value as supplied in the IXIT [6], and the OOB Information field and URI Hash field set to valid values.

2. The Lower Tester scans for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the IUT.

3. The Lower Tester attempts to execute setup preamble 4.2.2 using a Device UUID different than the Device UUID contained in the Unprovisioned Device beacon. The Lower Tester expects the setup to fail with the IUT not sending a Link ACK message.

4. The Lower Tester periodically scans for 100 ms for a Link Ack message containing the 1st allocated Link ID sent in step 3. If the Lower Tester receives a Link Ack containing the 1st allocated link ID, the test case ends in a fail verdict.

5. The Lower Tester executes the procedure setup preamble 4.2.2 using the Device UUID in the IUT’s Unprovisioned Device beacons.

6. The Lower Tester begins to periodically advertise a Link Close message using the 2nd Link ID to terminate the link. The Lower Tester continues to advertise for at least 5 seconds to ensure the message is received by the IUT.

• Expected Outcome

Pass verdict

The IUT broadcasts an Unprovisioned Device beacon containing its Device UUID.

The IUT ignores the Link Open message(s) sent by the Lower Tester with the incorrect Device UUID and does not send a Link Ack message containing the 1st allocated Link ID.

The IUT responds to the Lower Tester’s Link Open message containing the correct Device UUID with a Link Ack message establishing the PB-ADV link. The PB-ADV PDU contains the expected 2nd Link ID and Transaction Number.

4.3.10 MESH/NODE/PBADV/B1-02-C [Ignore Provisioning Bearer Control PDU with RFU BearerOpcode Value]

• Test Purpose

Verify that an unprovisioned device IUT will ignore a message that has the BearerOpcode field set to an RFU value.

• Reference

[3] Section 3.1.2, 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

• Initial Condition

- The IUT is an unprovisioned device.
- The Lower Tester is a Provisioner.
- The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].
• Test Procedure
  1. The Lower Tester begins advertising a PB-ADV PDU containing a random Link ID and the Transaction Number set to 0. The Generic Provisioning PDU field contains a Provisioning Bearer Control PDU, with the BearerOpcode set to an RFU value, the GPCF field to 0b11, and the Parameters set to the Device UUID of the IUT.
  2. The Upper Tester confirms that the IUT receives the message from the Lower Tester.
  3. The Lower Tester does not expect any message in response from the IUT.

• Expected Outcome
  Pass verdict
  The IUT does not send any message in response to detecting the PDU from the Lower Tester.

4.3.11 MESH/NODE/PBADV/BI-03-C [Receive Link Close Message with RFU Reason Value]
• Test Purpose
  Verify that an unprovisioned device IUT will accept a Link Close message that has the Reason field set to an RFU value.

• Reference
  [3] Section 3.1.2, 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

• Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

• Test Procedure
  1. The steps in setup preamble 4.2.1 are executed to establish a link over the advertising bearer.
  2. The Lower Tester begins to periodically advertise a Link Close message with the Reason field set to any RFU value. The Lower Tester sends this message until the IUT reports that the PB-ADV link has been closed.

• Expected Outcome
  Pass verdict
  In step 2, The IUT closes the PB-ADV link.

4.3.12 MESH/NODE/PBADV/BI-04-C [Ignore Connectable Advertisement using Mesh Provisioning AD Type]
• Test Purpose
  Verify that an unprovisioned device IUT can ignore a connectable advertisement using the Mesh Provisioning AD Type.
• Reference
  
  [3] Section 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

• Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

• Test Procedure
  1. The IUT is induced to send Unprovisioned Device beacons with the Device UUID value set to the IUT’s Device UUID.
  2. The Lower Tester scans for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the IUT.
  3. The Lower Tester begins to periodically advertise a connectable PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents, the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU containing the Link Open message has the Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the IUT.

• Expected Outcome
  
  Pass verdict

  The IUT ignores the connectable advertisement sent by the Lower Tester.

4.4 Mesh Provisioning Service

The test group objective is to the Mesh Provisioning Service. This includes tests covering GATT requirements of the Mesh Provisioning Service and functionality related to segmentation and reassembly for Proxy PDUs received and sent by the Mesh Provisioning Service.

4.4.1 MESH/NODE/MPS/BV-01-C [Advertising Unprovisioned State for GATT Provisioning]

• Test Purpose
  
  Verify that an unprovisioned device that supports GATT provisioning indicates this in an advertisement with a List of Service UUIDs that includes the Mesh Provisioning Service. Verify that the advertising payload Service Data is properly formatted with the expected Device UUID. Verify that a connectable advertisement is utilized.

• Reference
  
  [3] Section 3.3.2, 4.3.1.4, 4.3.1.5, 5.2.2, 5.4.1, 5.4.2, 7.1.2.1, 7.1.2.2.1

• Initial Condition
  
  - The IUT is an unprovisioned device.
- The IUT supports PB-GATT provisioning.
- The IUT is in either Limited Discoverable mode or General Discoverable mode.
- The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].
- The Lower Tester is a Provisioner.
- Support for optional AD Types is supplied by the manufacturer via IXIT [6].
- Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

**Test Procedure**

1. The Upper Tester orders the IUT to advertise the Mesh Provisioning Service as a «Primary Service» in connectable advertising PDUs. The advertisements include the following AD Types: a 3-octet «Flags» type, either a 4-octet «Incomplete List of 16-bit Service UUIDs» type or a 4-octet «Complete List of 16-bit Service UUIDs» type that includes «Mesh Provisioning Service», and a 22-octet «Service Data» type, including the Device UUID and OOB Information. Any of the following optional AD Types are included if manufacturer support is indicated: URI, Appearance, TX Power Level, Local Name.
2. The Lower Tester receives the advertisements from the IUT.

**Expected Outcome**

*Pass verdict*

The IUT sends connectable service advertisements containing the Mesh Provisioning Service UUID.

The included Device UUID matches the expected value for the IUT.

The IUT’s service advertisement uses a connectable advertising PDU and contains the expected AD Types as described in step 1. The list of Service Class UUIDs contains a Service UUID of «Mesh Provisioning Service». The Service Data field must contain the expected Device UUID and OOB Information field values.

**4.4.2 MESH/NODE/MPS/BV-02-C [PB-GATT Link Establishment and Link Close]**

**Test Purpose**

Verify that an unprovisioned device IUT can establish a PB-GATT provisioning bearer link with a Provisioner. Verify the IUT sends correctly formatted PB-GATT packets.

**Reference**

[3] Section 3.3.2, 5.2.2

**Initial Condition**

- The IUT is an unprovisioned device.
- The Lower Tester is a Provisioner.
- The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

**Test Procedure**

1. The steps in Setup Preamble 4.2.3 are executed using the IUT’s Device UUID.
2. The Lower Tester terminates the ATT bearer connection.

- **Expected Outcome**
  
  **Pass verdict**

  The IUT advertises the Mesh Provisioning Service in connectable advertising PDUs containing its Device UUID, interleaved with Unprovisioned Device beacons.

  The Lower Tester can establish an ATT bearer connection with the IUT, discover the Provisioning Data characteristic, and configure it for notifications.

  The ATT bearer connection is closed between the IUT and the Lower Tester.

### 4.4.3 MESH/NODE/MPS/BV-03-C [Expose Mesh Provisioning Service – Unprovisioned Device]

- **Test Purpose**
  
  Verify that an unprovisioned device can support discovery of the Mesh Provisioning Service as a Primary Service. Also, verify that the Mesh Proxy Service is not discoverable.

- **Reference**
  
  [3] Section 3.3.2, 5.2.2, 7.1, 7.1.2.1

- **Initial Condition**
  
  - The IUT is an unprovisioned device.
  - The IUT is in either Limited Discoverable mode or General Discoverable mode.
  - The Lower Tester is a Provisioner.
  - The IUT and the Lower Tester support the GAP Peripheral and GAP Central roles.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

- **Test Procedure**


- **Expected Outcome**
  
  **Pass verdict**

  The IUT returns one attribute handle range containing the starting handle and the ending handle of the instantiation of the Mesh Provisioning Service. The Attribute Type in that service declaration is «Primary Service».
The IUT returns zero attribute handle ranges in response to discovery requests of the Mesh Proxy Service.

### 4.4.4 Mesh Provisioning Service Characteristic Discovery as Unprovisioned Device

This test group is for generic use and contains one or more test cases to verify compliant operation in response to characteristic discovery as IUT acting as an unprovisioned device. The verification is done one value at a time, as enumerated in the test cases in Table 4.2 below, using this generic test procedure.

- **Reference**
  - [3] Section 3.3.2, 7.1.2, 7.1.3

- **Initial Condition**
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The Lower Tester has discovered the Mesh Provisioning Service and has saved the handle range for an instantiation of the Mesh Provisioning Service that exposes the characteristic specified in Table 4.2 (e.g., by executing test case MESH/NODE/MPS/BV-03-C [Expose Mesh Provisioning Service – Unprovisioned Device]).

- **Test Procedure**
  1. The Upper Tester orders the IUT to execute either of the procedures included in GATT.TS [5]: Discover All Characteristics of a Service, GATT/SR/GAD/BV-04-C, with the specified handle range for the instantiation of the Mesh Provisioning Service, or Discover Characteristic by UUID, GATT/SR/GAD/BV-05-C, with the specified handle range for the instantiation of the Mesh Provisioning Service and UUID set to the Characteristic UUID in Table 4.2. In the selected procedure, only one pass is needed with the server database defined in Initial Condition.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Characteristic UUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.4.2</td>
<td>MESH/NODE/MPS/BV-05-C [Expose Mesh Provisioning Data In Characteristic – Unprovisioned Device] «Mesh Provisioning Data In»</td>
</tr>
</tbody>
</table>

*Table 4.2: Mesh Provisioning Service Characteristic Discovery as Unprovisioned Device Test Cases*

- **Expected Outcome**
  - **Pass verdict**
  
  One attribute handle/value pair is returned containing the specified Characteristic UUID.
4.4.5 MESH/NODE/MPS/BV-06-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor – Unprovisioned Device]

- **Test Purpose**
  Verify that an unprovisioned device IUT allows discovery of the Mesh Provisioning Data Out Client Characteristic Configuration descriptor.

- **Reference**
  [3] Section 3.3.2, 7.1.2, 7.1.3

- **Initial Condition**
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The IUT has discovered the attribute handle/value pair of the Mesh Provisioning Data Out characteristic (e.g., by executing test case MESH/NODE/MPS/BV-04-C [Expose Mesh Provisioning Data Out Characteristic – Unprovisioned Device]).

- **Test Procedure**
  The Lower Tester executes one pass of the procedure included in GATT.TS [5]: Discover all Characteristic Descriptors – from Server, GATT/SR/GAD/BV-06-C using the specified handle range.

- **Expected Outcome**
  **Pass verdict**

  The IUT returns one attribute handle/value pair containing the UUID «Client Characteristic Descriptor».

  The characteristic descriptor does not require authentication or encryption for access.

4.4.6 MESH/NODE/MPS/BV-07-C [Enable Notifications for Mesh Provisioning Data Out – Unprovisioned Device]

- **Test Purpose**
  Verify that an unprovisioned device IUT can have notifications enabled for the Mesh Provisioning Data Out characteristic by a Provisioner.

- **Reference**
  [3] Section 3.3.2, 7.1.2, 7.1.3

- **Initial Condition**
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
- The handle of the Mesh Provisioning Data Out characteristic is known to the Lower Tester by executing the test procedure in MESH/NODE/MPS/BV-04-C [Expose Mesh Provisioning Data Out Characteristic – Unprovisioned Device] or is known to the IUT by other means.

- The handle of the Mesh Provisioning Data Out Client Characteristic Configuration descriptor has been previously discovered by the IUT by executing the test procedure in MESH/NODE/MPS/BV-06-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor – Unprovisioned Device] or is known to the IUT by other means.

- Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

• Test Procedure

1. The Lower Tester disables notification by sending an ATT_Write_Request with the handle of the Mesh Provisioning Data Out characteristic and the value 0x0000.

2. The Lower Tester sends a correctly formatted ATT_Write_Request (Code = 0x12) with the handle of the characteristic descriptor and a value of 0x0001 to the Lower Tester.

• Expected Outcome

  Pass verdict

  The characteristic descriptor is successfully written.

4.4.7 MESH/NODE/MPS/BV-08-C [Receive Mesh Provisioning Data Out Notifications – Unprovisioned Device]

• Test Purpose

Verify that an unprovisioned device IUT can send notifications of the Mesh Provisioning Data Out characteristic.

• Reference

  [3] Section 3.3.2, 5.2.2, 7.1.2, 7.1.3

• Initial Condition

- The IUT is an unprovisioned device.

- The Lower Tester is a Provisioner.

- The handle of the Mesh Provisioning Data Out characteristic is known to the Lower Tester by executing the test procedure in MESH/NODE/MPS/BV-04-C [Expose Mesh Provisioning Data Out Characteristic – Unprovisioned Device] or is known to the IUT by other means.

- The handle of the Client Characteristic Configuration descriptor of the Mesh Provisioning Data Out characteristic has been previously discovered by the IUT by executing the test procedure in MESH/NODE/MPS/BV-06-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor – Unprovisioned Device] or is known to the IUT by other means.

- Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
• Test Procedure

1. The Lower Tester configures the IUT for notifications (e.g., by executing test case MESH/NODE/MPS/BV-07-C [Enable Notifications for Mesh Provisioning Data Out – Unprovisioned Device]).

2. The Lower Tester sends a Provisioning message to the IUT (e.g., a Provisioning Invite PDU) and the IUT responds with a Provisioning message appropriate for the message received (e.g., a Provisioning Capabilities PDU) using Handle Value Notification ATT PDU.

• Expected Outcome

Pass verdict

The IUT responds to a Provisioning message from the Lower Tester with a Provisioning message after notifications are enabled for Mesh Provisioning Data Out.

Characteristics are transmitted in “little endian”.

4.4.8 MESH/NODE/MPS/BV-09-C [Unprovisioned Device Sends and Receives Complete Provisioning PDUs]

• Test Purpose

Verify that an unprovisioned device IUT can send and receive complete Provisioning PDUs within a single Handle Value Notification ATT PDU.

• Reference

[3] Section 3.3.2, 5.2.2, 5.3.1, 5.3.3, 7.1.2, 7.1.3

• Initial Condition

- The IUT is an unprovisioned device.
- The IUT is running the Mesh Provisioning Service.
- The Lower Tester is a Provisioner.
- The IUT is in State: Node PB-GATT Link Established after execution of setup preamble 4.2.3.

• Test Procedure

1. The Lower Tester writes to the Mesh Provisioning Data In characteristic exposed by the IUT to send a single Proxy PDU containing a complete Provisioning PDU in a single Handle Value Notification smaller than the negotiated ATT_MTU to the IUT (e.g., by executing the procedure in GATT.TS [5] GATT/SR/GAW/BV-01-C with the handle of the Mesh Provisioning Data In characteristic and the value containing a properly formatted Proxy PDU containing a Provisioning PDU). The SAR field is set to 0b00 (Complete higher level message), the MessageType field is set to 0x03 (Provisioning PDU), and the Data field contains the Provisioning PDU Parameters for the Provisioning PDU sent.

2. The IUT receives the message.

3. The Upper Tester orders the IUT to write to the Mesh Provisioning Data Out characteristic exposed by the Lower Tester to send a Provisioning PDU smaller than the negotiated ATT_MTU (e.g., by executing the procedure in GATT.TS [5] GATT/CL/GAW/BV-01-C). The Proxy PDU containing the Provisioning PDU has the SAR field set to 0b00 (“Complete higher level message”), the MessageType field set to 0x03 (“Provisioning PDU”), and a complete proxy message to the Lower Tester.
4. The Lower Tester receives a Proxy Protocol message from the IUT with the exact data contained in the Proxy Protocol message sent from the IUT.

- **Expected Outcome**
  
  **Pass verdict**

  In step 2, the IUT receives a single Proxy PDU message in one Handle Value Notification ATT PDU.

  All values received in the Handle Value Notification and Proxy PDU are formatted correctly and contain valid values.

  In step 3, The IUT sends the message using one Handle Value Notification ATT PDU containing a single Proxy message to the Lower Tester with SAR field set to b00 ("Complete higher level message"), RFU set to zero, MessageType equal to 0x3 ("Provisioning PDU"), and the complete Proxy message.

**4.4.9 MESH/NODE/MPS/BV-10-C [Unprovisioned Device Sends and Receives Segmented Provisioning PDU Message]**

- **Test Purpose**
  
  Verify that an unprovisioned device IUT can send and receive a Provisioning PDU that is segmented into multiple Proxy PDUs when the required Proxy PDU size is larger than the negotiated ATT_MTU.

- **Reference**
  
  [3] Section 5.2.2, 5.3.3, 5.4.1, 7.1.3

- **Initial Condition**
  
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].
  - The IUT is in State: Node PB-GATT Link Established after execution of setup preamble 4.2.3.

- **Test Procedure**
  
  1. The Lower Tester writes to the Mesh Provisioning Data In characteristic exposed by the IUT to send a Provisioning PDU that is larger than the negotiated ATT_MTU to the IUT. The Provisioning PDU is split into multiple Proxy PDUs and transmitted via multiple Handle Value Notification ATT PDUs.
  2. The IUT receives the complete Provisioning PDU and reassembles the fragments before processing the message. The Upper Tester receives confirmation from the IUT that the message was received.
  3. The Upper Tester orders the IUT to write to the Mesh Provisioning Data Out characteristic exposed by the Lower Tester to send a Provisioning PDU larger than the negotiated ATT_MTU.
  4. The Lower Tester receives the Provisioning PDU segmented in multiple Proxy PDUs. The first Proxy PDU has the SAR field set to b01 ("Data field contains the first segment of a message"), the MessageType field set to 0x03 ("Provisioning PDU"), and the Data field filled with the first segment of the Provisioning PDU message. Each intermediate Proxy PDU that is not the final segment has the SAR field set to 0b10, the MessageType field set to 0x03, and the Data field filled with each subsequent segment of the Provisioning PDU. The Proxy PDU containing the final
segment of the Provisioning PDU has the SAR field set to 0b11, the MessageType field set to 0x03, and the Data field containing the end of the Provisioning PDU message. (For example, Provisioning Public Key PDU. If necessary, start with Provisioning Invite PDU, Provisioning Capabilities PDU, Provisioning Start PDU as in the generic test procedure under Provisioning Procedure – Unprovisioned Device).

- **Expected Outcome**
  
  **Pass verdict**

  In step 2, the IUT receives the complete Provisioning PDU in multiple Handle Value Notification ATT PDUs.

  Each Proxy PDU received by the IUT is prepended with a 1-octet header containing the expected SAR value and MessageType of 0x03. The first fragment received has the SAR field set to 0b01, the last fragment received has the SAR field set to 0b11, and any intermediate fragments have the SAR field set to 0b10.

  In step 3, the IUT sends the complete Provisioning PDU segmented into multiple Proxy PDUs to the Lower Tester. Each Proxy PDU contains the expected fields and values. Each segment is sent using a Handle Value Notification ATT PDU.

**4.4.10 MESH/PVNR/MPS/BV-01-C [Advertising Unprovisioned State for GATT Provisioning – Provisioner]**

- **Test Purpose**
  
  Verify that a Provisioner IUT can receive Mesh Provisioning Service advertisements.

- **Reference**

  [3] Section 3.3.2, 5.2.2, 5.4.1, 5.4.2, 7.1.2.1, 7.1.2.2.1

- **Initial Condition**

  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The Lower Tester is in either Limited Discoverable mode or General Discoverable mode.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service.

- **Test Procedure**

  1. The Lower Tester advertises the Mesh Provisioning Service in connectable advertising PDUs, interleaved with Unprovisioned Device beacons. The connectable advertising PDUs shall contain the following AD Types: Flags, List of Service Class UUIDs, and Service Data. The List of Service Class UUIDs shall contain the Mesh Provisioning Service UUID. The Service Data field shall contain the Lower Tester’s 16-octet Device UUID.
  2. The Upper Tester expects the IUT to report the discovery of an unprovisioned device with the Lower Tester’s Device UUID.
• Expected Outcome
  Pass verdict

  The IUT reports the discovery of an unprovisioned device with the Lower Tester’s Device UUID to the Upper Tester.

4.4.11 MESH/PVNR/MPS/BV-02-C [PB-GATT Link Establishment and Link Close – Provisioner]

• Test Purpose
  Verify that a Provisioner IUT can establish a PB-GATT provisioning bearer link with an unprovisioned device. Verify the IUT sends correctly formatted PB-GATT packets.

• Reference
  [3] Section 3.3.2, 5.2.2

• Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester emulates an unprovisioned device.
  - The IUT has been triggered to provision the Lower Tester.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service.

• Test Procedure
  1. Execute the steps of Setup Preamble 4.2.5.
  2. The Lower Tester advertises the Mesh Provisioning Service in connectable advertising PDUs, interleaved with Unprovisioned Device beacons.
  3. The IUT is induced to discover the handles for the Mesh Provisioning Data In characteristic and the Mesh Provisioning Data Out characteristic exposed by the Mesh Provisioning Service on the Lower Tester by executing the procedure in GATT.TS [5] Discover All Characteristics of a Service – by Server, GATT/SR/GAD/BV-04-C.
  4. The IUT is induced to send an ATT_Write_Request to the Lower Tester with the handle of the Mesh Provisioning Data Out characteristic and the value set to 0x0001 to enable notifications.
  5. The Lower Tester terminates the ATT bearer connection.

• Expected Outcome
  Pass verdict

  The IUT successfully establishes an ATT bearer connection with the Lower Tester, discovers the Provisioning Data characteristic, and configures it to enable notifications.

  The IUT receives Mesh Provisioning Service advertisements from the Lower Tester containing the Lower Tester’s Device UUID, interleaved with Unprovisioned Device beacons.

  The ATT bearer connection is terminated between the IUT and the Lower Tester.
4.4.12 Mesh Provisioning Service Characteristic Discovery by Provisioner

This test group is for generic use and contains one or more test cases to verify compliant operation in response to characteristic discovery by an IUT acting as a Provisioner. The verification is done one value at a time, as enumerated in the test cases in Table 4.3, using this generic test procedure.

- **Test Purpose**
  Verify that a Provisioner IUT can discover characteristics exposed by the Mesh Provisioning Service on an unprovisioned device.

- **Reference**
  [3] Section 3.3.2, 7.1.3.1, 7.1.3.2

- **Initial Condition**
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service including all defined characteristics.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The IUT has previously discovered the Mesh Provisioning Service and has saved the handle range for an instantiation of the Mesh Provisioning Service that exposes the characteristic specified in Table 4.3 by executing the procedure in GATT.TS [5] GATT/CL/GAD/BV-02-C with the service UUID of «Mesh Provisioning Service» or by other means.

- **Test Procedure**
  1. The Upper Tester orders the IUT to execute either of the procedures included in GATT.TS [5]: Discover All Characteristics of a Service – by Client, GATT/CL/GAD/BV-04-C, with the specified handle range for the instantiation of the Mesh Provisioning Service, or Discover Characteristic by UUID, GATT/CL/GAD/BV-05-C, with the specified handle range for the instantiation of the Mesh Provisioning Service and UUID set to the value in Table 4.3. In the selected procedure, only one pass is needed with the server database defined in Initial Condition.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Characteristic UUID</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.12.1MESH/PVNR/MPS/BV-03-C [Discover Mesh Provisioning Data Out Characteristic - Provisioner]</td>
<td>«Mesh Provisioning Data Out»</td>
</tr>
<tr>
<td>4.4.12.2MESH/PVNR/MPS/BV-04-C [Discover Mesh Provisioning Data In Characteristic]</td>
<td>«Mesh Provisioning Data In»</td>
</tr>
</tbody>
</table>

*Table 4.3: Mesh Provisioning Service Characteristic Discovery by Provisioner*
4.4.13 MESH/PVNR/MPS/BV-05-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor - Provisioner]

- **Test Purpose**
  Verify that a Provisioner IUT can discover the Mesh Provisioning Data Out Client Characteristic Configuration descriptor on an unprovisioned device.

- **Reference**
  [3] Section 3.3.2, 7.1.2, 7.1.3

- **Initial Condition**
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service in which the Mesh Provisioning Data Out characteristic and its associated Client Characteristic Configuration descriptor are exposed.
  - The IUT has discovered the handle range of the Mesh Provisioning Data Out characteristic, by executing the procedure in MESH/PVNR/MPS/BV-03-C [Discover Mesh Provisioning Data Out Characteristic - Provisioner] or by other means.

- **Test Procedure**
  The Upper Tester orders the IUT to execute one pass of the procedure included in GATT.TS [5]: Discover all Characteristic Descriptors, GATT/CL/GAD/BV-06-C using the specified handle range for Mesh Provisioning Data Out, with the server database defined in Initial Condition.

- **Expected Outcome**
  Pass verdict
  One attribute handle/value pair is returned containing the UUID «Client Characteristic Descriptor».
  The Lower Tester does not require authentication or encryption to access the characteristic descriptor.

4.4.14 MESH/PVNR/MPS/BV-06-C [Enable Notifications for Mesh Provisioning Data Out - Provisioner]

- **Test Purpose**
  Verify that a Provisioner IUT can enable notifications to the Mesh Provisioning Data Out Client Characteristic Configuration Descriptor.
Mesh Profile (MESH) / Test Suite

- **Reference**
  [3] Section 3.3.2, 7.1.2, 7.1.3

- **Initial Condition**
  - The Lower Tester has one instantiation of the Mesh Provisioning Service in which the Mesh Provisioning Data Out characteristic and its associated Client Characteristic Configuration descriptor are exposed.
  - The handle of the Mesh Provisioning Data Out characteristic has been previously discovered by the IUT by executing the test procedure in MESH/PVNR/MPS/BV-03-C [Discover Mesh Provisioning Data Out Characteristic - Provisioner] or is known to the IUT by other means.
  - The handle of the Client Characteristic Configuration descriptor has been previously discovered by the IUT by executing the test procedure in MESH/PVNR/MPS/BV-05-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor - Provisioner] or is known to the IUT by other means.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

- **Test Procedure**
  1. The Upper Tester orders the IUT to send a correctly formatted ATT_Write_Request (Code = 0x12) with the handle of the characteristic descriptor and a value of 0x0001 to the Lower Tester.

- **Expected Outcome**
  **Pass verdict**
  The characteristic descriptor is successfully written.

**4.4.15 MESH/PVNR/MPS/BV-08-C [Provisioner Sends and Receives Complete Provisioning PDUs]**

- **Test Purpose**
  Verify that a Provisioner IUT can send and receive provisioning data contained in a single Provisioning PDU.

- **Reference**
  [3] Section 3.2.2, 5.2.2, 5.3.1, 5.3.3, 7.1.3

- **Initial Condition**
  - The IUT is a Provisioner.
  - The IUT is running the Mesh Provisioning Service.
  - The Lower Tester is an unprovisioned device.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service including all defined characteristics.
  - The IUT is in State: Provisioner PB-GATT Link Established after execution of setup preamble 4.2.4.
• **Test Procedure**

1. The Lower Tester writes to the Mesh Provisioning Data Out characteristic exposed by the IUT to send a Provisioning PDU smaller than the negotiated ATT_MTU (e.g., by executing the procedure in GATT.TS [5] GATT/CL/GAW/BV-01-C, Write Without Response – by Client with the handle of the Mesh Provisioning Data In characteristic and the value containing a properly formatted Proxy PDU containing a Provisioning PDU). The Proxy PDU containing the Provisioning PDU has the SAR field set to 0b00 (“Complete higher level message”), the MessageType field set to 0x03 (“Provisioning PDU”), and the Data field set to the complete Provisioning PDU message.

2. The Upper Tester confirms that the IUT receives the message from the Lower Tester.

3. The Upper Tester orders the IUT to write to the Mesh Provisioning Data In characteristic exposed by the Lower Tester to send a single Proxy PDU containing a complete Provisioning PDU in a single Handle Value Notification. The SAR field is set to 0b00 (Complete higher level message), the MessageType field is set to 0x03 (Provisioning PDU), and the Data field contains the Provisioning PDU Parameters for the Provisioning PDU sent.

4. The Lower Tester receives the complete message from the IUT in a single PDU.

• **Expected Outcome**

**Pass verdict**

In step 2, the IUT receives a single Proxy PDU containing a complete Provisioning PDU from the Lower Tester with the expected values.

In step 3, the IUT sends a single Proxy PDU containing the entire Provisioning PDU to the Lower Tester in one Handle Value Notification ATT PDU. All fields are formatted correctly and contain valid values.

4.4.16 MESH/PVNR/MPS/BV-09-C [Provisioner Sends and Receives Segmented Message]

• **Test Purpose**

Verify that a Provisioner IUT can send and receive a message when the complete message does not fit into one PDU.

• **Reference**

[3] Section 5.2.2, 5.3.3, 5.4.1, 7.1.3

• **Initial Condition**

- The IUT is a Provisioner.
- The Lower Tester is an unprovisioned device.
- The IUT is in State: Provisioner PB-GATT Link Established (4.2.4).

• **Test Procedure**

1. The Lower Tester writes to the Mesh Provisioning Data Out characteristic exposed by the IUT to send a Provisioning PDU larger than the negotiated ATT_MTU. The Provisioning PDU is segmented into multiple Proxy PDUs. The first Proxy PDU sent has the SAR field set to b01 (“Data field contains the first segment of a message”), the MessageType field set to 0x3 (“Provisioning PDU”), and the Data field containing the first segment of the Provisioning PDU
message. Each intermediate Proxy PDU that is not the final segment has the SAR field set to 0b10, the MessageType field set to 0x03, and the Data field filled with each subsequent segment of the Provisioning PDU. The Proxy PDU containing the final segment of the Provisioning PDU has the SAR field set to 0b11, the MessageType field set to 0x03, and the Data field containing the end of the Provisioning PDU message. (For example, Provisioning Public Key PDU. If necessary, start with Provisioning Invite PDU, Provisioning Capabilities PDU, Provisioning Start PDU as in the generic test procedure under Provisioning Procedure – Provisioner).

2. The Upper Tester confirms that the IUT receives all Proxy PDUs from the Lower Tester and that the reassembled message is valid.

3. The Upper Tester orders the IUT to write to the Mesh Provisioning Data In characteristic exposed by the Lower Tester to send a Provisioning PDU message larger than the negotiated MTU.

4. The Lower Tester receives all Proxy PDUs and reassembles the expected message from the segments.

- Expected Outcome

**Pass verdict**

In step 2, the IUT receives all Proxy PDUs containing the segmented Provisioning PDU message and reassembles the expected message.

In step 3, the IUT sends the complete Provisioning PDU message segmented among multiple Proxy PDUs. The initial Proxy PDU has the SAR field set to 0b01 ("Data field contains the first segment of a message"), the MessageType field set to 0x03 ("Provisioning PDU"), and the first segment of the Provisioning PDU message. Each subsequent Proxy PDU that does not contain the final segment has the SAR field set to 0b10, the MessageType field set to 0x03, and the Data field filled with that segment of the Provisioning PDU message. The final Proxy PDU has the SAR field set to 0b11, the MessageType field set to 0x03, and the Data field containing the end of the Provisioning PDU message.

4.4.17 MESH/NODE/MPS/BI-01-C [Mesh Provisioning Data In Characteristic Only Supports Proxy PDU containing Provisioning PDU]

- Test Purpose

Verify that the Mesh Provisioning Data In characteristic only supports Proxy PDU type messages that contain Provisioning PDUs and does not support other Proxy PDU type messages.

- Reference

[3] 7.1.3.1.1

- Initial Condition

- The IUT is an unprovisioned device.
- The IUT is running the Mesh Provisioning Service.
- The Lower Tester is a Provisioner.
- The IUT is in State: Node PB-GATT Link Established (4.2.3).
• Test Procedure

The procedure in GATT.TS [5] GATT/SR/GAW/BV-01-C, Write Without Response – to Server, is performed with the handle of the Mesh Provisioning Data In characteristic and the value containing a properly formatted Proxy PDU type that does not contain a Provisioning PDU.

• Expected Outcome
Pass verdict

The IUT does not accept the Proxy PDU written to the Mesh Provisioning Data In characteristic.

4.5 Provisioning Procedure

The test group objective is to verify the provisioning procedure. Provisioning relies on a common provisioning protocol and common procedures that may be transported by multiple different provisioning bearers. To leverage the common aspects of these procedures, the bearer-specific test cases have been isolated from the common procedures.

This section contains tests which are common to all provisioning bearers and that test the overall provisioning protocol and procedures used over all supported bearers. These tests are written to be bearer-agnostic, and are executed using each supporter bearer for the IUT.

4.5.1 Provisioning Procedure – Unprovisioned Device

• Test Purpose
This is a generic procedure to verify the behavior of an unprovisioned device in the Provisioning procedure. The verification is done for one combination of authentication values at a time, as enumerated in the test cases in Table 4.4, using this generic procedure.

• Reference
[3] Section 5.1, 5.3.3, 5.4.1, 5.4.2, 5.4.3

• Initial Condition
- The IUT is an unprovisioned device.
- The Lower Tester is a Provisioner.
- Field values of the Provisioning Capabilities PDU are specified in the IXIT [6].

• Test Procedure
1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
3. The Lower Tester receives a Provisioning Capabilities PDU (PDU Type 0x01) in response from the IUT. The Number of Elements field is set to a value between 0x01-0xFF and the Algorithms
field has bit 0 ("FIPS-P256 Elliptic Curve") set to 1 and all other bits set to 0. The rest of the fields are set as follows:

a. Public Key Type field bit 0 is set to 0 (zero) if Public Key Available OOB is "No", and set to 1 otherwise.

b. If the Authentication Method is:
   i. Output OOB, then the Static OOB Type field bits are set to 0, the Output OOB Size field is set to a random non-RFU value greater than 0x00, the Output OOB Action field bit is set to a randomly selected action supported by the IUT, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.
   ii. Input OOB, then the Static OOB Type field bits are set to 0, the Output OOB Size set to 0x00, the Output OOB Action set to 0x00, the Input OOB Size set to a non-RFU value greater than 0x00 as supplied in the IXIT [6], and the Input OOB Action field bit is set as supplied in the IXIT [6].
   iii. Static OOB, then the Static OOB Type field bit 0 is set to 1, the Output OOB Size set to 0x00, the Output OOB Action field bits are set to 0, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.
   iv. No OOB, then the Static OOB Type field bits are set to 0, the Output OOB Size set to 0x00, the Output OOB Action set to 0x00, the Input OOB Size field bits are set to 0, and the Input OOB Action field bits are set to 0.

4. The Lower Tester sends a Provisioning Start PDU (PDU Type 0x02) with the Algorithm field value set to 0x00, and the rest of the field values are as follows:

a. The Public Key field set to 0x00 if Public Key Available OOB is "No" and set to 0x01 otherwise.

b. If the Authentication Method is:
   i. "No OOB", then the Authentication Method field is set to 0x00, the Authentication Action field is set to 0x00, and the Authentication Size field is set to 0x00.
   ii. "Static OOB", then the Authentication Method field is set to 0x01, the Authentication Action field is set to 0x00, and the Authentication Size field is set to 0x00.
   iii. "Output OOB", then the Authentication Method field is set to 0x02, the Authentication Action field is set to a value between 0x00-0x04 as declared in step 3, and the Authentication Size field is set to a value between 0x01-0x08 as declared in step 3.
   iv. "Input OOB", then the Authentication Method field is set to 0x03, the Authentication Action field set to a value between 0x00-0x04 as declared in step 3, and the Authentication Size field set to a value between 0x01-0x08 as declared in step 3.

5. The Lower Tester sends a Provisioning Public Key PDU (PDU Type 0x03) to the IUT with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm, and expects a Provisioning Public Key PDU in the response containing the IUT’s public key, with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm.

6. If the Authentication Method is:

   a. "Input OOB", then the Lower Tester generates a random number and displays/reports this value so it can be entered on the IUT. The value is entered into the IUT using its
Input OOB mechanism. The Lower Tester expects a Provisioning Input Complete PDU (PDU Type 0x04) from the IUT once input has been completed.

b. “Output OOB”, then the IUT is expected to internally generate a random number and utilize its Output OOB mechanism to communicate this generated value. This value is inputted into the Lower Tester.

c. “Static OOB”, then no action is taken in this step.

d. “No OOB”, then no action is taken in this step.

7. The Lower Tester sends a Provisioning Confirmation PDU (PDU Type 0x05) to the IUT with the Confirmation field set to the calculated confirmation value. The Lower Tester expects a Provisioning Confirmation PDU in response containing the confirmation value calculated by the IUT.

8. The Lower Tester sends a Provisioning Random PDU (PDU Type 0x06) to the IUT containing the random value and expects to receive a Provisioning Random PDU in response from the IUT.

9. The Lower Tester authenticates the IUT based on the received data. If the authentication fails, the test case ends with a fail verdict and the Lower Tester terminates the provisioning bearer link.

10. The Lower Tester calculates the session key and sends a Provisioning Data PDU (PDU Type 0x07) to the IUT with the Encrypted Provisioning Data field containing an encrypted and authenticated network key, NetKey Index of the current NetKey, current value of the IV Index, and unicast address of the primary Element, and the Provisioning Data MIC field containing a PDU Integrity Check value. The Lower Tester expects a Provisioning Complete PDU (PDU Type 0x08) from the IUT in response.

11. The Lower Tester terminates the provisioning bearer link on receipt of the Provisioning Complete PDU.

12. If PB-GATT was used as provisioning bearer, then after the provisioning bearer link has been terminated, the Lower Tester expects the IUT to start advertising with Node Identity.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Public Key Available OOB</th>
<th>Authentication Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5.1.1 MESH/NODE/PROV/BV-01-C [Provisioning Procedure, No Public Key OOB, Output OOB Authentication]</td>
<td>No</td>
<td>Output OOB</td>
</tr>
<tr>
<td>4.5.1.2 MESH/NODE/PROV/BV-02-C [Provisioning Procedure, No Public Key OOB, Input OOB Authentication]</td>
<td>No</td>
<td>Input OOB</td>
</tr>
<tr>
<td>4.5.1.3 MESH/NODE/PROV/BV-03-C [Provisioning Procedure, No Public Key OOB, Static OOB Authentication]</td>
<td>No</td>
<td>Static OOB</td>
</tr>
<tr>
<td>4.5.1.4 MESH/NODE/PROV/BV-04-C [Provisioning Procedure, Public Key OOB, Output OOB Authentication]</td>
<td>Yes</td>
<td>Output OOB</td>
</tr>
<tr>
<td>4.5.1.5 MESH/NODE/PROV/BV-05-C [Provisioning Procedure, Public Key OOB, Input OOB Authentication]</td>
<td>Yes</td>
<td>Input OOB</td>
</tr>
<tr>
<td>Test Case</td>
<td>Public Key Available OOB</td>
<td>Authentication Method</td>
</tr>
<tr>
<td>-------------------------------------------------------------------</td>
<td>--------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>4.5.1.6 MESH/NODE/PROV/BV-06-C [Provisioning Procedure, Public Key OOB, Static OOB Authentication]</td>
<td>Yes</td>
<td>Static OOB</td>
</tr>
<tr>
<td>4.5.1.7 MESH/NODE/PROV/BV-07-C [Provisioning Procedure, No Public Key OOB, No OOB Authentication]</td>
<td>No</td>
<td>No OOB</td>
</tr>
<tr>
<td>4.5.1.8 MESH/NODE/PROV/BV-08-C [Provisioning Procedure, Public Key OOB, No OOB Authentication]</td>
<td>Yes</td>
<td>No OOB</td>
</tr>
</tbody>
</table>

*Table 4.4: Provisioning Procedure - Unprovisioned Device Test Cases*

- **Expected Outcome**

  **Pass verdict**

  The IUT and the Lower Tester successfully establish a provisioning bearer link.

  The IUT sends a correctly formatted Provisioning Capabilities PDU with all valid values.

  The Lower Tester successfully authenticates the IUT using the selected authentication method.

  If the Output OOB Action is Blink, Beep, or Vibrate (bits 0, 1, or 2 is set), the IUT outputs a random number as the selected sequence of events with a gap of at least 3 seconds between sequences. If the Output OOB Action is Output Numeric (bit 3 is set), the value is output using digits. If the Output OOB Action is Output Alphanumeric (bit 4 is set), the value is output using ASCII digits and uppercase letters.

  If the public key is not available via an OOB mechanism, the IUT sends a correctly formatted Provisioning Public Key PDU with valid values upon receipt of a Provisioning Public Key PDU from the Lower Tester.

  Upon receipt of the Lower Tester’s Public Key, the IUT calculates the ECDHSecret.

  The IUT sends a correctly formatted Provisioning Confirmation PDU with valid values upon receipt of a Provisioning Confirmation PDU from the Lower Tester.

  The IUT sends a correctly formatted Provisioning Random PDU with valid values to the Lower Tester upon receipt of a Provisioning Random PDU from the Lower Tester.

  The IUT sends a Provisioning Complete PDU (PDU Type 0x08) upon receipt of the Provisioning Data PDU for both values of Network Key and NetKey Index.

  All multiple-octet numeric values in Provisioning PDUs are sent as “big endian”.

  All RFU bits are ignored for each field in each Generic Provisioning PDU.
Each Generic Provisioning PDU is sent after a random delay between 20 ms and 50 ms.

If PB-ADV is used, the IUT increases the value of the Transaction Number field of the PB-ADV PDU by one for each new Provisioning PDU it sends.

In step 12, if PB-GATT was used as provisioning bearer, the IUT starts advertising with Node Identity, and the Lower Tester verifies that the advertising data meets the requirements from the Pass Verdict of MESH/SR/MPXS/BV-09-C [Advertise Mesh Proxy Service with Node Identity].

### 4.5.2 MESH/NODE/PROV/BV-09-C [Provisioning Procedure, Procedure Timeout]

- **Test Purpose**
  Verify that an unprovisioned device IUT can successfully respond to a provisioning procedure after experiencing a procedure timeout.

- **Reference**
  [3] Section 5.3.3, 5.4.1, 5.4.2, 5.4.4

- **Initial Condition**
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT are specified by the manufacturer in the IXIT [6].

- **Test Procedure**
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
  2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
  3. The Lower Tester expects a Provisioning Capabilities PDU (PDU Type 0x01) in response from the IUT.
  4. The Lower Tester sends a Provisioning Start PDU (PDU Type 0x02) selecting any method supported by the IUT.
  5. The Lower Tester does not respond further to the IUT and wait for the duration of the provisioning procedure timeout.
  6. After a period equal to 2 times the provisioning timeout has passed, the Lower Tester attempts to reestablish the provisioning bearer and initiate provisioning procedure by repeating steps 1-3.
  7. Lower Tester confirms that the IUT can respond to a new provisioning attempt following procedure timeout.

- **Expected Outcome**
  **Pass verdict**

  The IUT and the Lower Tester successfully establish a provisioning bearer link.

  The IUT sends a Provisioning Capabilities PDU that is correctly formatted with valid values.
The IUT and the Lower Tester successfully re-establish the provisioning bearer link after the provisioning procedure timeout.

### 4.5.3 MESH/NODE/PROV/BV-10-C [Unprovisioned Device Sends Provisioning Failed Message on Provisioning Error]

- **Test Purpose**
  Verify that an unprovisioned device IUT sends a Provisioning Failed PDU when it encounters a provisioning error in the provisioning protocol and waits for the Provisioner to close the provisioning bearer.

- **Reference**
  [3] Section 5.3.3, 5.4.1, 5.4.2, 5.4.4

- **Initial Condition**
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT are specified by the manufacturer in the IXIT [6].

- **Test Procedure**
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
  2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
  3. The Lower Tester receives a Provisioning Capabilities PDU in response.
  4. The Lower Tester sends a Provisioning Start PDU (PDU Type 0x02) to the IUT. The Algorithm field is set to 0x00 (FIPS P-256), Public Key is set to 0x00 (“Public key is not available via OOB”), Output OOB set to 0x00, the Input OOB Size set to a non-RFU, and Static OOB set to 0x00.
  5. The Lower Tester sends any valid Provisioning PDU to the IUT, other than the Provisioning Public Key PDU.
  6. The Lower Tester expects a Provisioning Failed PDU (PDU Type 0x09) from the IUT with the Value field set to 0x03 (“Unexpected PDU”).
  7. The Lower Tester sends a valid Provisioning Public Key PDU to the IUT.
  8. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 (“Unexpected PDU”).
  9. The Lower Tester sends a valid Provisioning Invite PDU to the IUT.
  10. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 (“Unexpected PDU”).
  11. The Lower Tester disconnects the provisioning bearer.
  12. Repeat steps 1–4, then continue the provisioning protocol to the point where the Lower Tester needs to send the Provisioning Random PDU.
  13. The Lower Tester sends a Provisioning Random PDU containing an incorrect Provisioner Random value, i.e., a value different than the one used to compute the Provisioner Confirmation value sent in the Provisioning Confirmation PDU in the previous step.
  14. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x04 (“Confirmation Failed”).
15. The Lower Tester sends a Provisioning Random PDU containing the correct Provisioner Random value, i.e., the same value used to compute the Provisioner Confirmation value sent in the Provisioning Confirmation PDU in the previous step.
16. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 ("Unexpected PDU").
17. The Lower Tester sends a valid Provisioning Invite PDU to the IUT.
18. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 ("Unexpected PDU").
19. The Lower Tester disconnects the provisioning bearer.
20. Repeat steps 1–4, then continue the provisioning protocol to the point where the Lower Tester needs to send the Provisioning Data PDU.
21. The Lower Tester sends a Provisioning Data PDU containing an incorrect Provisioning Data MIC field value.
22. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x06 ("Decryption Failed").
23. The Lower Tester sends a Provisioning Data PDU containing the correct Provisioning Data MIC field value.
24. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 ("Unexpected PDU").
25. The Lower Tester disconnects the provisioning bearer.

• Expected Outcome

Pass verdict

The IUT sends a Provisioning Failed PDU in with the appropriate error code whenever a protocol error occurs.

Notes:

Each Generic Provisioning PDU is sent after a random delay between 20 ms and 50 ms.

4.5.4 MESH/NODE/PROV/BV-11-C [Provisioned Node Updates Security Flags Based on the Provisioning Data PDU]

• Test Purpose

Verify that an IUT that has just been provisioned correctly updates the IV Update procedure state and the Key Refresh Phase based on the Flags field of the Provisioning Data PDU.

• Reference

[3] Section 5.3.3, 5.4.1.8, 5.4.2.5

• Initial Condition

- The IUT is an unprovisioned device.
- The Lower Tester is a Provisioner.
- The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].
• Test Procedure

1. The Lower Tester provisions the IUT by executing any of the procedures described in Provisioning Procedure – Unprovisioned Device. When sending the Provisioning Data PDU, the Lower Tester sets the Key Refresh flag and the IV Update flag of the Flags field to 1.
2. The Lower Tester disconnects the provisioning bearer.
3. The Upper Tester orders the IUT (or the IUT is triggered) to send a Secure Network beacon containing the IUT’s security flags.
4. The Lower Tester expects the IUT to send a Secure Network beacon containing the current IV Index, with the IV Update flag and the Key Refresh flag set to 1.

• Expected Outcome

Pass verdict

In step 4, the IUT sends the Secure Network beacon with the IV Update flag and the Key Refresh flag set to 1.

4.5.5 MESH/NODE/PROV/BV-12-C [Generate New Public Key on Each Provisioning Session]

• Test Purpose

Verify that the IUT, as an unprovisioned device, generates a new private-public key pair for each new provisioning session before sending the public key over the air.

• Reference

[3], [7] Section 5.4.1, 5.4.2, 5.4.3, 5.4.4

• Initial Condition

- The IUT is an unprovisioned device with the Device UUID specified in the IXIT [6].
- The Lower Tester is a Provisioner.
Test Procedure

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
2. The Lower Tester sends a Provisioning Invite PDU to the IUT, with the Attention Duration field set to any value.
3. The Lower Tester expects the IUT to respond with a Provisioning Capabilities PDU with valid fields.
4. The Lower Tester sends a Provisioning Start PDU with the Public Key field set to 0x00 (No OOB Public Key is used), and all other fields set to valid values.
5. The Lower Tester sends a Provisioning Public Key PDU with valid values.
6. The Lower Tester expects the IUT to send a Provisioning Public Key PDU with valid values. The Lower Tester saves the public key of the IUT.
7. The Lower Tester successfully completes the provisioning protocol with the IUT, then the IUT is reset to the unprovisioned state.
8. Repeat steps 1–6, expecting the IUT to send a public key value different than the one received in the previous iteration. The Lower tester saves the new public key of the IUT.
9. The Lower Tester fails the provisioning protocol before it is completed.
10. Repeat steps 1–6, expecting the IUT to send a public key value different than the ones received in the previous iterations.
11. The Lower Tester terminates the provisioning session.

Figure 4.1: Test Procedure for MESH/NODE/PROV/BV-12-C [Generate New Public Key on Each Provisioning Session]
• Expected Outcome

Pass verdict

In each new provisioning session, the IUT sends a public key that is different than the public keys sent in previous sessions.

4.5.6 Provisioning Procedure – Provisioner

• Test Purpose

This is a generic procedure to verify the behavior of a Provisioner in the Provisioning procedure. The verification is done for one combination of authentication values at a time, as enumerated in the test cases in Table 4.5, using this generic procedure.

• Reference

[3] Section 5.1, 5.3.3, 5.4.1, 5.4.2, 5.4.3

• Initial Condition

- The IUT is a Provisioner.
- The Lower Tester is an unprovisioned device.
- Field values of the Provisioning Capabilities PDU are specified in the IXIT [6].

• Test Procedure

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.2 State: Provisioner PB-ADV Link Established, or for PB-GATT use 4.2.4 State: Provisioner PB-GATT Link Established).
2. The Upper Tester orders the IUT to send a Provisioning Invite PDU (PDU Type 0x00) to the Lower Tester with the Attention Duration set to a value greater than 0.
3. Upon receipt of the Provisioning Invite PDU, the Lower Tester sends a Provisioning Capabilities PDU (PDU Type 0x01) to the IUT. The Number of Elements field is set to a value between 0x01-0xFF and the Algorithms field has bit 0 (“FIPS-P256 Elliptic Curve”) set to 1 and all other bits set to 0. The rest of the fields are set as follows:
   a. Public Key Type field bit 0 is set to 0 (zero) if Public Key Available OOB is “No”, and set to 1 otherwise.
   b. If the Authentication Method is:
      i. Output OOB, then the Static OOB Type field bits are set to 0, the Output OOB Size field is set to a random non-RFU value greater than 0x00, the Output OOB Action field bit is set to a randomly selected action supported by the IUT, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.
      ii. Input OOB, then the Static OOB Type field bits are set to 0, the Output OOB Size set to 0x00, the Output OOB Action set to 0x00, the Input OOB Size set to a non-RFU value greater than 0x00 as supplied in the IXIT [6], and the Input OOB Action set to the value supplied in the IXIT [6].
      iii. Static OOB, then the Static OOB Type field bit 0 is set to 1, the Output OOB Size field is set to 0x00, the Output OOB Action field bits are set to 0, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.
iv. No OOB, then the Static OOB Type field bits are set to 0, the Output OOB Size set to 0x00, the Output OOB Action field bits are set to 0, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.

4. The IUT is induced to send a Provisioning Start PDU (PDU Type 0x02) to the Lower Tester with the Algorithm field value set to 0x00, and the rest of the field values are as follows:
   a. The Public Key field set to 0x00 if Public Key Available OOB is “No” and set to 0x01 otherwise.
   b. If the Authentication Method is:
      i. “No OOB”, then the Authentication Method field is set to 0x00, the Authentication Action field is set to 0x00, and the Authentication Size field is set to 0x00.
      ii. “Static OOB”, then the Authentication Method field is set to 0x01, the Authentication Action field is set to 0x00, and the Authentication Size field is set to 0x00.
      iii. “Output OOB”, then the Authentication Method field is set to 0x02, the Authentication Action field is set to a value between 0x00-0x04 as declared in step 3, and the Authentication Size field is set to a value between 0x01-0x08 as declared in step 3.
      iv. “Input OOB”, then the Authentication Method field is set to 0x03, the Authentication Action field set to a value between 0x00-0x03 as declared in step 3, and the Authentication Size field set to a value between 0x01-0x08 as declared in step 3.

5. The IUT is induced to send a Provisioning Public Key PDU (PDU Type 0x03) to the Lower Tester with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm, and expects a Provisioning Public Key PDU in response from the Lower Tester containing the Lower Tester’s public key, with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm.

6. If the Authentication Method is:
   a. “Input OOB”, then the IUT generates a random number and displays/reports this value so it can be entered on the Lower Tester. The value is entered into the Lower Tester using its Input OOB mechanism. The Lower Tester sends a Provisioning Input Complete PDU (PDU Type 0x04) to the IUT once input has been completed.
   b. “Output OOB”, then the Lower Tester is expected to internally generate a random number and utilize its Output OOB mechanism to communicate this generated value. This value is inputted into the IUT.
   c. “Static OOB”, then no action is taken in this step.
   d. “No OOB”, then no action is taken in this step and the value is zero.

7. The IUT is induced to send a Provisioning Confirmation PDU (PDU Type 0x05) to the Lower Tester with the Confirmation field set to any values exchanged between IUT and the Lower Tester based on the OOB Authentication method. The Lower Tester sends a Provisioning Confirmation PDU in response containing the confirmation value calculated by the Lower Tester.

8. The IUT is induced to send a Provisioning Random PDU (PDU Type 0x06) to the Lower Tester containing the random value and expects to receive a Provisioning Random PDU in response from the Lower Tester.

9. The IUT authenticates the Lower Tester based on the received data. If the authentication fails, the test case ends with a fail verdict and the IUT terminates the provisioning bearer link.
10. The IUT calculates the session key and sends a Provisioning Data PDU (PDU Type 0x07) to the Lower Tester with the Encrypted Provisioning Data field containing an encrypted and authenticated network key, NetKey Index of the current NetKey, the IV Index field set to 0x00000000, and unicast address of the primary Element, and the Provisioning Data MIC field containing a PDU Integrity Check value. Upon receipt of the Provisioning Data PDU, the Lower Tester sends a Provisioning Complete PDU (PDU Type 0x08) to the IUT.

11. The IUT terminates the provisioning bearer link on receipt of the Provisioning Complete PDU.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Public Key Available OOB</th>
<th>Authentication Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5.6.1 MESH/PVNR/PROV/BV-01-C [Provisioning Procedure as Provisioner, No Public Key OOB, Output OOB Authentication]</td>
<td>No</td>
<td>Output OOB</td>
</tr>
<tr>
<td>4.5.6.2 MESH/PVNR/PROV/BV-02-C [Provisioning Procedure as Provisioner, No Public Key OOB, Input OOB Authentication]</td>
<td>No</td>
<td>Input OOB</td>
</tr>
<tr>
<td>4.5.6.3 MESH/PVNR/PROV/BV-03-C [Provisioning Procedure as Provisioner, No Public Key OOB, Static OOB Authentication]</td>
<td>No</td>
<td>Static OOB</td>
</tr>
<tr>
<td>4.5.6.4 MESH/PVNR/PROV/BV-04-C [Provisioning Procedure as Provisioner, Public Key OOB, Output OOB Authentication]</td>
<td>Yes</td>
<td>Output OOB</td>
</tr>
<tr>
<td>4.5.6.5 MESH/PVNR/PROV/BV-05-C [Provisioning Procedure as Provisioner, Public Key OOB, Input OOB Authentication]</td>
<td>Yes</td>
<td>Input OOB</td>
</tr>
<tr>
<td>4.5.6.6 MESH/PVNR/PROV/BV-06-C [Provisioning Procedure as Provisioner, Public Key OOB, Static OOB Authentication]</td>
<td>Yes</td>
<td>Static OOB</td>
</tr>
<tr>
<td>4.5.6.7 MESH/PVNR/PROV/BV-07-C [Provisioning Procedure as Provisioner, No Public Key OOB, No OOB Authentication]</td>
<td>No</td>
<td>No OOB</td>
</tr>
<tr>
<td>Test Case</td>
<td>Public Key Available OOB</td>
<td>Authentication Method</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td><strong>4.5.6.8 MESH/PVNR/PROV/BV-08-C</strong> [Provisioning Procedure as Provisioner, Public Key OOB, No OOB Authentication]</td>
<td>Yes</td>
<td>No OOB</td>
</tr>
</tbody>
</table>

*Table 4.5: Provisioning Procedure - Provisioner Test Cases*

- **Expected Outcome**

  **Pass verdict**

  The IUT and the Lower Tester successfully establish a provisioning bearer link.

  The IUT sends a correctly formatted Provisioning Invite PDU with valid values.

  The IUT sends a correctly formatted Provisioning Start PDU with valid values in response to the Provisioning Capabilities PDU.

  The IUT sends a Provisioning Public Key PDU to the Lower Tester and receives a Provisioning Public Key PDU from the Lower Tester in response. Upon receipt of the Lower Tester’s Public Key, the IUT calculates the ECDHSecret.

  The IUT sends a correctly formatted Provisioning Confirmation PDU with valid values to the Lower Tester and receives a Provisioning Confirmation PDU from the Lower Tester in response.

  The IUT sends a correctly formatted Provisioning Random PDU with valid values to the Lower Tester and receives a Provisioning Random PDU from the Lower Tester in response.

  The IUT successfully authenticates the Lower Tester using the selected authentication method.

  If the Input OOB Action is Push or Twist, the IUT selects a random number to input by the designated action.

  The IUT sends a correctly formatted Provisioning Data PDU with valid values to the Lower Tester and receives a Provisioning Complete PDU from the Lower Tester in response.

  The IUT successfully terminates the provisioning bearer link.

  All multiple-octet numeric values in Provisioning PDUs are sent as “big endian”.

  All RFU bits are ignored for each field in each Generic Provisioning PDU.

  Each Generic Provisioning PDU is sent after a random delay between 20 ms and 50 ms.

  If PB-ADV is used, the IUT increases the value of the Transaction Number field of the PB-ADV PDU by one for each new Provisioning PDU it sends.
4.5.7 MESH/PVNR/PROV/BV-09-C [Provisioning Procedure, Procedure Timeout – Provisioner]

- **Test Purpose**
  Verify that a Provisioner IUT can initiate the provisioning procedure after experiencing a procedure timeout in a previous attempt.

- **Reference**
  [3] Section 5.3.3, 5.4.1, 5.4.2, 5.4.4

- **Initial Condition**
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned node.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].

- **Test Procedure**
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.2 State: Provisioner PB-ADV Link Established, or for PB-GATT use 4.2.4 State: Provisioner PB-GATT Link Established).
  2. The IUT is induced to send a Provisioning Invite PDU with the Attention Duration set to a value greater than 0.
  3. The Lower Tester sends a Provisioning Capabilities PDU in response.
  4. The Lower Tester expects a Provisioning Start PDU selecting any method supported by the IUT.
  5. The Lower Tester expects a Provisioning Public Key PDU from the IUT.
  6. The Lower Tester does not respond further to the IUT and waits for the duration of the provisioning procedure timeout. After the timeout has expired, the IUT disconnects the provisioning bearer.
  7. After a period, equal to 2 times the provisioning timeout has passed, the IUT attempts to reestablish the provisioning bearer and initiate provisioning by repeating steps 1–3.
  8. Lower Tester confirms that the IUT can initiate a new provisioning attempt following procedure timeout.

- **Expected Outcome**
  Pass verdict
  - The IUT and the Lower Tester successfully establish a provisioning bearer link.
  - The IUT terminates the procedure when the timeout has expired.
  - The IUT does not transmit a Provisioning Failed PDU.
  - The IUT can reestablish a provisioning bearer and restart the provisioning procedure following procedure timeout.
4.5.8 MESH/PVNR/PROV/BV-10-C [Provisioner Disconnects Provisioning Bearer on Provisioning Protocol Failure from Unprovisioned Device]

• Test Purpose
  Verify that a Provisioner IUT disconnects the provisioning bearer when it encounters a provisioning protocol failure from an unprovisioned device.

• Reference
  [3] Section 5.2, 5.3.3, 5.4.1, 5.4.2, 5.4.3, 5.4.4

• Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an Unprovisioned Device.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].

• Test Procedure
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.2 State: Provisioner PB-ADV Link Established, or for PB-GATT use 4.2.4 State: Provisioner PB-GATT Link Established).
  2. The Upper Tester sends a command to the IUT to send a Provisioning Invite PDU (PDU Type 0x00) with the Attention Duration set to a value greater than 0.
  3. The Lower Tester sends a Provisioning Capabilities PDU (PDU Type 0x01) in response, indicating the Lower Tester supports No OOB authentication. The Number of Elements field is set to a value between 0x01-0xFF, the Algorithms field has bit 0 ("FIPS-P256 Elliptic Curve") set to 1 and all other bits set to 0, the Static OOB Type field bit 0 is set to 0, the Output OOB Size field is set to 0x00, the Output OOB Action field is set to 0x00, the Input OOB Size field is set to 0x00, the Input OOB Action field is set to 0x00.
  4. The Lower Tester expects a Provisioning Start PDU (PDU Type 0x02) with the Algorithm field value set to 0x00, the Public Key field set to 0x00 (No OOB Public Key), the Authentication Method field set to 0x00 (No OOB), the Authentication Action field set to 0x00, and the Authentication Size field set to 0x00.
  5. The Lower Tester expects a Provisioning Public Key PDU (PDU Type 0x03) from the IUT with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm.
  6. The Lower Tester sends any Provisioning PDU that does not follow the correct Provisioning procedure.
  7. The IUT disconnects the provisioning bearer on receipt of the Provisioning PDU sent in step 6.

• Expected Outcome
  Pass verdict

  In step 7, the IUT disconnects the provisioning bearer.
4.5.9 MESH/NODE/PROV/BI-01-C [Device Resets Attention Timer State when Provisioning Bearer Dropped]

- Test Purpose
  Verify that an unprovisioned device IUT resets the attention timer state to 0x00 when the provisioning bearer is dropped unexpectedly before the provisioning protocol is complete.

- Reference
  [3] Section 3.1.2, 3.9.2, 3.10.1, 5.4.1, 5.4.2.2

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester acts as a Provisioner.
  - The Device UUID for the IUT are specified by the manufacturer in the IXIT [6].

- Test Procedure
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
  2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
  3. The Lower Tester expects a Provisioning Capabilities PDU (PDU Type 0x01) in response from the IUT.
  4. The Lower Tester closes the link on the provisioning bearer.
  5. The Upper Tester verifies that the IUT sets the Attention Timer state of its primary Element to 0x00.

- Expected Outcome
  Pass verdict
  The IUT sets the Attention Timer state of its primary Element to 0x00.

4.5.10 MESH/NODE/PROV/BI-02-C [Reject Provisioning Start PDU with RFU values]

- Test Purpose
  Verify that an unprovisioned device IUT can reject a Provisioning Start PDU containing RFU values.

- Reference
  [3] Section 3.1.2, 3.9.2, 3.10.1, 5.4.1.3

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester acts as a Provisioner.
  - The Device UUID for the IUT are specified by the manufacturer in the IXIT [6].
• Test Procedure

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
3. The Lower Tester expects a Provisioning Capabilities PDU (PDU Type 0x01) in response from the IUT.
4. The Lower Tester sends a Provisioning Start PDU (PDU Type 0x02) with the Algorithm field set to any RFU value.
5. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.
6. Repeat steps 1–3.
7. The Lower Tester sends a Provisioning Start PDU with the Algorithm field set to 0x00 and the Public Key set to any RFU value.
8. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.
9. Repeat steps 1–3.
10. The Lower Tester sends a Provisioning Start PDU with the Algorithm field set to 0x00, the Public Key field set to 0x00, and the Authentication Method field set to any RFU value.
11. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.
12. Repeat steps 1–3.
13. The Lower Tester sends a Provisioning Start PDU with the Algorithm field set to 0x00, the Public Key field set to 0x01, the Authentication Method field set to 0x00, and the Authentication Action field set to any RFU value.
14. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.
15. Repeat steps 1–3.
16. The Lower Tester sends a Provisioning Start PDU with the Algorithm field set to 0x00, the Public Key field set to 0x01, the Authentication Method field set to 0x00, the Authentication Action field set to 0x00, and the Authentication Size field set to any RFU value.
17. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.

• Expected Outcome

Pass verdict

In steps 5, 8, 11, 14, and 17, the IUT sends a Provisioning Failed PDU with the appropriate error code.

4.5.11 MESH/NODE/PROV/BI-03-C [Reject Provisioning Start PDU with Inconsistent Values]

• Test Purpose

Verify that an unprovisioned device IUT rejects Provisioning Start PDUs with values that are inconsistent with its capabilities.
• Reference

[3] Section 5.4.1.10

• Initial Condition

- The IUT is an unprovisioned device.
- The Lower Tester acts as a Provisioner.
- The Device UUID of the IUT is specified in the IXIT [6].

![Diagram of provisioning process]

Figure 4.2: Test Procedure for MESH/NODE/PROV/BI-03-C

• Test Procedure

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State:
Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).

2. The Lower Tester sends a Provisioning Invite PDU to the IUT and expects the IUT to respond with a Provisioning Capabilities PDU.

3. The Lower Tester sends a Provisioning Start PDU with values that are valid according to the definition of the PDU (e.g., not RFU or Prohibited), but inconsistent with the capabilities indicated by the IUT. For example, if the Provisioning Capabilities PDU received in step 2 indicates that an OOB Public Key is present (bit 0 of the Public Key Type field is set to 1), then the Public Key field of the Provisioning Start PDU is set to 0x00, and vice versa.

4. The Lower Tester expects the IUT to send a Provisioning Failed PDU with the Error Code field set to Invalid Format.

5. The Lower Tester sends the same Provisioning Start PDU from step 3.

6. The Lower Tester expects the IUT to send a Provisioning Failed PDU with the Error Code field set to Unexpected PDU.

7. The Lower Tester sends a new Provisioning Start PDU with field values that are consistent with the IUT capabilities.

8. The Lower Tester expects the IUT to send a Provisioning Failed PDU with the Error Code field set to Unexpected PDU.

9. The Lower Tester sends a new Provisioning Invite PDU.

10. The Lower Tester expects the IUT to send a Provisioning Failed PDU with the Error Code field set to Unexpected PDU.

11. The Lower Tester closes the provisioning bearer link.

- **Expected Outcome**

  **Pass verdict**

  The IUT rejects the inconsistent Provisioning Start PDU with the Invalid Format error code.

  After sending the Provisioning Failed PDU, the IUT rejects any other PDU with the Unexpected PDU error code.

### 4.5.12 MESH/NODE/PROV/BI-13-C [Reject Invalid Public Keys]

- **Test Purpose**

  Verify that the IUT, as an unprovisioned device, detects and rejects an invalid public key during provisioning.

- **Reference**

  [3], [7] Section 5.4.1, 5.4.2, 5.4.3, 5.4.4

- **Initial Condition**

  - The IUT is an unprovisioned device with the Device UUID specified in the IXIT [6].
  - The Lower Tester is a Provisioner.
Test Procedure

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
2. The Lower Tester sends a Provisioning Invite PDU to the IUT, with the Attention Duration field set to any value, and expects the IUT to respond with a Provisioning Capabilities PDU with valid fields.
3. The Lower Tester sends a Provisioning Start PDU with parameters that are valid with respect to the IUT capabilities received in step 2.
4. The Lower Tester sends a Provisioning Public Key PDU containing invalid public key coordinates that are not on the P-256 curve. To obtain the invalid public key, the Lower Tester generates a valid public key, then flips a single bit from the Y-coordinate (in the Public Key Y field of the Provisioning Public Key PDU) before sending it over the air. The Lower Tester verifies that these new coordinates are not on the curve before sending them over the air; if the new coordinates happen to be valid, then the generation procedure is repeated.
5. The Lower Tester expects a Provisioning Failed PDU from the IUT.
6. The Lower Tester disconnects the provisioning bearer.
7. Repeat steps 1–6 using, in step 4, another invalid public key obtained by generating a valid public key and changing the Y coordinate to 0 (the Public Key Y field of the Provisioning Public Key PDU is set to all-zeroes).
8. Repeat steps 1–6 using, in step 4, the invalid public key coordinates (0, 0). Both the Public Key X and the Public Key Y fields of the Provisioning Public Key PDU are set to all-zeroes.
• **Expected Outcome**

**Pass verdict**

The IUT detects each invalid public key and sends a Provisioning Failed PDU on each iteration.

**4.5.13 MESH/PVNR/PROV/BI-14-C [Reject Invalid Public Keys]**

• **Test Purpose**

Verify that the IUT, as a Provisioner, detects and rejects an invalid public key during provisioning.

• **Reference**

[3], [7] Section 5.4.1, 5.4.2, 5.4.3, 5.4.4

• **Initial Condition**

- The IUT is a Provisioner.
- The Lower Tester is an unprovisioned device with the Device UUID specified in the IXIT [6].

• **Test Procedure**

![Test Procedure Diagram](image)

*Figure 4.4: Test Procedure for MESH/PVNR/PROV/BI-14-C [Reject Invalid Public Keys]*

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).

2. The IUT is induced to begin the provisioning procedure. The Lower Tester expects a Provisioning Invite PDU from the IUT, with the Attention Duration field set to any value, and responds with a Provisioning Capabilities PDU with any valid fields, indicating that a public key is not available OOB (i.e., setting bit 0 of the Public Key Type field to 0).
3. The Lower Tester expects a Provisioning Start PDU with parameters that are valid with respect to the capabilities sent in step 2, followed by a Provisioning Public Key PDU containing a valid public key. The Lower Tester saves the public key of the IUT.

4. The Lower Tester sends a Provisioning Public Key PDU containing invalid public key coordinates that are not on the P-256 curve. To obtain the invalid public key, the Lower Tester generates a valid public key, then flips a single bit from the Y-coordinate (in the Public Key Y field of the Provisioning Public Key PDU) before sending it over the air. The Lower Tester verifies that these new coordinates are not on the curve before sending them over the air; if the new coordinates happen to be valid, then the generation procedure is repeated.

5. The Lower Tester expects the IUT to disconnect the provisioning bearer.

6. Repeat steps 1–5 using, in step 4, another invalid public key obtained by generating a valid public key and changing the Y coordinate to 0 (the Public Key Y field of the Provisioning Public Key PDU is set to all-zeros).

7. Repeat steps 1–5 using, in step 4, the invalid public key coordinates (0, 0). Both the Public Key X and the Public Key Y fields of the Provisioning Public Key PDU are set to all-zeros.

• Expected Outcome

Pass verdict

The IUT detects each invalid public key and disconnects the provisioning bearer on each iteration.

In each repetition of step 3, the IUT sends a public key value different that in the previous provisioning session.

4.6 Secure Network Beacons

The test group objective is to verify the behavior of Secure Network beacons.

4.6.1 MESH/NODE/BCN/SNB/BV-01-C [Advertising Secure Network Beacons]

• Test Purpose

Verify that a Node IUT generates a valid Secure Network beacon from the current NetworkID and IV Index. Confirm that beacon is advertised and can be received by the Lower Tester. Confirm the advertising payload is properly formatted. Confirm that a non-connectable advertising packet type is utilized.

• Reference

[3] Section 3.3.1, 3.9.1, 3.9.3, 3.9.3.1

• Initial Condition

- The IUT is a Node.
- The Lower Tester is a Node.
- IUT and the Lower Tester share previously established network security credentials.

• Test Procedure

1. The Upper Tester orders the IUT to generates and advertise a Secure Network beacon with AD Type is set to «Mesh Beacon», the Beacon Type field set to 0x01 (Secure Network beacon), and the Flag AD Type is not included in Beacon Data (non-connectable).
2. The Lower Tester scans for and receives the advertising packets sent by the IUT.
3. The Lower Tester observes the advertisement by the IUT and validates the payload.

- **Expected Outcome**
  
  **Pass verdict**

  The IUT sends a Secure Network beacon with AD Type set to «Mesh Beacon», the Beacon Type field set to 0x01 (Secure Network beacon), and the Flag AD Type is not included in Beacon Data (non-connectable).

  All multiple-octet numeric values in mesh beacons are sent in “network byte order” defined as “big endian”.

### 4.7 Network Layer

The test group objective is to verify functionality of the network layer. Test functionality specific to the ADV bearer for the network layer.

#### 4.7.1 Transmit Mesh Messages over Advertising Bearer to Different Destination Address Types

- **Test Purpose**

  This test group is for generic use and contains seven test cases to verify that a node can transmit non-connectable mesh message AD Type advertisements over the advertising bearer in the context of a mesh Subnet, using previously established network security credentials, to each of the possible Destination Address types (unicast, group, or virtual). The verification is done one value at a time, as enumerated in the test cases in Table 4.6.

  The test cases include verification that a Node can send advertisements using TTL values varying from minimum to maximum allowable.

- **Reference**

  [3] Section 3.3.1, 3.4.2, 3.4.4, 3.4.6.3, 3.8.6.2, 3.8.6.3

- **Initial Condition**

  - The IUT is a Node.
  - The IUT supports the GAP Broadcaster role.
  - The Lower Tester is a Node.
  - The Lower Tester supports the GAP Observer role.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT’s output filter allows any messages specified to pass.
  - The Network Transmit state on the IUT is equal to 0x00.
  - Condition specified for this Test Case in Table 4.6.

- **Test Procedure**

  1. The Lower Tester begins scanning for advertising packets from the IUT.
2. The Upper Tester sends a command to the IUT to initiate advertising using non-connectable advertising packets with a payload that consists an AD Type value of "mesh message", a payload length of 29 bytes, and a valid mesh packet as payload. The fields of the packet are set as follows, with all multiple-octet values sent in "big endian", with DST and TransportPDU encrypted with AES-CCM using the Encryption Key (derived from shared Network Key), a new Network Nonce (Nonce Type set to 0x00, and the CTL and TTL, SEQ, SRC, and IVI fields as set below), and with CTL, TTL, SEQ, and SRC obfuscated with AES-ECB using the Privacy Key (derived from shared Network Key) and a Privacy Random (generated from 0x0000000000, IV Index, encrypted DST, and encrypted TransportPDU):
   - IVI set to the least significant bit of the current IV Index
   - NID set to the least 7 significant bits of the Network Key Identifier of the Network Key
   - CTL set to 0
   - TTL set to 0x00
   - SEQ set to any value between 0x000000 and 0xFFFFFFFF inclusive
   - SRC set to one of the IUT’s local unicast addresses
   - DST set to the specified value in Table 4.6 for this Test Case
   - TransportPDU set to between 1 and 16 inclusive octets of data
   - NetMIC set to a 32-bit value generated by the IUT

3. The Lower Tester expects an advertising packet from the IUT containing the expected AD Type and payload. The Lower Tester examines the IVI and NID fields to confirm they are set using the expected security credentials. The Lower Tester confirms the packet is authenticated using the previously established network security credentials and the NetMIC field is set correctly.

4. Repeat steps 2-3 with a TTL value of 10 in step 2.
5. Repeat steps 2-3 with a TTL value of 127 in step 2.
6. The Lower Tester sends a Config Network Transmit Set message to the IUT with the NetworkTransmitCount set to 0b010 and the NetworkTransmitIntervalSteps set to 0b10000.
7. The Lower Tester expects the IUT to respond with a Config Network Transmit Status message containing the field values requested in step 6.
8. The Upper Tester orders the IUT to send a Network PDU with parameters as described in step 2, using any valid DST field value defined in Table 4.6.
9. The Lower Tester expects the IUT to send the requested Network PDU 3 times.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Initial Condition</th>
<th>DST Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4.7.1.1 MESH/NODE/NET/BV-01-C</strong></td>
<td>The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to a unicast address.</td>
<td>any value between 0x0001 and 0x7FFF inclusive (a valid unicast address)</td>
</tr>
</tbody>
</table>
### Table 4.6: Transmit Mesh Messages over Advertising Bearer Test Cases

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Initial Condition</th>
<th>DST Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4.7.1.2 MESH/NODE/NET/BV-02-C</strong> <em>(Send Mesh Message to Virtual Address)</em></td>
<td>The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to a virtual address based on a virtual label UUID.</td>
<td>any value between 0x8000 and 0xBFFF inclusive</td>
</tr>
<tr>
<td><strong>4.7.1.3 MESH/NODE/NET/BV-03-C</strong> <em>(Send Mesh Message to Dynamically Assigned Group Address)</em></td>
<td>The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to a dynamically assigned group address.</td>
<td>any value between 0xC000 and 0xFEFF inclusive</td>
</tr>
<tr>
<td><strong>4.7.1.4 MESH/NODE/NET/BV-04-C</strong> <em>(Send Mesh Message to All-proxies Address)</em></td>
<td>The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to the all-proxies address.</td>
<td>0xFFFFC</td>
</tr>
<tr>
<td><strong>4.7.1.5 MESH/NODE/NET/BV-05-C</strong> <em>(Send Mesh Message to All-friends Address)</em></td>
<td>The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to the all-friends address.</td>
<td>0xFFFFD</td>
</tr>
<tr>
<td><strong>4.7.1.6 MESH/NODE/NET/BV-06-C</strong> <em>(Send Mesh Message to All-relays Address)</em></td>
<td>The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to the all-relays address.</td>
<td>0xFFFFE</td>
</tr>
<tr>
<td><strong>4.7.1.7 MESH/NODE/NET/BV-07-C</strong> <em>(Send Mesh Message to All-nodes Address)</em></td>
<td>The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to the all-nodes address.</td>
<td>0xFFFF</td>
</tr>
</tbody>
</table>

**Expected Outcome**

**Pass verdict**

The IUT sends an advertising packet to the Lower Tester with the correct AD type and the expected payload, with all fields containing valid values and formatted correctly.

Any message sent by the IUT is delivered to all network interfaces supported by the node. The IVI and NID fields are set using the appropriate network security credentials.

The NetMIC field is set correctly using the appropriate network security credentials.

The SEQ field value is unique for each Network PDU sent by the IUT.
All multiple-octet numeric values are in “big endian” byte order.

In step 7, the IUT sends with a Config Network Transmit Status message containing the field values requested in step 7.

In step 9, the IUT sends the Network PDU 3 times.

4.7.2 Receive Mesh Messages over Advertising Bearer Sent to Different Destination Address Types

• Test Purpose

This test group is for generic use and contains seven test cases to verify that a node can receive authenticated non-connectable mesh message AD Type advertisements over the advertising bearer in the context of a mesh Subnet, using previously established network security credentials, to each of the possible Destination Address types (unicast, group, or virtual). The verification is done one value at a time, as enumerated in the test cases in Table 4.7.

The test cases include verification that a Node can receive advertisements using TTL values varying from minimum to maximum allowable.

• Reference

[3] Section 3.3.1, 3.4.2, 3.4.4, 3.4.6.2, 3.8.6.2, 3.8.6.3

• Initial Condition

- The IUT is a Node.
- The IUT supports the GAP Observer role.
- The Lower Tester is a Node.
- The Lower Tester supports the GAP Broadcaster role.
- The IUT and the Lower Tester share previously established network security credentials.
- The IUT’s network layer input filter allows any messages specified to pass.
- Condition specified for this Test Case in Table 4.7.

• Test Procedure

1. The Lower Tester initiates advertising using non-connectable advertising packets with a payload that consists of the mesh message AD Type value, and the Contents containing a Network PDU of maximum size (29 bytes).
2. The Upper Tester sends a command to the IUT to scan for advertising packets and receives the advertising packets sent by the Lower Tester.
3. The Upper Tester confirms that the IUT receives one or more non-connectable advertising packets containing the expected AD Type and payload and notifies the Upper Tester of packet receipt.
4. The Lower Tester sends a network packet with fields set as follows, with all multiple-octet values sent in "big endian", with DST and TransportPDU encrypted with AES-CCM using the Encryption Key (derived from shared Network Key), a new Network Nonce (Nonce Type set to 0x00, and the CTL and TTL, SEQ, SRC, and IVI fields as set below), and with CTL, TTL, SEQ, and SRC obfuscated with AES-ECB using the Privacy Key (derived from shared Network Key) and a
Privacy Random (generated from 0x0000000000, IV Index, encrypted DST, and encrypted TransportPDU):
- IVI set to the least significant bit of the current IV Index
- NID set to the least 7 significant bits of the Network Key Identifier of the Network Key
- CTL set to 0
- TTL set to 0x00
- SEQ set to any value between 0x000000 and 0xFFFFFFFF inclusive
- SRC set to the Lower Tester’s address, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address)
- DST set to the DST Value for this test case in Table 4.7
- TransportPDU set to between 1 and 16 inclusive octets of data
- NetMIC set to a 32-bit value generated by the Lower tester

5. Repeat step 4 with a TTL value of 1.
6. Repeat step 4 with a TTL value of 10.
7. Repeat step 4 with a TTL value of 127.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Initial Condition</th>
<th>DST Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.7.2.1 MESH/NODE/NET/BV-08-C [Receive Mesh Message Sent to Unicast Address]</td>
<td>N/A</td>
<td>one of the IUT’s local unicast addresses, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address)</td>
</tr>
<tr>
<td>4.7.2.2 MESH/NODE/NET/BV-09-C [Receive Mesh Message Sent to Virtual Address]</td>
<td>The Network Layer on the IUT knows a virtual address based on a virtual label UUID.</td>
<td>a virtual address the IUT knows, a value between 0x8000 and 0xBFFF inclusive (a valid virtual address), where the 14 least significant bits are set to the 14 least significant bits of the result generated by AES-CMAC (Virtual Label UUID, “vtad”)</td>
</tr>
<tr>
<td>4.7.2.3 MESH/NODE/NET/BV-10-C [Receive Mesh Message sent to Dynamic Group Address]</td>
<td>The Network Layer on the IUT knows a dynamic group address.</td>
<td>a group address the IUT knows, a value between 0xC000 and 0xFFFE inclusive.</td>
</tr>
<tr>
<td>4.7.2.4 MESH/NODE/NET/BV-11-C [Receive Mesh Message sent to All-proxyes Address]</td>
<td>The IUT supports the proxy feature and it is enabled.</td>
<td>0xFFFC</td>
</tr>
<tr>
<td>4.7.2.5 MESH/NODE/NET/BV-12-C [Receive Mesh Message sent to All-friends Address]</td>
<td>The IUT supports the friend feature and it is enabled.</td>
<td>0xFFFFD</td>
</tr>
<tr>
<td>Test Case</td>
<td>Initial Condition</td>
<td>DST Value</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td><strong>4.7.2.6 MESH/NODE/NET/BV-13-C [Receive Mesh Message sent to All-relays Address]</strong></td>
<td>The IUT supports the relay feature and it is enabled.</td>
<td>0xFFFE</td>
</tr>
<tr>
<td><strong>4.7.2.7 MESH/NODE/NET/BV-14-C [Receive Mesh Message sent to All-nodes Address]</strong></td>
<td>N/A</td>
<td>0xFFFF</td>
</tr>
</tbody>
</table>

*Table 4.7: Receive Mesh Messages over Advertising Bearer Test Cases*  

- **Expected Outcome**  
  **Pass verdict**  
  The IUT receives an advertising packet from the Lower Tester with the correct AD Type and the expected payload, with all fields containing valid values and formatted correctly. All encrypted fields are authenticated successfully.

  The NID field value matches an NID known by the IUT.

  If the mesh message received is sent to a unicast address, it is processed by only one Element.

  If the mesh message received is sent to a group or a virtual address, it is processed by all Elements subscribed to this address.

---

**4.7.3 MESH/NODE/NET/BI-01-C [Ignore Connectable Advertisements on ADV Bearer]**

- **Test Purpose**  
  Verify that a Node IUT silently ignores mesh packets sent in connectable advertisement PDUs.

- **Reference**  
  [3] Section 3.3.1

- **Initial Condition**
  - The IUT is Observer.
  - The Lower Tester is Broadcaster.
  - The IUT and the Lower Tester share previously established network security credentials.

- **Test Procedure**
  1. The Lower Tester initiates advertising using connectable advertising packets with a payload that consists of the Mesh Packet AD Type value, a payload length of 29 bytes, and a valid mesh packet as payload.
  2. The IUT scans for and receives the advertising packet(s) sent by the Lower Tester.
  3. The IUT silently ignores the mesh packets received in connectable advertising PDUs and does not notify the Upper Tester of their receipt.
• Expected Outcome
  Pass verdict

  The IUT ignores any messages received without delivering the Transport PDU to the lower transport layer.

4.7.4 MESH/NODE.NET/BI-02-C [Discard Messages with Invalid Addresses]

• Test Purpose
  Verify that a Node IUT can discard messages received with an invalid SRC or DST address.

• Reference
  [3] Section 3.4.4, 3.4.6.2

• Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Node.
  - The IUT and the Lower Tester share previously established network security credentials.
  - If the IUT supports message relaying, relaying is enabled.

• Test Procedure
  1. The Lower Tester sends a message to the IUT with SRC set to a valid address and DST set to 0b0000000000000000.
  2. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
  3. The Lower Tester sends a message to the IUT with SRC set to the unassigned address and DST set to the IUT’s address.
  4. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
  5. The Lower Tester sends a message to the IUT with SRC set to a virtual address the IUT knows and DST set to the IUT’s address.
  6. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
  7. The Lower Tester sends a message to the IUT with SRC set to a virtual address the IUT does not know and DST set to the IUT’s address.
  8. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
  9. The Lower Tester sends a message to the IUT with SRC set to a group address the IUT knows and DST set to the IUT’s address.
 10. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
 11. The Lower Tester sends a message to the IUT with SRC set to a group address the IUT does not know and DST set to the IUT’s address.
 12. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
• Expected Outcome

Pass verdict

When the IUT receives a Mesh Packet with an invalid SRC or DST address, the IUT discards the message without delivering the Transport PDU to the lower transport layer.

If the IUT supports message relaying, the message with invalid SRC or DST is not relayed.

4.7.5 MESH/NODE/NET/BI-03-C [Receive Network Packet with Invalid NID Field]

• Test Purpose

Verify that if a Node IUT can ignore a message containing a NID that does not match any known NIDs.

• Reference

[3] Section 3.3.1, 3.4.4, 3.4.6.2

• Initial Condition

- The IUT is a Node.
- The Lower Tester emulates a Node.
- The IUT and the Lower Tester share previously established network security credentials.

• Test Procedure

1. The Lower Tester sends a network packet with fields set as follows, with all multiple-octet values sent in "big endian", but authenticated and encrypted using a Network Key that the IUT does not know; with DST and TransportPDU encrypted with AES-CCM using the Encryption Key (derived from a Network Key the IUT does not know), a new Network Nonce (Nonce Type set to 0x00, and the CTL and TTL, SEQ, SRC, and IV Index fields as set below), and with CTL, TTL, SEQ, and SRC obfuscated with AES-ECB using the Privacy Key (derived from a Network Key the IUT does not know) and a Privacy Random (generated from 0x0000000000, IV Index, encrypted DST, and encrypted TransportPDU):
   - IVI set to the least significant bit of the current IV Index
   - NID set to a value that does not match any NID values known to the IUT
   - CTL set to 0
   - TTL set to any value between 0x00 and 0x3F inclusive
   - SEQ set to any value between 0x000000 and 0xFFFFFFFF inclusive
   - SRC set to the Lower Tester’s address, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address)
   - DST set to one of the IUT’s local unicast addresses, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address)
   - TransportPDU set to between 1 and 16 inclusive octets of data
   - NetMIC set to the 4 most significant octets of the MIC generated by using AES-CCM to encrypt the DST and TransportPDU fields using the Encryption Key and Network Nonce
2. The Upper Tester expects the IUT’s Network Layer to discard the packet without delivering the Transport PDU to the next upper layer (the Transport Layer).

- **Expected Outcome**
  
  **Pass verdict**
  
  The IUT’s Network Layer receives the network packet, processes it, and discards it without delivering the Transport PDU to the Lower Transport layer.

### 4.7.6 MESH/NODE/NET/BI-04-C [Receive Network Packet with RFU Address Value]

- **Test Purpose**
  
  Verify that a node ignores a network layer message sent to a Destination Address set to an RFU value.

- **Reference**
  
  [3] Section 3.1.2, 3.3.1, 3.4.2, 3.4.4, 3.4.6.2, 3.8.6.2, 3.8.6.3

- **Initial Condition**
  
  - The IUT is a Node.
  - The Lower Tester emulates a Node.
  - The IUT and the Lower Tester share previously established network security credentials.

- **Test Procedure**

  1. The Lower Tester sends a network packet with fields set as follows, with all multiple-octet values sent in "big endian", with DST and TransportPDU encrypted with AES-CCM using the Encryption Key (derived from shared Network Key), a new Network Nonce (Nonce Type set to 0x00, and the CTL and TTL, SEQ, SRC, and IVI fields as set below), and with CTL, TTL, SEQ, and SRC obfuscated with AES-ECB using the Privacy Key (derived from shared Network Key) and a Privacy Random (generated from 0x0000000000, IV Index, encrypted DST, and encrypted TransportPDU):
     
     - IVI set to the least significant bit of the current IV Index
     - NID set to a value that does not match any NID values known to the IUT
     - CTL set to 0
     - TTL set to any value between 0x00 and 0x3F inclusive
     - SEQ set to any value between 0x000000 and 0xFFFFFFFF inclusive
     - SRC set to the Lower Tester’s address, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address)
     - DST set to any RFU value between 0xFF00-0xFFFB inclusive
     - TransportPDU set to between 1 and 16 inclusive octets of data
     - NetMIC set to the 4 most significant octets of the MIC generated by using AES-CCM to encrypt the DST and TransportPDU fields using the Encryption Key and Network Nonce
2. The Upper Tester expects the IUT’s Network Layer to discard the packet without delivering the Transport PDU to the next upper layer (the Transport Layer).

- Expected Outcome
  Pass verdict

The IUT’s Network Layer receives the network packet, processes it, and discards it without delivering the Transport PDU to the Lower Transport layer.

### 4.8 Message Relaying

The test group objective is to verify functionality related to message relaying.

#### 4.8.1 MESH/NODE/RLY/BV-01-C [Relay Network PDUs based on SRC, DST, TTL and Relay Retransmit]

- **Test Purpose**
  Verify that an IUT acting as Relay node forwards valid network PDUs selectively, based on the SRC, DST and TTL field values and on the Relay Retransmit state value.

- **Reference**
  [3] Section 3.8.4, 3.4.4, 3.4.6.1, 3.4.6.3, 4.2.20, 4.3.2, 13, 4.3.2.14

- **Initial Condition**
  - The IUT and the Lower Tester share previously established network security credentials and a virtual address.
  - The IUT supports the Relay feature and the feature is enabled.
  - The Relay Retransmit Count state on the IUT is equal to 0b000.
  - The Lower Tester is configured to generate valid network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the SEQ field starting from 0 and increasing on each new PDU, and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU. All network PDUs are encrypted using the shared network key.

- **Test Procedure**
  1. The Lower Tester sends a network PDU with the SRC field set to its own unicast address, the TTL field set to a random valid value greater than or equal to 2, and the DST field set to a unicast address not matching the Element addresses of the IUT.
  2. The Lower Tester expects the IUT to send the network PDU received in step 1 with the TTL field decremented by 1 and re-encrypted using the same shared network key.
  3. The Lower Tester sends a new network PDU with the SRC field set to its own unicast address, the TTL field set to 1, and the DST field set to a unicast address not matching the Element addresses of the IUT.
  4. The Lower Tester expects the IUT to not forward the received network PDU.
  5. The Lower Tester sends a new network PDU with the SRC field set to its own unicast address, the TTL field set to 0, and the DST field set to a unicast address not matching the Element addresses of the IUT.
  6. The Lower Tester expects the IUT to not forward the received network PDU.
7. The Lower Tester repeats steps 1–6 two more times, using a new network PDU on each iteration through step 1, and setting the DST field first to a dynamically assigned group address, then to the shared virtual address.

8. The Lower Tester sends a new network PDU with the SRC field set to its own unicast address, the TTL field set to a random valid value greater than or equal to 2, and the DST field set to the unicast address of the IUT.

9. The Lower Tester expects the IUT to not forward the received network PDU.

10. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01, the RelayRetransmitCount field set to 0b001, and the RelayRetransmitIntervalSteps field set to 0b10000.

11. The Lower Tester expects to receive a Config Relay Status message in response from the IUT with the Relay field set to 0x01, the RelayRetransmitCount field value set to 0b001, and the RelayRetransmitIntervalSteps field value set to 0b10000.

12. The Lower Tester sends a network PDU with the SRC field set to its own unicast address, the TTL field set to a random valid value greater than or equal to 2, and the DST field set to a unicast address not matching the Element addresses of the IUT.

13. The Lower Tester expects the IUT to forward the network PDU received in step 12, with the TTL field value decremented by 1, two times.
Figure 4.5: MESH/NODE/RLY/BV-01-C

• Expected Outcome

Pass verdict

In each iteration through step 2, the IUT sends a copy of the received network PDU, without retransmit, with the TTL field value decremented by 1 and all other plain field values unchanged. The
DST and the TransportPDU fields are re-encrypted and the NetMIC field contains the new generated MIC. The CTL, TTL, SEQ and SRC fields are properly obfuscated.

In each iteration through steps 4, 6 and 9, the IUT does not forward the received network PDU.

In step 13, the IUT sends a copy of the received network PDU, with the TTL field value decremented by 1, two times.

### 4.8.2 MESH/NODE/RLY/BV-02-C [Network Message Cache]

- **Test Purpose**
  Verify that an IUT acting as Relay node maintains a properly implemented message cache.

- **Reference**
  [3] Section 2.2, 3.4.4.4, 3.4.4.6, 3.4.6.1, 3.4.6.4

- **Initial Condition**
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT supports the Relay feature and the feature is enabled.
  - The maximum number of entries supported by the network message cache on the IUT has been supplied in the IXIT [6]. This number is denoted as N and is greater than or equal to 2.
  - The Lower Tester is configured to generate valid network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the TTL field set to a random valid value greater than or equal to 2 on each new PDU, the SRC field set to a unique unicast address (not matching the Element addresses of the IUT) on each new PDU, the DST set to the all nodes address and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU. The network PDUs are encrypted using the shared network key.

- **Test Procedure**
  1. The Lower Tester sends a new network PDU.
  2. The Lower Tester expects the IUT to send the network PDU received in step 1 with the TTL field decremented by 1.
  3. The Lower Tester resends the same network PDU as in step 1.
  4. The Lower Tester expects the IUT to not forward the same network PDU a second time.
  5. The Lower Tester repeats steps 1–2 an additional (N – 1) times.
  6. The Lower Tester repeats steps 3–4 an additional N times, iterating through all N network PDUs that have been sent in the N passes through step 1.
  7. The Lower Tester repeats steps 1–4 one time.
  8. The Lower Tester resends all the (N – 1) network PDUs sent in step 5 and expects the IUT to not forward these PDUs.
  9. The Lower Tester resends the same network PDU that it has sent in the first iteration through step 1.
  10. The Lower Tester expects the IUT to resend the network PDU that it sent in the first iteration through step 2.
  11. The Lower Tester sends a new network PDU with the defined field value generation procedure, but with the TTL field set to a random valid value greater than or equal to 4.
  12. The Lower Tester expects the IUT to send the network PDU received in step 11 with the TTL field decremented by 1.
13. The Lower Tester forwards the network PDU received in step 12 after decrementing the TTL by 1.

14. The Lower Tester expects the IUT to not forward the network PDU received in step 13.

15. The Upper Tester orders the IUT to send a valid network PDU with the DST field set to the all nodes address and the TTL field set to a random valid value greater than or equal to 4.

16. The Lower Tester expects the IUT to send a valid network PDU with the requested field values.

17. The Lower Tester forwards the network PDU received in step 16 after decrementing the TTL by 1.

18. The Lower Tester expects the IUT to not forward the network PDU received in step 17.
Figure 4.6: MESH/NODE/RLY/BV-02-C
• Expected Outcome

Pass verdict

In all passes through step 2, the IUT forwards the received network PDU with the TTL field decremented by 1 and re-encrypted using the shared network key.

In all passes through step 4, the IUT does not forward the received network PDU.

In step 8, the IUT does not forward the received network PDUs, confirming that those are still in the cache.

In step 10, the IUT forwards the received network PDU, confirming that it has been removed from the cache in step 7.

In step 12, the IUT forwards the received network PDU.

In step 14, the IUT does not forward the received network PDU, confirming that the cache filters the same network PDU with lower TTL values.

In step 16, the IUT sends a valid network PDU with the DST and the TTL fields set to the requested values.

In step 18, the IUT does not forward the received network PDU, confirming that the cache filters the network PDUs generated by the IUT and relayed by other nodes.

4.8.3 MESH/NODE/RLY/BI-01-C [Ignore Invalid IV Index]

• Test Purpose

Verify that an IUT acting as Relay node does not relay network PDUs encrypted using an invalid IV Index.

• Reference

[3] Section 2.2, 3.4.4, 3.4.4.1, 3.4.6.3

• Initial Condition

- The IUT and the Lower Tester share previously established network security credentials.
- The IUT supports the Relay feature and the feature is enabled.
- The Lower Tester is configured to generate network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the TTL field set to a random valid value greater than or equal to 2 on each new PDU, the SRC field set to a unique unicast address (not matching the Element addresses of the IUT) on each new PDU, the SEQ field set randomly on each new PDU, the DST set to the all nodes address and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU. The network PDUs are encrypted using the shared network key.

• Test Procedure

1. The Lower Tester sends a new network PDU encrypted with the current IV index plus one.
2. The Lower Tester expects the IUT to not forward the received network PDU.
3. The Lower Tester sends a new network PDU encrypted with the current IV index minus two.
4. The Lower Tester expects the IUT to not forward the received network PDU.
• Expected Outcome
  Pass verdict

In steps 2 and 4, the IUT does not forward the received network PDUs.

4.8.4 MESH/NODE/RLY/BI-02-C [Ignore Invalid SRC]

• Test Purpose
  Verify that an IUT acting as Relay node does not relay network PDUs with an invalid SRC field value.

• Reference
  [3] Section 3.4.4, 3.4.6.3

• Initial Condition
  - The IUT and the Lower Tester share previously established network security credentials and a virtual address.
  - The IUT supports the Relay feature and the feature is enabled.
  - The Lower Tester is configured to generate network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the SEQ field set randomly on each new PDU, the DST set to the all nodes address and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU. The network PDUs are encrypted using the shared network key.

• Test Procedure
  1. The Lower Tester sends a network PDU, with the SRC field set to the unicast address of the IUT and the TTL field set to a random valid value greater than or equal to 2.
  2. The Lower Tester expects the IUT to not forward the received network PDU.
  3. The Lower Tester repeats steps 1-2 three more times, using a new network PDU on each iteration through step 1, and setting the SRC field to the unassigned address, a dynamically assigned group address, and to the shared virtual address, respectively.

• Expected Outcome
  Pass verdict

In each iteration through step 2, the IUT does not forward the received network PDUs.

4.9 Transport Layers
Test functionality of the Upper Transport Layer and the Lower Transport Layer of the Mesh specification.

4.9.1 MESH/NODE/TNPT/BV-01-C [Ignore Message with Unknown Application Key]

• Test Purpose
  Verify that an IUT node ignores messages encrypted with an unknown application key.

• Reference
  [3] Section 3.5.4.1
• Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
  - An application on the IUT expects to receive acknowledged messages encrypted with one of the IUT’s application keys.

• Test Procedure
  1. The Lower Tester sends an acknowledged message to the IUT encrypted with an application key different than all the IUT’s application keys.
  2. The Lower Tester expects to receive no response from the IUT.

• Expected Outcome
  Pass verdict
  In step 2, the IUT does not send any response.

4.9.2 MESH/NODE/TNPT/BV-02-C [Ignore Message with Unknown Device Key]

• Test Purpose
  Verify that an IUT node ignores messages encrypted with an unknown device key.

• Reference
  [3] Section 3.5.4.2

• Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
  - An application on the IUT expects to receive acknowledged messages encrypted with the IUT’s device key.

• Test Procedure
  1. The Lower Tester sends an acknowledged message to the IUT encrypted with a device key different than the IUT’s device key.
  2. The Lower Tester expects to receive no response from the IUT.

• Expected Outcome
  Pass verdict
  In step 2, the IUT does not send a response.

4.9.3 MESH/NODE/TNPT/BV-03-C [Send an Unsegmented Access Message]

• Test Purpose
  Verify that an IUT node can send Unsegmented Access Messages.

• Reference
  [3] Section 3.5.2.3, 3.5.2.3.1, 3.5.3
• Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

• Test Procedure
  1. The Upper Tester orders the IUT to send an Unsegmented Access Message with specified valid values for the TTL, DST and Access PDU fields, and to encrypt the message using a shared network key and a shared application key.
  2. The Lower Tester expects to receive a valid Network PDU, with the field values specified in step 1 and properly encrypted with the specified keys.

• Expected Outcome
  Pass verdict

In step 2, the IUT sends a Network PDU containing the expected field values. The Access PDU has been encrypted with the specified application key and the Lower Transport PDU has been encrypted with the specified network key. The CTL field is set to 0, the SEG field is set to 0, the AKF field is set to 1, the AID field is set to the correct identifier of the shared application key, and the TransMIC has a size of 4 octets.

4.9.4 MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast]

• Test Purpose
  Verify that an IUT node can send Segmented Access Messages to a unicast address.

• Reference
  [3] Section 3.5.2.3.1, 3.5.2.4, 3.5.2.2, 3.5.2.3

• Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

• Test Procedure
  1. The Upper Tester orders the IUT to send an access message with specified valid values for the TTL, DST and Access PDU fields, and to encrypt the message using a shared network key, a shared application key and with a specified TransMIC size. The destination address is a unicast address. The size of the access message is greater than the maximum size that can fit in an Unsegmented Access Message and a number of Segmented Access Messages, denoted as N, are needed to send the access message.
  2. The Lower Tester expects the IUT to send N Segmented Access Messages with the SEG field set to 1, the SegN field set to N-1 and the same value for the, AKF, AID, SZMIC and SeqZero fields. The SegO fields contain all values from 0 to N-1. All messages carry the maximum payload, except for the segment with the SegO field set to N-1.
  3. The Lower Tester sends a Segment Acknowledgment Message to the IUT with the SeqZero field set to the same value from the Segmented Access Message received in step 2, and the BlockAck field indicating that not all segments have been received from the IUT.
  4. The Lower Tester expects the IUT to resend the segments that have not been acknowledged in step 3.
5. The Lower Tester sends another Segment Acknowledgment Message to the IUT with the BlockAck field indicating that all segments have been received.

6. The Lower Tester expects the IUT to not send any more segments from this transaction.

![Diagram of Mesh Profile (MESH) Test Suite](image)

**Figure 4.7: MESH/NODE/TNPT/BV-04-C**

- **Expected Outcome**

  **Pass verdict**

  In step 2, the IUT sends all Segmented Access Messages. For all these messages, the CTL field is set to 0, the SEG field is set to 1, the AKF field is set to 1, the AID field is set to the correct identifier of the shared application key, the TransMIC has the requested size, the SZMIC field is set to 1 only if the size of the TransMIC is 8 octets, the SeqZero field is set to the least significant 13 bits of the value of the SEQ field from the segment with the SegO field set to 0, the SegN field is set to N-1 and the SegO fields have all the values from 0 to N-1.

  In step 4, the IUT sends only the segments that have not been acknowledged in step 3.

  In step 6, the IUT does not send any other segments.

  Each message is formatted correctly, and the Lower Tester can decrypt the reassembled message and confirm it is the same message requested by the Upper Tester in step 1.
4.9.5 MESH/NODE/TNPT/BV-05-C [Send a Segmented Access Message – Group/Virtual Address]

- **Test Purpose**
  Verify that an IUT node can send Segmented Access Messages to a group address, and a virtual address.

- **Reference**
  [3] Section 3.5.2.3.1, 3.5.2.4, 3.5.2.2, 3.5.2.3, 3.7.1

- **Initial Condition**
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
  - A known virtual address has been programmed into the IUT.

- **Test Procedure**
  1. The Upper Tester orders the IUT to send an access message with specified valid values for the TTL, DST and Access PDU fields, and to encrypt the message using a shared network key and a shared application key. The destination address is a group address. The size of the access message is greater than the maximum size that can fit in an Unsegmented Access Message and a number of Segmented Access Messages, denoted as N, are needed to send the access message.
  2. The Lower Tester expects the IUT to send N Segmented Access Messages with the SEG field set to 1, the SegN field set to N-1 and the same value for the, AKF, AID, SZMIC and SeqZero fields. The SegO fields contain all values from 0 to N-1. All messages carry the maximum payload, except for the segment with the SegO field set to N-1. The IUT sends the entire set of N Segmented Access Messages at least two times.
  3. Repeat steps 1–2 using a virtual address as destination address.

- **Expected Outcome**
  **Pass verdict**

  In step 2, the IUT sends all Segmented Access Messages at least two times. For all these messages, the CTL field is set to 0, the SEG field is set to 1, the AKF field is set to 1, the AID field is set to the correct identifier of the shared application key, the TransMIC has the requested size, the SZMIC field is set to 1 only if the size of the TransMIC is 8 octets, the SeqZero field is set to the least significant 13 bits of the value of the SEQ field from the segment with the SegO field set to 0, the SegN field is set to N-1 and the SegO fields have all the values from 0 to N-1.

  Each message is formatted correctly, and the Lower Tester can decrypt the reassembled message and confirm it is the same message requested by the Upper Tester in step 1.

4.9.6 MESH/NODE/TNPT/BV-06-C [Receive an Unsegmented Access Message]

- **Test Purpose**
  Verify that an IUT node can receive and process an Unsegmented Access Message.

- **Reference**
  [3] Section 3.5.2, 3.5.2.1
• **Initial Condition**
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

• **Test Procedure**
  1. The Lower Tester sends a small access message as an Unsegmented Access Message to the IUT, encrypted with a shared network key and a shared application key.
  2. The Upper Tester expects an indication from the IUT containing the same access message data sent by the Lower Tester in step 1.

• **Expected Outcome**
  
  Pass verdict

  In step 2, the IUT indicates to the Upper Tester that it has received an access message from the Lower Tester. The message data is the same as the data sent by the Lower Tester in step 1.

### 4.9.7 MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast]

• **Test Purpose**
  Verify that an IUT node can receive and reassemble Segmented Access and Control Messages directed to a unicast address.

• **Reference**
  [3] Section 3.1.2, 3.5.2.3.1, 3.5.2.4, 3.5.2.3, 3.5.2.2

• **Initial Condition**
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

• **Test Procedure**
  1. The Lower Tester sends a large access message in a series of Segmented Access Messages to the IUT, encrypted with a shared network key and a shared application key. All messages contain valid field values.
  2. The Lower Tester expects a Segment Acknowledgement Message from the IUT with the same SeqAuth value as the Segmented Access Messages, and the BlockAck field value indicating at least one segment has been received.
  3. The Lower Tester repeats steps 1–2 until the Segment Acknowledgement Message received in step 2 indicates that all segments have been received. When repeating step 1, the Lower Tester sends only the segments that have not been acknowledged by the IUT in step 2.
  4. The Upper Tester expects an indication from the IUT containing the data of the large message segmented by the Lower Tester in the first iteration through step 1.
  5. The Lower Tester resends the segments from the last iteration through step 1.
  6. The Lower Tester expects the IUT to immediately send the Segment Acknowledgement Message indicating that all segments have been received.
  7. Repeat steps 1–6 but with a large control message, sending it in a series of Segmented Control Messages. If the IUT does not support any Transport Control Message opcode that allows segmentation, then use an RFU value (the IUT will still report the reception of the complete message in step 4 but will not respond to the control opcode).
Figure 4.8: MESH/NODE/TNPT/BV-07-C

**Expected Outcome**

**Pass verdict**

In both iterations (for access and control messages):

In step 2, the IUT sends a Segment Acknowledgement Message. TheCTL field is set to 1, the SEG field is set to 0, the Opcode field is set to 0x00, the OBO field is set to 0, the SeqZero field is set to the SeqZero field from the Segmented Access or Control Messages sent by the Lower Tester in step 1, the RFU bits are set to 0 and the BlockAck field has a valid value. In the final iteration through steps 1–2, the IUT sends the message with the BlockAck field indicating all segments have been received. If more than one iteration through steps 1–2 is executed, then each BlockAck field value from the message sent by the IUT acknowledges all the segments from the previous pass.

In step 4, after the IUT has acknowledged all segments, the IUT reassembles the complete message (and decrypts the access message, if applicable) and indicates to the Upper Tester that it has received a message from the Lower Tester. The message data is the same as the data of the large message segmented by the Lower Tester in the first pass of step 1.

In step 6, the IUT sends the Segment Acknowledgement Message immediately after step 5, i.e., faster than the acknowledgement timer period.
4.9.8 MESH/NODE/TNPT/BV-08-C [Receive a Segmented Message – Group/Virtual Address]

- **Test Purpose**
  Verify that an IUT node can receive and reassemble Segmented Access and Control Messages directed to a group address or a virtual address.

- **Reference**
  [3] Section 3.5.2.3.1, 3.5.2.4, 3.5.2.3, 3.5.2.2

- **Initial Condition**
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
  - The IUT is subscribed to a shared group address and to a shared virtual address.

- **Test Procedure**
  1. The Lower Tester sends a large access message in a series of Segmented Access Messages to the shared group address, encrypted with a shared network key and a shared application key. All messages contain valid field values. The Lower Tester sends the complete set of Segmented Access Messages three times.
  2. The Upper Tester expects an indication from the IUT containing the data of the large message segmented by the Lower Tester in step 1.
  3. The Lower Tester repeats steps 1–2 replacing the shared group address from the DST field with the shared virtual address.
  4. Repeat steps 1–3 but with a large control message, sending it in a series of Segmented Control Messages. If the IUT does not support any Transport Control Message opcode that allows segmentation, then use an RFU value (the IUT will still report the reception of the complete message in step 2 but will not respond to the control opcode).

- **Expected Outcome**
  **Pass verdict**

  In both iterations (for access and control messages):

  In step 2, the IUT reassembles (and decrypts, if applicable) the complete message and indicates to the Upper Tester that it has received a message from the Lower Tester. The message data is the same as the data of the large message segmented by the Lower Tester in the first pass of step 1.

4.9.9 MESH/NODE/TNPT/BV-09-C [Send Segmented Message to Unicast—Stop When Transaction Rejected]

- **Test Purpose**
  Verify that an IUT node sending Segmented Access Messages to a unicast address stops resending segments when the transaction is rejected by the receiver.

- **Reference**
  [3] Section 3.4.4.4, 3.5.3.3
• Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

• Test Procedure
  1. The Upper Tester starts the procedure described in MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast], executing only steps 1–2.
  2. The Lower Tester sends a Segment Acknowledgement Message with the BlockAck field set to 0.
  3. The Lower Tester expects the IUT to stop resending the segments.

![Diagram showing the flow of messages between the Lower Tester, IUT, and Upper Tester.]

Figure 4.9: MESH/NODE/TNPT/BV-09-C

• Expected Outcome
  Pass verdict

In step 3, the IUT does not resend any segment.

4.9.10 MESH/NODE/TNPT/BV-10-C [Send Segmented Message - Resend Unacknowledged Segments After Segment Transmission Timer Expires]

• Test Purpose
  Verify that an IUT node sending Segmented Access Messages to a unicast address resends unacknowledged segments after the segment transmission timer expires with no Segment Acknowledgement Message received.
• Reference

[3] Section 3.5.3.3, 3.4.4.3

• Initial Condition

- The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

- The timer period is greater than or equal to 200 + 50 * TTL.

• Test Procedure

1. The Upper Tester orders the IUT to send an access message with specified valid values for the TTL, DST and access payload fields, and to encrypt the message using a shared network key, a shared application key and with a specified TransMIC size. The destination address is a unicast address. The size of the access message is greater than the maximum size that can fit in an Unsegmented Access Message and a number of Segmented Access Messages, denoted as N, are needed to send the access message.

2. The Lower Tester expects the IUT to send N Segmented Access Messages with the SEG field set to 1, the SegN field set to N-1 and the same value for the, AKF, AID, SZMIC and SeqZero fields. The SegO fields contain all values from 0 to N-1. All messages carry the maximum payload, except for the segment with the SegO field set to N-1.

3. The Lower Tester waits for the segment transmission timer to expire without sending a message to the IUT.

4. The Lower Tester expects the IUT to resend all segments from step 2.

5. Repeat steps 3–4.
Figure 4.10: MESH/NODE/TNPT/BV-10-C

- **Expected Outcome**

  **Pass verdict**

  The IUT successfully executes the segmentation procedure by resending all segments each time the segment transmission timer expires.

4.9.11 MESH/NODE/TNPT/BV-11-C [Receive Segmented Message - Acknowledgement Timer Expiration]

- **Test Purpose**

  Verify that an IUT node receiving a segmented message sends the Segment Acknowledgement Message when the acknowledgement timer expires.

- **Reference**

  [3] Section 3.4.4.4, 3.5.3.3

- **Initial Condition**

  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
**Test Procedure**

1. The Lower Tester sends a large access message in a series of Segmented Access Messages to the IUT, encrypted with a shared network key and a shared application key. All messages contain valid field values. The Lower Tester sends all but the last segment.
2. The Lower Tester expects a Segment Acknowledgement Message from the IUT with the same SeqAuth field value as the Segmented Access Messages, and the BlockAck field value indicating at least one segment has been received and that the last segment has not been received.
3. The Lower Tester resends the segments sent in step 1.
4. The Lower Tester expects the IUT to resend the same Segment Acknowledgement Message sent in step 2.

![Diagram of test procedure](image)

*Figure 4.11: MESH/NODE/TNPT/BV-11-C*

**Expected Outcome**

**Pass verdict**

In step 2 and step 4, the IUT sends a Segment Acknowledgement Message with the expected parameters, after a time interval greater than or equal to $150 + 50 \times \text{TTL}$, in milliseconds, after the first segment sent by the Lower Tester in steps 1 and 3, respectively.

**4.9.12 MESH/NODE/TNPT/BV-12-C [Receive Segmented Message - Receive New Segment After Incomplete Timer Expires]**

**Test Purpose**

Verify that an IUT node receiving a segmented message cancels the transaction when the incomplete timer expires.

**Reference**

[3] Section 3.8.7
• Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

• Test Procedure
  1. The Lower Tester sends a large access message in a series of Segmented Access Messages to the IUT, encrypted with a shared network key and a shared application key. All messages contain valid field values. The Lower Tester sends all but the last segment.
  2. The Lower Tester expects a Segment Acknowledgement Message from the IUT with the same SeqAuth field value as the Segmented Access Messages, and the BlockAck field value indicating at least one segment has been received and that the last segment has not been received.
  3. The Lower Tester waits for the incomplete timer on the IUT to expire, ignoring all Segment Acknowledgement Messages from the IUT.
  4. The Lower Tester resends the unacknowledged segments to the IUT.
  5. The Lower Tester expects no response from the IUT.
Expected Outcome

Pass verdict

In step 2, the IUT sends a Segment Acknowledgement Message with the expected parameters.

In step 5, the IUT does not send a Segment Acknowledgement Message.

4.9.13 MESH/NODE/TNPT/BV-13-C [Ignore Replayed Message]

Test Purpose

Verify that an IUT node performs replay protection on both the previous and the current IV Index.
• Reference
  [3] Section 3.8.7

• Initial Condition
  - The IUT, the Lower Tester 1 and the Lower Tester 2 are mesh nodes that share previously established network and application security credentials.
  - An application on the IUT expects to receive acknowledged messages encrypted with one of the IUT’s application keys.
  - The replay protection list on the IUT is empty.

• Test Procedure
  1. The Lower Tester 1 sends an acknowledged message to the IUT using the previous IV Index and a sequence number greater than 0.
  2. The Lower Tester 1 expects to receive a response message from the IUT.
  3. The Lower Tester 2 sends an acknowledged message to the IUT using the previous IV Index and the same sequence number as in step 1.
  4. The Lower Tester 2 expects to receive a response message from the IUT.
  5. The Lower Tester 1 sends an acknowledged message to the IUT using the previous IV Index and the same sequence number as in step 1.
  6. The Lower Tester 1 expects to receive no response from the IUT.
  7. The Lower Tester 2 sends an acknowledged message to the IUT using the previous IV Index and the same sequence number as in step 3.
  8. The Lower Tester 2 expects to receive no response from the IUT.
  9. The Lower Tester 1 sends an acknowledged message to the IUT using the previous IV Index and a smaller sequence number than in step 1.
 10. The Lower Tester 1 expects to receive no response from the IUT.
 11. The Lower Tester 2 sends an acknowledged message to the IUT using the previous IV Index and a smaller sequence number than in step 3.
 12. The Lower Tester 2 expects to receive no response from the IUT.
 13. The Lower Testers repeat steps 1–12 using the current IV Index for all messages.
 14. The Upper Tester orders the IUT to disable the replay protection feature, or to clear and then fill out the replay protection list only with dummy source addresses different than the source address of the Lower Tester 1.
 15. The Lower Tester 1 sends an acknowledged message to the IUT using the current IV Index and a sequence number larger than the one used last time.
 16. The Lower Tester 1 expects the IUT not to send a response message.
Figure 4.13: MESH/NODE/TNPT/BV-13-C
• Expected Outcome
  Pass verdict

When the IUT receives a message with a sequence number that is not greater than the last saved sequence number for that SRC address, the IUT ignores the message.

When the IUT does not have enough resources to perform replay protection for that SRC address, the IUT ignores the message.

4.9.14 MESH/NODE/TNPT/BI-01-C [Ignore Invalid RFU Bits in Segment Acknowledgement Message]

• Test Purpose
  Verifies that an IUT node ignores invalid RFU bits that are set to 1 in the Segment Acknowledgement Message while sending Segmented Access Messages.

• Reference
  [3] Section 3.1.2, 3.5.2.3

• Initial Condition
  - The IUT and the Lower Tester are mesh nodes that share previously established network and application security credentials.

• Test Procedure
  The Lower Tester executes the procedure described in MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast], but sets all the RFU bits to 1 in all Segment Acknowledgement Messages.

• Expected Outcome
  Pass verdict

  The IUT behaves as described in the referenced test case, ignoring the RFU bits that are set to 1.

4.9.15 MESH/NODE/TNPT/BI-02-C [Ignore Message When Segments Received with Old SeqAuth]

• Test Purpose
  Verifies that an IUT node ignores Segmented Access Messages with older SeqAuth values.

• Reference
  [3] Section 3.5.3.4

• Initial Condition
  - The IUT and the Lower Tester are mesh nodes that share previously established network and application security credentials.
• Test Procedure
  1. The Lower Tester executes steps 1 and 2 of the procedure described in MESH/NODE/TNPT/BV-11-C [Receive Segmented Message - Acknowledgement Timer Expiration].
  2. The Lower Tester resends the unacknowledged segments but with an older SeqAuth value.
  3. The Lower Tester expects the IUT not to respond to the segments with the old SeqAuth value.

\[ \text{Figure 4.14: MESH/NODE/TNPT/BI-02-C} \]

• Expected Outcome
  Pass verdict

In step 3, the IUT does not send any Segment Acknowledgement Messages to the Lower Tester.

4.10 IV Update

The test group objective is to verify the IV Update feature.

4.10.1 MESH/NODE/IVU/BV-01-C [IV Update Initiated by Receiving Secure Network Beacons]

• Test Purpose
  Verify that a Node IUT can transition into the IV Update in Progress state upon receiving Secure Network beacon with the IV Update Flag set to 1 and IV Index set to current IV Index + 1.

• Reference
  [3] Section 3.8.2, 3.9.3.1, 3.10.5
• **Initial Condition**
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update procedure is not already in progress.
  - IV Update test mode is activated on the IUT.

• **Test Procedure**
  1. The Lower Tester generates a Secure Network beacon (Mesh Beacon with Beacon Type = 0x01) with the same Network ID and IV Index as the IUT (IV Index = n), the Key Refresh flag and IV Update flag set to 0, and a valid value for Authentication Value. The Lower Tester begins advertising this beacon periodically while also scanning for neighboring beacons.
  2. The Upper Tester orders the IUT to generate a non-connectable Secure Network beacon (Mesh Beacon with Beacon Type = 0x01) with the same Network ID value and IV Index value (n) as the Lower Tester, the Key Refresh and IV Index flags set to 0, and a valid value for Authentication Value. The IUT begins advertising this Mesh Beacon.
  3. The Lower Tester observes the Secure Network beacon advertisement(s) by the IUT.
  4. The Lower Tester sends a mesh message with IV Index set to the previous IV Index (n - 1) to the IUT (e.g., Config Beacon Get) and verifies that the message is accepted and processed by the IUT.
  5. The Lower Tester sends a mesh message with IV Index set to the current IV Index (n) to the IUT.
  6. The Upper Tester observes the IUT to confirm it accepts and processes the mesh message.
  7. The Lower Tester calculates a new Secure Network beacon with the IV Update Flag set to 1, IV Index is incremented by 1 (new IV Index m = n+1), and begins to advertise the updated beacon.
  8. The Upper Tester observes the IUT to confirm that it receives the updated Secure Network beacon from the Lower Tester, enters IV Update in Progress state, and begins advertising Secure Network beacons with the new IV Index (m).
  9. The Lower Tester receives Secure Network beacons from the IUT.
  10. The Lower Tester sends a mesh message with IV Index set to the old IV Index (m-1) to the IUT and verifies that the message is accepted and processed by the IUT by checking the message returned. The message received from the IUT has the sequence number set to zero.
  11. The Lower Tester sends a mesh message with IV Index set to the new IV Index (m) to the IUT and verifies the message is accepted and processed by the IUT by checking the message returned.
  12. The Upper Tester sends a Transit to Normal signal to the IUT to transition to the Normal Operation state.
  13. The Lower Tester expects to receive a Secure Network beacon from the IUT with the IV Index set to the new IV Index (m) and the IV Update flag set to 0.
  14. Repeat step 11.
Figure 4.15: MESH/NODE/IVU/BV-01-C
• **Expected Outcome**

**Pass verdict**

The IUT transmits a Secure Network beacon that is correctly formatted and contains valid values.

When the IUT is in Normal Operation state, it transmits Secure Network beacons using the current IV Index (n). Mesh messages are also transmitting using the current IV Index (n).

When the IUT is in Normal Operation state, it accepts and processes messages with the current IV Index (n) and with the previous IV Index (n-1).

The IUT observes the updated Secure Network beacon from the Lower Tester with the new IV Index value (m, or n+1) and the IV Update Flag set to 1.

When the IUT transitions from the Normal Operation state to the IV Update in Progress state, the IV Index on the node is incremented by one.

When the IUT is in IV Update in Progress state, it transmits Secure Network beacons using the new IV Index (m).

When the IUT is in IV Update in Progress state, it accepts and processes messages with the old IV Index (m-1, or n) and with the new IV Index (m, or n+1).

When the IUT is in IV Update in Progress state, the IV Update Flag is set to 1.

When the IUT is in IV Update in Progress state, it transmits using the new IV Index (m).

When the IUT returns to the Normal Operation state, the sequence number is reset to 0.

When the IUT returns to the Normal Operation state, the IV Update Flag is reset to 0.

When the IUT returns to the Normal Operation state, it transmits using the new IV Index (m, or n+1).

When the IUT transitions from IV Update in Progress state to Normal Operation state, it accepts and processes messages with the old IV Index (m-1) and with the new IV Index (m).

4.10.2 MESH/NODE/IVU/BV-02-C [IV Update Initiated by Sending Secure Network Beacons]

• **Test Purpose**

Verify an IUT Node can initiate the IV update Procedure.

Note: This test covers the starting procedure up to returning to the Normal Operation state by injecting a Transit to Normal signal to IUT. The behaviors in the Normal Operation state before and after the IV Update are the same as being a receiving node.

• **Reference**

[3] Section 3.9.3.1, 3.10.5
• Initial Condition
  - The IUT is a Node
  - The IUT is provisioned to the primary Subnet.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
  - IUT and the Lower Tester share previously established network security credentials.
  - IV Update test mode is activated on the IUT.

• Test Procedure
  1. The Lower Tester generates a Secure Network beacon (Mesh Beacon with Beacon Type = 0x01) with the same Network ID and IV Index as the IUT (IV Index = n), the Key Refresh flag and IV Update flag set to 0, and a valid value for Authentication Value. The Lower Tester begins advertising this beacon periodically while also scanning for neighboring beacons.
  2. The Upper Tester orders the IUT to generate and send a Secure Network beacon using the same network credentials and IV Index as the Lower Tester (n).
  3. The Lower Tester receives the Secure Network beacon from the IUT.
  4. The Upper Tester sends a Transit to IV Update in Progress signal to the IUT, triggering the IUT to transition to the IV Update in Progress state.
  5. The Lower Tester expects the IUT to advertise a Secure Network beacon with the IV Update Flag set to 1 (IV Update in progress) and with the IV Index set to the new IV Index value (m, or n+1). The Lower Tester validates the MIC based on the shared network security credentials.
  6. The Lower Tester generates a Secure Network beacon using the same network credentials and IV Index (m, or n+1) as the IUT, and IV Update Flag set to 1. The Lower Tester begins advertising this beacon periodically while also scanning for neighboring beacons.
  7. The Lower Tester sends a mesh message with the old IV Index (m-1, or n) and verifies it is accepted and processed by IUT by checking the message returned.
  8. The Lower Tester sends a mesh message with the new IV Index (m, or n+1) and verifies it is accepted and processed by IUT by checking the message returned.
  9. The Upper Tester sends a Transit to Normal signal to the IUT to transition to the Normal Operation state.
10. The Lower Tester receives a Secure Network beacon from the IUT showing it is in Normal Operation state with the IV Update flag set to 0 and the IV Index set to the new IV Index (m).
11. The Lower Tester sends a message to IUT with the old IV Index (m-1) and a different SRC address and verifies the message is accepted and processed by the IUT by checking the message returned.
12. The Lower Tester sends a message to IUT with the new IV Index (m) and verifies the message is accepted and processed by the IUT by checking the message returned.
Figure 4.16: MESH/NODE/IVU/BV-02-C
• Expected Outcome
   
   **Pass verdict**

   The IUT correctly sets the SEQ field as the message originator.

   The IUT increments the SEQ field by 1 with every subsequent message sent to the Lower Tester.

   The IUT transitions to IV Update in Progress state upon receiving the Test Mode Transit to IV Update in Progress signal.

   When the IUT is in IV Update in Progress state, the IUT begins to advertise a new Secure Network beacon with the IV Update Flag set to 1 and with the new IV Index (m).

   When the IUT is in IV Update in Progress state, it accepts and processes messages with new IV Index (m, or n+1) and with the previous IV Index (n, or m-1).

   The IUT continues advertising the new Secure Network beacon after another node reacts to the IV Update until the IUT receives a Test Mode Transit to Normal signal and transitions to Normal Operation state.

   After the IUT returns to Normal Operation state, it advertises Secure Network beacons with the IV Update flag set to 0 and the IV Index set to the new IV Index (m).

4.10.3 MESH/NODE/IVU/BV-03-C [IV Update when Operating in Normal Operation for a Normal IV Index]

• Test Purpose
   
   Verify that a Node IUT operating in the normal operation state will update its IV Index after receiving a Secure Network beacon with an updated valid IV Index.

• Reference
   
   [3] Section 3.9.3.1, 3.10.5

• Initial Condition
   
   - The IUT is a Node.
   - The Lower Tester is a Configuration Client.
   - IUT and the Lower Tester share previously established network security credentials.
   - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
   - IV Update test mode is activated on the IUT.
   - IUT is in IV Index recovery state and ready to accept any IV index update less than n + 42.

• Test Procedure
   
   1. The Lower Tester transmits Secure Network beacons with the current IV Index and IV Update Flag set to 0.
   2. The Upper Tester orders the IUT to generate and send Secure Network beacons with the IV Update Flag set to 0 (zero).
3. The Lower Tester generates and transmits a Secure Network beacon with the IV Index set to the current value + 42 and the IV Update Flag set to 0.
4. The Upper Tester confirms that the IUT receives the Secure Network beacon.
5. The IUT begins transmitting Secure Network beacons with the IV Index set to the value received in the Lower Tester’s Secure Network beacons.

**Figure 4.17: MESH/NODE/IVU/BV-03-C**

- **Expected Outcome**
  - **Pass verdict**

The IUT transmits Secure Network beacons with the updated IV Index value after receiving an updated value in Secure Network beacons from the Lower Tester.

**4.10.4 MESH/NODE/IVU/BV-04-C [Stay in IV Update In Progress Operation When Waiting On Segmented Message Transport PDU Acknowledgement]**

- **Test Purpose**
  Verify that a Node IUT can remain in IV Update in Progress state while waiting for acknowledgement of a Transport PDU segmented message and ignore Secure Network beacons with the IV Update Flag set to 0.

- **Reference**
  - [3] Section 3.9.3.1, 3.10.5

- **Initial Condition**
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
- The IUT and the Lower Tester share previously established network security credentials.
- The IUT is advertising Secure Network beacons with the IV Update Flag set to 1.
- IV Update test mode is activated on the IUT.
- The IUT’s segment transmission timer is set to 150 + 50 * TTL ms.

**Test Procedure**

1. The Upper Tester orders the IUT to transmit Segmented Transport PDUs to the Lower Tester.
2. The Lower Tester does not reply with the Segment Acknowledgment messages to the IUT, instead, it generates a Secure Network beacon using the same network credentials and IV Update Flag set to 0 and begins advertising this beacon periodically.
3. The Lower Tester expects the IUT to perform segment retransmissions (if any) using the previous IV Index, and to receive updated Secure Network beacons from the IUT after the IUT’s segment transmission timer expires and the IUT has performed all segment retransmissions and considers the transaction failed.

![Diagram]

**Figure 4.18: MESH/NODE/IVU/BV-04-C**

**Expected Outcome**

**Pass verdict**

The IUT observes the Secure Network beacon with the IV Update Flag set to 0.

The IUT continues to retransmit segments using the previous IV Index and sends Secure Network beacons with the IV Update Flag set to 0 and the IV Index = m after its segment transmission timer expires and the IUT has performed all segment retransmissions and considers the transaction failed.
4.10.5 MESH/NODE/IVU/BV-05-C [IV Update for a Node Not on Primary Subnet]

- **Test Purpose**
  Verify the IV Update procedure when the IUT is a node not on the primary Subnet that receives a Secure Network beacon with the IV Update announcement (IV Update Flag set to 1 and IV Index set to current IV Index + 1). Verify that the IUT can passively accept IV Index updates in the IV Update procedure.

- **Reference**
  [3] Section 3.9.3.1, 3.10.5

- **Initial Condition**
  - The IUT is a Node.
  - The IUT belongs to a subnet that is not the primary Subnet (provisioned with Network Key not at NetKeyIndex = 0).
  - The Lower Tester is a Node.
  - The Lower Tester belongs both to the primary Subnet and to the same subnet as the IUT.
  - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update procedure is not already in progress.
  - IV Update test mode is activated on the IUT.

- **Test Procedure**
  1. The Upper Tester orders the IUT to generate and transmit Secure Network beacons with the IV Update flag set to 0 and the IV Index set to the current value (n).
  2. The Lower Tester generates a Secure Network beacon using the same network credentials and IV Index as the IUT (n), advertises this beacon periodically, and scans for neighboring beacons.
  3. The Lower Tester observes the Secure Network beacon advertisement(s) by the IUT.
  4. The Lower Tester sends a mesh message with the IV Index field set to the current IV Index - 1 (n-1) to the IUT.
  5. The IUT accepts and processes the message from the Lower Tester.
  6. The Lower Tester sends a mesh message with IV Index field set to current IV Index (n) to the IUT.
  7. The IUT accepts and processes the message from the Lower Tester.
  8. The Lower Tester calculates a new Secure Network beacon with the IV Update Flag set to 1, and the IV Index value incremented by 1 (m = n+1), and begins to advertise the updated beacon.
  9. The IUT accepts and processes the message from the Lower Tester. The IUT updates its IV Index to the new value (m = n+1) and begins sending Secure Network beacons with the updated IV Index (m).
  10. The Lower Tester sends a mesh message with IV Index set to the updated IV Index (m) to the IUT.
  11. The IUT accepts the message from the Lower Tester.
  12. The Lower Tester sends a mesh message with IV Index set to updated IV Index -1 (m-1) to the IUT.
  13. The IUT accepts and processes the message from the Lower Tester.
14. The Upper Tester sends a Transit to Normal signal to the IUT to transition to the Normal Operation state.
15. The Lower Tester expects to receive a Secure Network beacon from the IUT with the IV Update Flag set to 0 and the IV Index set to the new value (m).

• Expected Outcome

Pass verdict

The IUT accepts and processes the message from the Lower Tester signaling the start of the IV Update process.

The IUT transmits Secure Network beacons with the updated IV Index (m) while the IUT is in IV Update in Progress state.

In step 15, the IUT sends a Secure Network beacon with the IV Update Flag set to 0 and the IV Index set to the new IV Index value (m).

4.10.6 MESH/NODE/IVU/BI-01-C [Ignore IV Update Flag When Operating In Normal Operation For Less Than The Minimum Time]

• Test Purpose

Verify that an IUT that has been in Normal Operation state for less than 96 hours after an IV Update procedure has completed can ignore Secure Network beacons with the IV Update Flag set to 1.

• Reference

[3] Section 3.10.5

• Initial Condition

- The IUT is a Node.
- The Lower Tester is a Node.
- The IUT and the Lower Tester share previously established network security credentials.
- The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).

• Test Procedure

1. The Lower Tester generates a Secure Network beacon using the same network credentials and IV Index (n) as the IUT and begins advertising this beacon periodically, and scanning for neighboring beacons.
2. The Upper Tester orders the IUT to generate and send a Secure Network beacon with the same security as the Lower Tester.
3. The Lower Tester observes the Secure Network beacon advertisement(s) by the IUT.
4. The Lower Tester calculates a new Secure Network beacon with the IV Update Flag set to 1 and IV Index incremented by 1 (n+1), and begins to advertise the updated beacon.
5. The Upper Tester confirms that the IUT receives the Secure Network beacon.
6. The Lower Tester expects to receive Secure Network beacons from the IUT with the IV Update Flag set to 0 and IV Index set to the original IV Index value (n).
**Expected Outcome**

**Pass verdict**

In step 6, the IUT sends a Secure Network beacon with the IV Update Flag set to 0 and the IV Index set to the original IV Index (n).

### 4.10.7 MESH/NODE/IVU/BI-02-C [Ignore IV Update Flag When Operating In Normal Operation For an Abnormal IV Index]

**Test Purpose**

Verify that a Node IUT that has been in Normal Operation state for more than 96 hours can ignore Secure Network beacons with the IV Update Flag set to 1 if the IV Index in the Secure Network beacon is greater than the last known IV Index plus 42.

**Reference**

[3] Section 3.10.5

**Initial Condition**

- The IUT is a Node.
- The Lower Tester is a Node.
- The IUT and the Lower Tester share previously established network security credentials.
- The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
- IV Update test mode is activated on the IUT.
• Test Procedure

1. The Upper Tester orders the IUT to generate and send Secure Network beacons with the IV Update Flag set to 0 and the current IV Index value (n).
2. The Lower Tester generates a Secure Network beacon using the same NetKey, network credentials and IV Index (n) as the IUT and begins advertising this beacon periodically, while also scanning for neighboring beacons.
3. The Lower Tester observes the Secure Network beacon advertisement(s) by the IUT.
4. The Lower Tester calculates a new Secure Network beacon with the IV Update Flag set to 1, and IV Index is greater than the last known IV Index plus 42 (n+43), and begins to advertise the updated beacon.
5. The Upper Tester confirms that the IUT receives the Secure Network beacon.
6. The Lower Tester expects to receive Secure Network beacons from the IUT with the IV Update Flag set to 0 and the original IV Index (n).

Figure 4.20: MESH/NODE/IVU/BI-02-C

• Expected Outcome

Pass verdict

In step 5, the IUT continues sending Secure beacons with IV Update Flag = 0 and the original IV Index (n).

4.10.8 MESH/NODE/IVU/BI-04-C [Ignore IV Update Flag When Operating In Normal Operation For Message From Other Subnets]

• Test Purpose

Verify that a Node IUT that is a member of a primary subnet and has been in the Normal Operation state will ignore Secure Network beacons with the IV Update Flag set to 1 and an IV Index greater than the last known IV Index from other subnets.

• Reference

[3] Section 3.10.5
• **Initial Condition**
  - The IUT is a Node.
  - The IUT is provisioned to the primary Subnet.
  - The IUT has the Network Key of a Subnet that is not the primary Subnet.
  - The Lower Tester is a Node.
  - The Lower Tester is provisioned to a Subnet that is not the primary Subnet.
  - The IUT and the Lower Tester share previously established network security credentials for the same non-primary Subnet.
  - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
  - IV Update test mode is activated on the IUT.

• **Test Procedure**
  1. The Lower Tester generates a Secure Network beacon using the same network credentials and IV Index \(n\) as the IUT and begins advertising this beacon periodically, while scanning for neighboring beacons.
  2. The Upper Tester orders the IUT to send Secure Network beacons with the current IV Index value \(n\) and the IV Update Flag set to 0 (zero).
  3. The Lower Tester observes the Secure Network beacon advertisement(s) by the IUT.
  4. The Lower Tester calculates a new Secure Network beacon with the IV Update Flag set to 1, and IV Index is greater than the last known IV Index \((n+1)\), and begins to advertise the updated beacon from a different subnet.
  5. The Upper Tester confirms that the IUT receives the Secure Network beacon.
  6. The Lower Tester expects to receive a Secure Network beacon from the IUT with the IV Update Flag set to 0 and the IV Index set to the original value \(n\).

---

![Diagram](image-url)  
*Figure 4.21: MESH/NODE/IVU/BI-04-C*
• Expected Outcome
  Pass verdict
  In step 4, the IUT sends Secure Network beacons with IV Update Flag = 0 and the original IV Index (n).

4.10.9 MESH/NODE/IVU/BI-05-C [Ignore IV Update Flag When in Normal Operation state within 96 Hours of Exiting IV Update In Progress State]

• Test Purpose
  Verify that a Node IUT in Normal Operation state will ignore Secure Network beacons with the IV Update Flag set to 1 within 96 hours of completing an IV Update procedure.

• Reference
  [3] Section 3.10.5

• Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Node.
  - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update test mode is activated on the IUT.

• Test Procedure
  1. The IUT performs the IV Update procedure, e.g., by executing the procedure in MESH/NODE/IVU/BV-01-C [IV Update Initiated by Receiving Secure Network Beacons].
  2. After the IV Update procedure is complete, the IV Update test mode is deactivated on the IUT in order to set it to a state where it cannot accept a new IV Update procedure for at least 96 hours.
  3. The Upper Tester orders the IUT to generate and send Secure Network beacons with the IV Update Flag = 0 and the IV Index.
  4. The Lower Tester calculates a new Secure Network beacon with an IV Update Flag = 1 and IV Index = n+1, and begins to advertise the updated beacon.
  5. The Upper Tester confirms that the IUT receives the new Secure Network beacon from the Lower Tester.
  6. The Lower Tester expects to receive a Secure Network beacon from the IUT with the IV Update Flag set to 0 and the IV Index set to the original value (n).
**4.10.10 MESH/NODE/IVU/BI-06-C [IV Update – Invalid IV Index value]**

- **Test Purpose**
  Verify the IV Update procedure when the IUT is a node belonging to a primary Subnet and receives a Secure Network beacon with the IV Index set to a value that cannot be accepted during an IV Update procedure.

- **Reference**
  [3] Section 3.10.5, 3.10.5.1

- **Initial Condition**
  - The IUT is a Node.
  - The IUT is provisioned to the primary Subnet.
  - The Lower Tester is a Node.
  - The Lower Tester is provisioned to the primary Subnet.
- The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
- The IUT and the Lower Tester share previously established network security credentials.
- IV Update test mode is activated on the IUT.

**Test Procedure**

1. The Lower Tester generates a Secure Network beacon using the same network credentials and IV Index (n) as the IUT. The Lower Tester begins advertising this beacon periodically and is also scanning for neighboring beacons.
2. The IUT detects the Secure Network beacon using the correct IV Index value (n) from the Lower Tester.
3. The Upper Tester sends a Transit to IV Update in Progress signal to the IUT to transition to the IV Update in Progress state.
4. The Lower Tester expects the IUT to advertise a Secure Network beacon with the IV Index value incremented by one (m = n+1), the IV Update Flag set to 1 (IV Update in Progress). The Lower Tester validates the MIC based on the shared network security credentials.
5. The Lower Tester generates and advertises a Secure Network beacon using the same network credentials as the IUT and with the IV Index set to the value greater than the new IV Index (greater than n+1).
6. The Upper Tester confirms that the IUT receives the Secure Network beacon.
7. The Lower Tester expects the IUT to send the same Secure Network beacon as in step 4.

**Expected Outcome**

*Pass verdict*

The IUT ignores Secure Network beacons from the Lower Tester with an invalid IV Index value.

### 4.11 Key Refresh

The test group objective is to verify the Key Refresh procedure.

**4.11.1 MESH/NODE/KR/BV-01-C [Key Refresh Procedure as Node via Configuration Client Messages]**

**Test Purpose**

Verify that a Node IUT supporting the Configuration Server Model can participate in the Key Refresh procedure initiated by a Configuration Client.

**Reference**

[3] Section 3.10.4, 4.2.14, 4.3.2.58, 4.3.2.59, 4.3.2.60, 4.4.2.2.14

**Initial Condition**

- The IUT is a Node.
- The Lower Tester is a Configuration Client.
- The IUT and the Lower Tester share previously established network security credentials.
- The IUT is in normal operation. Key Refresh procedure is not in progress.
If a second subnet is supported, the IUT has at least one application key bound to the network key used in the second subnet.

- **Test Procedure**

  1. The Lower Tester sends a Config NetKey Update message to the IUT with a NetKeyIndex value of 0 and a new NetKey. The IUT stores the new key and continues advertising the same Secure Network beacon as in step 1.

  2. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key.

  3. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the old keys.

  4. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x02.

  5. The Lower Tester receives a Config Key Refresh Phase Status from the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x02 (Second phase).

  6. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.

  7. Repeat steps 4–6 to verify that the Phase remains at the second phase.

  8. The Lower Tester sends a Config Key Refresh Phase Set message to the Lower Tester with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x03.

  9. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x00.

  10. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT using the new keys.

  11. If a second subnet is supported, then repeat steps 1–10 using the second NetKeyIndex.

  12. Repeat steps 1–10, skipping step 2, using only the updated NetKey in the Key Refresh procedure.

  13. Repeat steps 1–3 and steps 8–10.
**Expected Outcome**

*Pass verdict*

In step 2, the IUT does not update any AppKey values when updating the NetKey value.

In step 5 and the repeated step from step 7, the IUT sends a Config Key Refresh Phase Status message with the Status of 0x00, the NetKeyIndex of the new NetKey, and Phase set to 0x02.

In step 9 and the repeated step from step 13, the IUT sends a Config Key Refresh Phase Status message to the Lower Tester with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x00.

All messages are encrypted and authenticated with the DevKey of the IUT.
4.11.2 MESH/NODE/KR/BV-02-C [Key Refresh Procedure as Node via Secure Network Beacons]

- **Test Purpose**
  Verify that a Node IUT can participate in the Key Refresh procedure where the phases are initiated by the receipt of Secure Network beacons.

- **Reference**
  [3] Section 3.9.3.1, 3.10.4

- **Initial Condition**
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
  - The IUT is in normal operation. Key Refresh procedure is not in progress.
  - If a second network key is supported, the IUT has at least one application key bound to it.

- **Test Procedure**
  1. The Lower Tester generates and advertises a Secure Network beacon using the IUT with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0, the Network ID set to the current Network ID, the IV Index set to the current IV Index, and the Authentication Value set to a calculated value using the values of the Flags field, Network ID field, and IV Index field.
  2. The Upper Tester orders the IUT to generate and send a Secure Network beacon using the same values as the Lower Tester uses in step 1.
  3. The Lower Tester sends a new NetKey to the IUT by sending a Config NetKey Update message to the IUT with a NetKeyIndex value of 0 and a new NetKey. The IUT stores the new key and continues advertising the same Secure Network beacon as in step 1.
  4. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key and continues advertising the same Secure Network beacon as in step 1.
  5. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
  6. The Lower Tester generates and advertises a new Secure Network beacon using the same network credentials as the IUT with the Key Refresh Flag (Flags bit 0) set to 1, the IV Update Flag (Flags bit 1) set to 0, the Network ID set to the current Network ID, the IV Index set to the current IV Index, and the Authentication Value set to a newly calculated value using the values of the Flags field, Network ID field, and IV Index field.
  7. The Lower Tester expects to receive a new Secure Network beacon from the IUT with the following field values: the Key Refresh Flag (Flags bit 0) is set to 1, the IV Update Flag (Flags bit 1) is set to 0, the Network ID is set to the current Network ID, the IV Index is set to the current IV Index, and the Authentication value is set to the computed value. The AD Type is formatted correctly.
  8. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
9. The Lower Tester generates and transmits a new Secure Network beacon with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0, the Network ID is set to the current Network ID, the IV Index set to the current IV Index, and the Authentication Value calculated with the new NetKey.

10. The Lower Tester expects to receive a new Secure Network beacon from the IUT using the new credentials with the Key Refresh Flag set to 0 and the Network ID set to the current Network ID, and a new Authentication Value.

11. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT using the new keys.

12. If a second subnet is supported, then repeat steps 1–11 using the second NetKeyIndex.

13. Repeat steps 1–11, skipping step 4, using only the updated NetKey in the Key Refresh procedure.

---

Figure 4.24: MESH/NODE/KR/BV-02-C
• Expected Outcome
  Pass verdict

  In step 2, the IUT transmits a correctly formatted Secure Network beacon with valid values.

  In step 3, the IUT does not update any AppKey values when updating the NetKey value.

  In step 7, the IUT sends Secure Network beacons with the Key Refresh Flag enabled and a new
  Authentication Value.

  In step 10, the IUT sends Secure Network beacons using the new credentials with the Key Refresh
  Flag set to 0, the Network ID set to the current Network ID, and a new Authentication Value.

  All messages are encrypted and authenticated with the DevKey of the IUT.

4.11.3 MESH/NODE/KR/BV-03-C [Key Refresh Procedure Skipping Phase 2]

• Test Purpose
  Verify that a Node IUT in Phase 1 of the Key Refresh procedure can skip Phase 2 of the Key Refresh
  procedure when it receives a Secure Network beacon with the Key Refresh flag set to 0 and the new
  NetKey.

• Reference
  [3] Section 3.9.3.1, 3.10.4.1

• Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT is in normal operation. Key Refresh procedure is not in progress.

• Test Procedure
  1. Execute steps 1–5 of MESH/NODE/KR/BV-01-C [Key Refresh Procedure as Node via
     Configuration Client Messages].
  2. The Lower Tester generates and sends a new Secure Network beacon using the same network
     credentials as the IUT with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags
     bit 1) set to 0, the Network ID set to the current Network ID, the IV Index set to the current IV
     Index, and the Authentication Value set to a new value calculated with the new NetKey.
  3. The Lower Tester expects to receive an updated Secure Network beacon from the IUT with the
     new security credentials, the Key Refresh flag set to 0, and the Network ID set to the current
     Network ID.
• Expected Outcome

Pass verdict

In step 3, the IUT sends a Secure Network beacon with the same network credentials as Lower Tester with the Key Refresh flag set to 0, the Network ID set to the current Network ID, and the Authentication Value set to a new value calculated with the new NetKey.

4.11.4 MESH/CFGCL/KR/BV-01-C [Key Refresh as Configuration Client via Configuration Messages]

• Test Purpose

Verify that an IUT supporting the Configuration Client Model can perform the Key Refresh procedure with a Node supporting the Configuration Server Model. The test procedure is executed with the Relay feature disabled and again with the Relay feature enabled if supported.

• Reference

[3] Section 3.10.4, 4.2.14, 4.3.2.58, 4.3.2.59, 4.3.2.60, 4.4.2.2.14

• Initial Condition

- The IUT is a Configuration Client.
- The Lower Tester is a Node.
- The IUT and the Lower Tester share previously established network security credentials.

• Test Procedure

1. The Upper Tester orders the IUT to transmit a Config Key Refresh Phase Get message to the Lower Tester with the NetKeyIndex set to the current Network Key.
2. The Upper Tester confirms that the IUT receives a Config Key Refresh Phase Status message in response from the Lower Tester using the correct keys with the Status set to 0x00 and the Phase set to 0x00 (Normal operation).
3. The Upper Tester orders the IUT to send a new NetKey to the Lower Tester using the procedure in MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages]. The Lower Tester stores the new key and continues advertising the same Secure Network beacon as in step 1.

4. The Upper Tester orders the IUT to send a new AppKey to the Lower Tester using the procedure in MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages].

5. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the old keys.

6. The Upper Tester orders the IUT to send a Config Key Refresh Phase Set message with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x02.

7. The Lower Tester sends a Config Key Refresh Phase Status message to the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the NetKey, and the Phase set to 0x02.

8. The Upper Tester orders the IUT to send a Config Key Refresh Phase Set message to the Lower Tester with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x03.

9. The Lower Tester transmits a Config Key Refresh Phase Status message to the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x00.

10. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
* Expected Outcome

**Pass verdict**

The IUT transmits new keys to the Lower Tester.

In step 6, the IUT transmits a Config Key Refresh Phase Set message with the NetKeyIndex for the new NetKey, and the Transition set to 0x02. The IUT receives a Config Key Refresh Phase Status message in response from the Lower Tester with Status of 0x00, the new key, and Phase set to 0x02.
In step 8, the IUT transmits a Config Key Refresh Phase Set message with the NetKeyIndex for the new NetKey, and Transition set to 0x02 and receives a Config Key Refresh Phase Status message in response from the Lower Tester with Status of 0x00, the NetKeyIndex for the new key, and Phase set to 0x00.

All messages are encrypted and authenticated with the DevKey of the Lower Tester.

4.11.5 MESH/CFGCL/KR/BV-02-C [Key Refresh as Configuration Client via Secure Network Beacons]

• Test Purpose
Verify that an IUT supporting the Configuration Client Model can perform the Key Refresh procedure with a Node supporting the Configuration Server Model. The test procedure is executed with the Relay feature enabled (if supported) and with the Relay feature disabled.

• Reference
[3] Section 3.9.3.1, 3.10.4

• Initial Condition
- The IUT is a Configuration Client.
- The Lower Tester is a Node.
- The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
- The IUT and the Lower Tester share previously established network security credentials.

• Test Procedure
1. The IUT is induced to send a new NetKey to the Lower Tester, using the procedure in MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages]. The Lower Tester stores the new key and continues advertising the same Secure Network beacon as in step 1.
2. The IUT is induced to send a new AppKey to the Lower Tester, using the procedure in MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages]. The Lower Tester stores the new key and continues advertising the same Secure Network beacon as in step 1.
3. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
4. The Upper Tester orders the IUT to generate and advertise a new Secure Network beacon with the Key Refresh Flag (Flags bit 0) set to 1, the IV Update Flag (Flags bit 1) set to 0, the Network ID set to the current Network ID, the IV Index set to the current IV Index, and the Authentication Value set to a newly calculated value using the values of the Flags field, Network ID field, and IV Index field.
5. The Lower Tester receives the new Secure Network beacon.
6. The Lower Tester generates and sends a new Secure Network beacon with the Key Refresh Flag set to 1 and the Authentication Value set to a new value calculated with the new NetKey.
7. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
8. The Upper Tester orders the IUT to generate and advertise a new Secure Network beacon with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0, the Network ID set to the current Network ID, the IV Index set to the current IV Index, and the Authentication
Value set to a newly calculated value using the values of the Flags field, Network ID field, and IV Index field.

9. The Lower Tester expects to receive the new Secure Network beacon from the IUT. Upon receipt, it discards the old credentials and begins transmitting Secure Network beacons using the new credentials with the Key Refresh bit cleared and the Network ID set to the current Network ID.

10. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.

*Figure 4.27: MESH/CFGCL/KR/BV-02-C*
• Expected Outcome

Pass verdict

The IUT receives a Secure Network beacon from the Lower Tester that is formatted correctly and with valid values.

The IUT sends a new NetKey to the Lower Tester.

The IUT sends a new AppKey to the Lower Tester.

The IUT transmits a Secure Network beacon to the Lower Tester with the Key Refresh Flag enabled and begins advertising Secure Network beacons with the Key Refresh Flag enabled and a new Authentication Value.

The IUT transmits a new Secure Network beacon with the expected values. The IUT revokes the old keys.

All messages are encrypted and authenticated with the DevKey of the Lower Tester.

4.11.6 MESH/CFGCL/KR/BV-03-C [Key Refresh with Blacklisting via Configuration Messages]

• Test Purpose

Verify an IUT acting as Configuration Client can perform the Key Refresh procedure using configuration messages while excluding nodes that are blacklisted by the IUT.

• Reference

[3] Section 3.10.4

• Initial Condition

- The IUT is a Configuration Client.
- Lower Tester 1 is a Node.
- Lower Tester 2 is a Node.
- The Relay feature is not enabled on the Lower Tester 1 (if Relay Mode is supported, the Relay state is set to 0x00, otherwise it is set to 0x02).
- The IUT and both Lower Testers share previously established network security credentials.

• Test Procedure

1. The Upper Tester orders the IUT to send a new NetKey to the IUT using the procedure in MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Message].
2. The Upper Tester orders the IUT to send a new AppKey to Lower Tester 1 using the procedure in MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Message].
3. The Upper Tester orders the IUT to send a Config Key Refresh Phase Set message to Lower Tester 1 with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition field set to 0x02.
4. Lower Tester 1 sends a Config Key Refresh Phase Status message to the IUT with the Status set to 0x00, the NetKeyIndex set to the index of the new NetKey, and the Phase set to 0x02.
5. The Upper Tester orders the IUT to send a Config Key Refresh Phase Set message to Lower Tester 1 with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition field set to 0x03.

6. Lower Tester 1 sends a Config Key Refresh Phase Status message to the IUT with the Status set to 0x00, and the Phase set to 0x00.

7. Lower Tester 2 sends a mesh message to the IUT. The Upper Tester confirms that the IUT ignores the message.

8. If Relay feature is supported by Lower Tester 1, set its Relay state to 0x01 and repeat steps 1-7 with Lower Tester 1 acting as a Relay node.

• Expected Outcome

Pass verdict

The IUT sends new keys to Lower Tester 1.

In step 3, the IUT transmits a Config Key Refresh Phase Set message with the NetKeyIndex for the new NetKey, and Transition set to 0x02 and receives a Config Key Refresh Phase Status message in response from the Lower Tester with Status of 0x00, the new key, and Phase set to 0x02.

In step 5, the IUT transmits a Config Key Refresh Phase Set message with the NetKeyIndex for the new NetKey, and Transition set to 0x03 and receives a Config Key Refresh Phase Status message in response from the Lower Tester with Status of 0x00, the new key, and Phase set to 0x00.

In step 7, the IUT ignores the mesh message.

All messages are encrypted and authenticated with the DevKey of the Lower Tester.

4.11.7 MESH/CFGCL/KR/BV-04-C [Key Refresh with Blacklisting via Secure Network Beacons]

• Test Purpose

Verify an IUT acting as Configuration Client can perform the Key Refresh procedure using Secure Network beacons while excluding nodes that are blacklisted by the IUT.

• Reference

[3] Section 3.9.3.1, 3.10.4

• Initial Condition

- The IUT is a Configuration Client.
- Lower Tester 1 is a Node.
- Lower Tester 2 is a Node that has been blacklisted.
- The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
- The Relay feature is not enabled on the Lower Tester 1 (if Relay Mode is supported, the Relay state is set to 0x00, otherwise it is set to 0x02).
- The IUT and both Lower Testers share previously established network security credentials.
• **Test Procedure**

1. Lower Tester 1 advertises Secure Network beacons (Mesh beacon with Beacon Type set to 0x01) with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0, the Network ID set to the current Network ID, the IV Index set to the current IV Index, and the Authentication Value set to the computed value.

2. The IUT is induced to send a new AppKey to Lower Tester 1, using the procedure in MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Message]. The IUT sends a new NetKey to the IUT using the procedure in MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Message]. Lower Tester 1 stores the new keys and continues advertising the same Secure Network beacon as in step 1. Lower Tester 2 does not receive new keys.

3. The Upper Tester orders the IUT to generate and send a new Secure Network beacon with the Key Refresh Flag set to 1 and the Authentication Value set to a new value calculated with the new NetKey.

4. Lower Tester 1 receives the new Secure Network beacon from the IUT. Lower Tester 1 begins advertising a new Secure Network beacon with the Key Refresh Flag set to 1 and the Authentication Value set to a newly calculated value using the values of the Flags field, Network ID field, and IV Index field.

5. Lower Tester 2 does not expect to receive the new Secure Network beacon from the IUT.

6. The Upper Tester orders the IUT to generate and send a new Secure Network beacon with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0, the Network ID set to the new Network ID, the IV Index set to the current IV Index, and the Authentication Value calculated with the new NetKey.

7. Lower Tester 1 expects to receive the new Secure Network beacon from the IUT. Upon receipt, it discards the old credentials and begins transmitting Secure Network beacons using the new credentials with the Key Refresh bit cleared and the Network ID set to the current Network ID.

8. Lower Tester 2 does not expect to receive the new Secure Network beacon.

9. If Relay feature is supported by Lower Tester 1, set its Relay state to 0x01 and repeat steps 1–8 with Lower Tester 1 acting as a Relay node.

• **Expected Outcome**

**Pass verdict**

The IUT sends new keys to Lower Tester 1.

In step 3, the IUT sends a new Secure Network beacon with the Key Refresh Flag enabled and a new Authentication Value.

In step 6, the IUT sends a new Secure Network beacon with the Key Refresh Flag set to 0 and a new Authentication Value.

**4.11.8 MESH/NODE/KR/BI-01-C [Receive Config Key Refresh Phase Set Messages with Valid and Invalid Transition Values]**

• **Test Purpose**

Verify that a Node IUT participating in the Key Refresh procedure responds appropriately to Config Key Refresh Phase Set messages with valid and invalid Transition field values during each Key Refresh Phase.

• **Reference**

[3] Section 3.1.2, 3.10.4, 4.3.2.59, 4.3.2.60
• Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The Lower Tester has previously discovered the ElementAddress and Model Identifier of the Mesh Configuration Model and using that ElementAddress for all communication.
  - Key Refresh procedure is not in progress (Key Refresh Phase=0x00) on the IUT.

• Test Procedure
  1. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x01.
  2. The IUT ignores the Config Key Refresh Phase Set message from the Lower Tester and the Lower Tester expects no response.
  3. Repeat step 1 substituting a Transition field value of 0x02.
  4. The IUT ignores the Config Key Refresh Phase Set message from the Lower Tester and the Lower Tester expects no response.
  5. Repeat step 1 substituting a Transition field value of 0x03.
  6. The Lower Tester receives a Config Key Refresh Phase Status message with the Status set to 0x00, the NetKeyIndex set to the value sent in the Config Key Refresh Phase Set message and the Phase field set to 0x00 (Normal operation).
  7. To begin Phase 1 of Key Refresh procedure, the Lower Tester sends a new NetKey to the IUT using the procedure in MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message]. The IUT stores the new key.
  8. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key.
  9. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x00.
 10. The IUT ignores the Config Key Refresh Phase Set message from the Lower Tester and the Lower Tester expects no response.
 11. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x02.
 12. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00 (Success), the NetKeyIndex set to the value sent in the Config Key Refresh Phase Set message and the Phase set to 0x02 (Second phase).
 13. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x01.
 14. The IUT ignores the Config Key Refresh Phase Set message from the Lower Tester and the Lower Tester expects no response.
 15. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x02.
16. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00, the NetKeyIndex set to the value sent in the Config Key Refresh Phase Set message and the Phase set to 0x02 (Second phase).

*Expected Outcome*

**Pass verdict**

In step 2, the IUT does not transmit a Config Key Refresh Phase Status message.

_Figure 4.28: MESH/NODE/KR/BI-01-C_
In step 4, the IUT does not transmit a Config Key Refresh Phase Status message.

In step 6, the IUT transmits a Config Key Refresh Phase Status message with the Status set to 0x00 and the Phase set to 0x00.

In step 10, the IUT does not transmit a Config Key Refresh Phase Status message.

In step 12, the IUT transmits a Config Key Refresh Phase Status message with the Status set to 0x00 and the Phase set to 0x02.

In step 14, the IUT does not transmit a Config Key Refresh Phase Status message.

In step 16, the IUT transmits a Config Key Refresh Phase Status message with the Status set to 0x00 and the Phase set to 0x02.

**4.11.9 MESH/NODE/KR/BI-02-C [Ignore New Keys During Key Refresh Procedure]**

- **Test Purpose**
  Verify that a Node IUT supporting the Configuration Server Model and participating in the Key Refresh procedure will ignore Config NetKey Update and Config AppKey Update messages after Phase 1 of Key Refresh Procedure is complete.

- **Reference**
  [3] Section 3.10.4, 4.3.2.32, 4.3.2.34, 4.3.2.38, 4.3.2.40, 4.3.2.59, 4.3.2.60

- **Initial Condition**
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - Key Refresh procedure is not in progress (Key Refresh Phase=0x00) on the IUT.

- **Test Procedure**
  1. The Lower Tester attempts to send a new AppKey to the IUT using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT does not store the new AppKey value.
  2. The Lower Tester receives an Config AppKey Status from the IUT with the Status field set to 0x0B (Cannot Update).
  3. To begin Phase 1 of Key Refresh procedure, the Lower Tester sends a new NetKey to the IUT using the procedure in MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message]. The IUT stores the new key.
  4. The Lower Tester receives a Config NetKey Status message with the Status field set to 0x00.
  5. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key.
  6. The Lower Tester receives an Config AppKey Status message with the Status field set to 0x00.
  7. The Lower Tester attempts to send a new NetKey to the IUT using the procedure in MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message].
8. The Lower Tester receives a Config NetKey Status message from the IUT with the Status field set to 0x0B (Cannot Update).

9. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition set to 0x02.

10. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00, the NetKeyIndex set to the value sent in the Config Key Refresh Phase Set message and the Phase set to 0x02 (Second phase).

11. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The Lower Tester sends a new NetKey to the IUT using the procedure in MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message]. The IUT does not store the new keys.

12. The Lower Tester receives from the IUT an Config AppKey Status message with the Status field set to 0x0B and a Config NetKey Status message with the Status field set to 0x0B.

**Figure 4.29: MESH/NODE/KR/BI-02-C**

- Expected Outcome

  **Pass verdict**

In step 2, the IUT sends a Config AppKey Status message with the Status field set to 0x0B.
In step 4, the IUT sends a Config NetKey Status message with the Status field set to 0x00.

In step 6, the IUT sends an Config AppKey Status message with the Status field set to 0x00.

In step 8, the IUT sends a Config NetKey Status message with the Status field set to 0x0B.

In step 10, the IUT sends a Config Key Refresh Phase Status message.

In step 12, the IUT sends an Config AppKey Status with the Status field set to 0x0B and a Config NetKey Status message with the Status field set to 0x0B.

4.11.10 MESH/NODE/KR/BI-03-C [Receive Config Key Refresh Phase Get with Invalid NetKeyIndex]

• Test Purpose
Verify that a Node IUT can respond to a Config Key Refresh Phase Get message with an invalid NetKeyIndex value with a Config Key Refresh Phase Status message containing the correct Status code.

• Reference
[3] Section 3.9.3.1, 3.10.4, 4.4.1.2.14

• Initial Condition
- The IUT is a Node.
- The Lower Tester is a Configuration Client.
- The IUT and the Lower Tester share previously established network security credentials.
- The IUT is in normal operation. Key Refresh procedure is not in progress.

• Test Procedure
1. The Lower Tester sends a Config Key Refresh Phase Get message to the IUT with the NetKeyIndex set to a value higher than the largest NetKey Index on the IUT.
2. The Lower Tester expects a Config Key Refresh Phase Status message in response from the IUT with the Status set to 0x04.

• Expected Outcome
Pass verdict

In step 2, the IUT sends a Config Key Refresh Phase Status message with the Status set to 0x04.

4.11.11 MESH/NODE/KR/BI-04-C [Receive Messages with Old Keys During Configuration Client Key Refresh]

• Test Purpose
Verify that a Node IUT supporting the Configuration Server Model can participate in the Key Refresh procedure initiated by a Configuration Client.

• Reference
[3] Section 3.10.4, 4.2.14, 4.3.2.58, 4.3.2.59, 4.3.2.60, 4.4.2.2.14
• Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT is in normal operation. Key Refresh procedure is not in progress.

• Test Procedure
  1. The Lower Tester transmits a Config Key Refresh Phase Get message to the IUT with the NetKeyIndex set to the current Network Key and expects a Config Key Refresh Phase Status message in response from the IUT with the Phase set to 0x00 (Normal operation).
  2. The Lower Tester sends a Config NetKey Update message to the IUT with a NetKeyIndex value of 0 and a new NetKey. The IUT stores the new key and continues advertising the same Secure Network beacon as in step 1.
  3. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key.
  4. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
  5. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x02.
  6. The Lower Tester receives a Config Key Refresh Phase Status from the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x02 (Second phase).
  7. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
  8. The Lower Tester sends a Config Key Refresh Phase Set message to the Lower Tester with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x03.
  9. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x00.
  10. The Lower Tester sends a mesh message with the old keys to the IUT and receives no response from the IUT.

• Test Procedure
  Pass verdict
  In step 4, the IUT sends a reply to the Lower Tester using the old keys.
  In step 7, the IUT sends a reply to the Lower Tester using the new keys.
  In step 10, the IUT does not send a reply to the Lower Tester.
  All messages are encrypted and authenticated with the DevKey of the IUT.
4.11.12 MESH/NODE/KR/BI-05-C [Receive Messages with Old Keys During Secure Network Beacon Key Refresh]

- **Test Purpose**
  Verify that a Node IUT can participate in the Key Refresh procedure where the phases are initiated by the receipt of Secure Network beacons.

- **Reference**
  [3] Section 3.9.3.1, 3.10.4

- **Initial Condition**
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state = 0x01).
  - The IUT is in normal operation. Key Refresh procedure is not in progress.

- **Test Procedure**
  1. The Lower Tester generates and advertises a Secure Network beacon using the IUT with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0, the Network ID set to the current Network ID, the IV Index set to the current IV Index, and the Authentication Value set to a calculated value using the values of the Flags field, Network ID field, and IV Index field.
  2. The Upper Tester orders the IUT to generate and send a Secure Network beacon using the same values as the Lower Tester uses in step 1.
  3. The Lower Tester sends a new NetKey to the IUT by sending a Config NetKey Update message to the IUT with a NetKeyIndex value of 0 and a new NetKey. The IUT stores the new key and continues advertising the same Secure Network beacon as in step 1.
  4. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key and continues advertising the same Secure Network beacon as in step 1.
  5. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
  6. The Lower Tester generates and advertises a new Secure Network beacon using the same network credentials as the IUT with the Key Refresh Flag (Flags bit 0) set to 1, the IV Update Flag (Flags bit 1) set to 0, the Network ID set to the current Network ID, the IV Index set to the current IV Index, and the Authentication Value set to a newly calculated value using the values of the Flags field, Network ID field, and IV Index field.
  7. The Lower Tester expects to receive a new Secure Network beacon from the IUT with the following field values: the Key Refresh Flag (Flags bit 0) is set to 1, the IV Update Flag (Flags bit 1) is set to 0, the Network ID is set to the current Network ID, the IV Index is set to the current IV Index, and the Authentication value is set to the computed value. The AD Type is formatted correctly.
  8. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
  9. The Lower Tester generates and transmits a new Secure Network beacon with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0, the Network ID is set to the
current Network ID, the IV Index set to the current IV Index, and the Authentication Value calculated with the new NetKey.

10. The Lower Tester expects to receive a new Secure Network beacon from the IUT using the new credentials with the Key Refresh Flag set to 0 and the Network ID set to the current Network ID, and a new Authentication Value.

11. The Lower Tester sends a mesh message with the old keys to the IUT and receives no response.

- Expected Outcome
  
  **Pass verdict**

  In step 5, the IUT sends a message to the Lower Tester authenticated using the old keys.

  In step 8, the IUT sends a message to the Lower Tester authenticated using the new keys.

  In step 11, the IUT does not send a response to the Lower Tester.

  All messages are encrypted and authenticated with the DevKey of the IUT.

### 4.12 Friendship

The test group objective is to verify functionality related to the Friend feature, including Friend Establishment, Friend Messaging, and Friend Management.

#### 4.12.1 Friend Node

This section contains tests related to behavior from the perspective of a node participating in Friendship procedures as a Friend node.

##### 4.12.1.1 MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]

- **Test Purposes**
  
  Verify that an IUT node supporting the Friend feature can perform the Friendship Establishment procedure with a Low Power Node.

- **Reference**
  
  [3] Section 3.6.6, 3.6.5.1, 3.6.5.2, 3.6.5.3, 3.6.5.4, 3.6.6.3.1, 4.2.2.4

- **Initial Condition**
  
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester is a node that supports the Low Power Feature.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- **Test Procedure**
  
  1. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set
to the all-friends address, a TTL field value of 0x00, Criteria field value of 0x21, ReceiveDelay value of 0xA0, PollTimeout value of 0x00012C, a LPNCounter field set to 0x00, NumElements field value set to 0x01, and a PreviousAddress field value set to 0x0000. Lower Tester then waits 100 ms.

2. The Lower Tester expects to receive a Friend Offer message. The Friend Offer message should be sent after the Friend Offer Delay which is calculated by using the RSSI factor and Received Window Factor.

3. The Lower Tester sends a Friend Poll message addressed to the IUT within 1 second of receiving the Friend Offer message with an FSN field value set to 0. The Friend Poll message is secured using the friendship security credentials found in [3] Section 3.8.5.3.1.

4. The Lower Tester expects to receive a Friend Update message from the IUT.

![Diagram](Figure 4.30: MESH/NODE/FRND/FN/BV-01-C)

- **Expected Outcome**

**Pass verdict**

The IUT sends a Friend Offer message to the Lower Tester’s unicast address, with a TTL value of 0, a TCM Message Opcode value of 0x04, a ReceiveWindow field value between 0x01 – 0xFF, RSSI value between 0x00 – 0xFF, and a FriendshipCounter field with valid 2-byte value starting at 1 for first Friend Offer and incrementing by 1 for each additional.

The IUT sends the Friend Offer message after the calculated Friend Offer Delay.

The IUT sends a Friend Update message within the [ReceiveDelay, ReceiveDelay + ReceiveWindow] time interval from the Friend Poll message from the Lower Tester.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

**Fail verdict**

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.
4.12.1.2 MESH/NODE/FRND/FN/BV-02-C [Friendship Messaging – Friend Node]

- **Test Purpose**
  Verify that an IUT node acting as a Friend Node can cache Unsegmented Access messages and report them to an LPN.

- **Reference**
  [3] Section 3.1.2, 3.5.2.1, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

- **Initial Condition**
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1

- **Test Procedure**
  1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time after the friendship establishment in step 1, the Transport Control Message Opcode field is set to 0x01, the RFU bits set to 0, TTL field set to 0x00, and FSN bit toggled for each new Friend Poll message if the previous Friend Poll message has triggered a response.
  3. The Lower Tester 1 expects a Friend Update message from the IUT with the MD field set to 0x00.
  4. The Lower Tester 2 sends an Unsegmented Access message to the Lower Tester 1’s address emulating the LPN and expects the IUT to cache this message.
  5. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
  6. The Lower Tester 1 expects to start receiving the cached message from Lower Tester 2 sent in step 4. The Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the message sent in step 4 to the Lower Tester 1. The Lower Tester 1 will continue sending the Friend Poll message until it receives a Friend Update message with the MD field set to 0x00.
  7. The Lower Tester 1 and 2 repeat steps 2–6 for 3 more cycles, alternating between one and N messages, where N can be any value from 2 to 10 and alternating the FSN bit for each Friend Poll message.
  8. The Lower Tester 2 sends a message to the IUT with a different destination address as Lower Tester 1 and the IUT is expected to ignore and not pass this message to the Lower Tester 1 acting as the LPN.
**Figure 4.31: MESH/NODE/FRND/FN/BV-02-C**

- **Expected Outcome**

  **Pass verdict**

  In step 2, the IUT sends a Friend Update message within the interval \([\text{ReceiveDelay}, \text{ReceiveDelay}+\text{ReceiveWindow}]\) from the Friend Poll, with the MD field set to 0x00, the TransOpCode field set to 0x02, and the Key Refresh and IV Index flags set to 0.

  The IUT successfully transmits all cached messages to the Lower Tester 1.
The IUT does not cache the message sent by Lower Tester 2 in step 8 and does not relay the message to the Lower Tester 1.

The IUT sends each cached message within the \([\text{ReceiveDelay}, \text{ReceiveDelay} + \text{ReceiveWindow}]\) time interval from the Friend Poll message. When there are no more cached messages, the IUT sends a Friend Update message with the MD field set to 0x00.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

**Fail verdict**

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

### 4.12.1.3 MESH/NODE/FRND/FN/BV-03-C [Segment Acknowledgement – Friend Node]

**Test Purpose**

Verify that an IUT node acting as a Friend Node can respond to segmented messages destined to an LPN, cache and deliver the Segmented Access message to the LPN.

**Reference**

[3] Section 3.1.2, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1, 3.6.6.5, 3.6.6.5.1

**Initial Condition**

- The IUT is a node that supports the Friend Feature and the feature is enabled.
- The Lower Tester 1 is a node that supports the Low Power Feature.
- Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

**Test Procedure**

1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. The Lower Tester 2 sends a message as in test MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast] except the unicast destination address is that of Lower Tester 1.
3. The Lower Tester 2 expects the IUT to perform the SAR operation on the message on behalf of the Lower Tester 1, for the entire test case.
4. The Lower Tester 2 expects the OBO bit of all Segment Acknowledgement messages to be set to 1 indicating it is acknowledging this message on behalf of an LPN.
5. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time after the friendship establishment in step 1, the Transport Control Message Opcode field set to 0x01, RFU
bits set to 0, TTL field set to 0x00, and FSN bit toggled for each new Friend Poll message if the previous Friend Poll message has triggered a response.

6. The Lower Tester 1 expects to start receiving the cached fragments of a Segmented Access Message from Lower Tester 2 sent in step 2. The Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the all fragments of a Segmented Access message sent in step 2 to the Lower Tester 1 with the expected contents. The Lower Tester 1 will continue sending the Friend Poll message until it receives a Friend Update message with the MD field set to 0x00.

---

**Expected Outcome**

**Pass verdict**

The IUT receives the segments from the Lower Tester 2 and preserves them in the order they were received.
The IUT sends all Segment Acknowledgement messages with the OBO bit set to 1 and SRC set to the address of Lower Tester 1 to the Lower Tester 2.

The IUT successfully transmits all cached segments of the Segmented Access Message to the Lower Tester 1 and then it sends a Friend Update message with the MD field set to 0x00.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

### 4.12.1.4 MESH/NODE/FRND/FN/BV-04-C [Friend IV Update – Friend Node]

- **Test Purpose**
  
  Verify that an IUT node acting as a Friend Node can properly handle an IV update procedure.

- **Reference**
  
  [3] Section 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3, 3.6.6.4.1, 3.8.3, 3.10.5

- **Initial Condition**
  
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending Secure Network Beacons.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - IV Index update is not already in progress.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- **Test Procedure**
  
  1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll has a TTL field set to 0x00.
  3. The Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester and indicating the current IV Index is still in use.
  4. The Upper Tester induces the IUT to ignore IV Update procedure time constrains.
  5. The Lower Tester 2 begins to broadcast a Secure Network Beacon, emulating a different device on the same network, with the IV Index incremented by one and the IV Update Flag set to 1.
  6. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout and with the new IV Index minus 1. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
7. The Lower Tester 1 expects a Friend Update message from the IUT indicating the IV Index has been incremented.
8. The Lower Tester 2 begins to broadcast a Secure Network Beacon, emulating a different device on the same network, with the IV Index and the IV Update Flag set to 0.
9. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout and with the new IV Index - 1. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
10. The Lower Tester 1 expects a Friend Update message from the IUT.

Figure 4.33: MESH/NODE/FRND/FN/BV-04-C

- Expected Outcome

Pass verdict

The IUT communicates the updated IV Index to the Lower Tester with the MD bit set to 0 and, the IV Update Flag set to 1, and the Key Refresh flag set to 0 in the Friend Update message after receiving the updated IV Index via Secure Network Beacon in step 5. The Friend Update message is using old IV Index value indicated by IVI bit.
The IUT communicates the updated IV Index to the Lower Tester with the MD bit set to 0 and, the IV Update Flag set to 0, and the Key Refresh flag set to 0 in the Friend Update message in step 10 after receiving the updated IV Index via Secure Network Beacon in step 8. The Friend Update message is using new IV Index value indicated by IV bit.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

4.12.1.5 MESH/NODE/FRND/FN/BV-05-C [Friendship – Respond to Friend Subscription List Add Message]

• Test Purpose

Verify that an IUT acting as a Friend Node can respond to and act conformingly upon reception of the Friend Subscription List Add message.

• Reference

[3] Section 3.6.5.1, 3.6.5.2, 3.6.5.7, 3.6.5.9, 3.6.6, 3.6.6.3.3

• Initial Condition

- The IUT is a node that supports the Friend Feature and the feature is enabled.
- The Lower Tester 1 is a node that supports the Low Power Feature.
- The Lower Tester 2 is acting as another (3rd) node sending messages to the LPN.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Lower Tester and IUT have previously established a friend relationship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure

1. The Lower Tester 2 sends an Unsegmented Access Message to one group address from the Group Address(es) and expects the IUT to ignore this message.
2. The Lower Tester 1 sends a Friend Poll message to the IUT within the ReceiveDelay time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
3. The Lower Tester 1 expects a Friend Update message from the IUT indicating it has no message for the Lower Tester 1 in cache.
4. The Lower Tester 1 sends a Friend Subscription List Add message to the IUT with the AddressList field value containing a valid Group Address(es) value(s) and a random TransactionNumber field value.
5. The Lower Tester 1 expects to receive a Friend Subscription List Confirm message with a TransactionNumber field equal to that of the value it sent in step 4 from the IUT.
6. The Lower Tester 2 sends an Unsegmented Access Message to one group address from the Group Address(es) and expects the IUT to cache this message.

7. The Lower Tester 1 sends a Friend Poll message to the Lower Tester 1 within the “PollTimeout” time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.

8. The Lower Tester 1 expects a message from the IUT and confirms that it retransmits the message sent in step 6 to the Lower Tester 1 with the expected contents.

![Diagram](image)

Figure 4.34: MESH/NODE/FRND/FN/BV-05-C

- Expected Outcome

  **Pass verdict**

  The IUT sends a Friend Subscription List Confirm message within the interval \([\text{ReceiveDelay}, \text{ReceiveDelay}+\text{ReceiveWindow}]\) and with a TransactionNumber field equal to that sent in step 2.

  The IUT caches messages for the Lower Tester when they are addressed to the Lower Tester’s Group Address and relays those messages to the Lower Tester when requested.

  All Friend Poll, Friend Update, Friend Subscription List Add, and Friend Subscription List Confirm messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  **Fail verdict**

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.
4.12.1.6 MESH/NODE/FRND/FN/BV-06-C [Friendship – Respond to Friend Subscription List Remove Message]

- **Test Purpose**
  Verify that an IUT node acting as a Friend Node can respond to and act conformingly upon reception of the Friend Subscription List Remove message.

- **Reference**
  [3] Section 3.6.5.1, 3.6.5.2, 3.6.5.7, 3.6.5.8, 3.6.5.9, 3.6.6, 3.6.6.3.3

- **Initial Condition**
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - The Lower Tester 2 is another node on the network.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Lower Tester and IUT have executed test MESH/NODE/FRND/FN/BV-05-C [Friendship – Respond to Friend Subscription List Add Message].
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- **Test Procedure**
  1. The Lower Tester 1 sends a Friend Subscription List Remove message to the IUT with the AddressList field containing the same values in the Friend Subscription List Add Message and the TransactionNumber field equal to 2.
  2. The Lower Tester 1 expects to receive a Friend Subscription List Confirm message from the IUT with a TransactionNumber field equal to 2.
  3. The Lower Tester 2 sends an Unsegmented Access Message to one group address from Group Address(es) and expects the IUT not to cache this message.
  4. The Lower Tester 1 sends a Friend Poll message to the IUT within the ReceiveDelay time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  5. The Lower Tester 1 expects a Friend Update message from the IUT with the MD field set to 0x00.
Expected Outcome

Pass verdict

The IUT sends a Friend Subscription List Confirm message within the ReceiveWindow time window of the Lower Tester and with a TransactionNumber field equal to 2.

The IUT does not relay the message sent to the Lower Tester’s Group Address.

All Friend Poll, Friend Update, Friend Subscription List Add, and Friend Subscription List Confirm messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

4.12.1.7 MESH/NODE/FRND/FN/BV-07-C [Friendship – Friend Node Friendship Not Established on Missed Friend Poll]

Test Purpose

Verify that an IUT node acting as a Friend Node can clear all data associated with an LPN when it does not receive the Friend Poll from it in time.

Reference

[3] Section 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.3.3
• Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester is a node that supports the Low Power Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

• Test Procedure
  1. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment procedure with a Transport Control Message Opcode field set to 0x03, the DST field set to the all-friends address, a TTL field set to 0x00, Criteria field value set to 0x21, ReceiveDelay value set to 0x0A, PollTimeout value of 0x000064, a LPNCounter set to 0, NumElements value of 0x01, and a PreviousAddress value of 0x0000. Lower Tester then waits 100 ms.
  2. The Lower Tester expects to receive a Friend Offer message from the IUT.
  3. The Lower Tester waits for 2 seconds after the offer window and then sends a Friend Poll Message.
  4. The Lower Tester expects to receive no response from the Friend Poll message as the friendship procedure was not established.

![Diagram of test procedure](image)

Figure 4.36: MESH/NODE/FRND/FN/BV-07-C

• Expected Outcome
  Pass verdict
  The IUT sends Friend Offer message to the Lower Tester’s unicast address, with a TTL value of 0, a Transport Control Message Opcode value of 0x04, and a ReceiveWindow field value between 0x01 – 0xFF.

  The IUT does not respond to the Friend Poll sent in step 3.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.
Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

4.12.1.8 MESH/NODE/FRND/FN/BV-08-C [Friendship – Friend Node Circular Queue]

- Test Purpose
  Verify that an IUT node acting as a Friend Node implements its cache as a circular queue and messages are only cleared from the queue once delivery is confirmed and older messages are discarded when the queue is full.

- Reference
  [3] Section 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.2, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure
  1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester 2 uses the cache size information of the IUT retained in step 1 and sends messages-N to the Lower Tester 1 where N is equal or smaller than the size of the cache of the IUT, and expects the IUT to cache these messages. All messages are sent within the PollTimeout time.
  3. The Upper Tester expects to receive positive acknowledgement of each message sent by Lower Tester 2 in step 2. If there are messages that are not acknowledged Lower Tester 2 will re-send until the number of cached message is known.
  4. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
  5. The Lower Tester 1 expects a message from the IUT and confirms that it retransmits the message sent in step 2 to the Lower Tester 1 with the expected contents and in the order of reception.
  6. The Lower Tester 1 will continue sending the Friend Poll message until it receives a Friend Update message with the MD field set to 0x00.
  7. The Lower Tester 2 uses the cache size information of the IUT retained in step 1 and sends messages-n to the Lower Tester 1 until n is greater than the size of the cache of the IUT and expects the IUT to cache these messages. All messages are sent within the PollTimeout time.
  8. The Lower Tester 1 sends a Friend Poll Message to the IUT within the PollTimeout time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
9. The Lower Tester 1 expects a message from the IUT and confirms that it does not retransmit Message-1 sent in step 6 as this message should have been discarded due to cache size. The Lower Tester 1 confirms that messages are retransmitted by IUT with expected contents and in the order of reception.

![Mesh Profile (MESH) Test Suite Diagram](image)

**Figure 4.37: MESH/NODE/FRND/FN/BV-08-C**

- **Expected Outcome**
  - **Pass verdict**

  The IUT successfully transmits all cached messages to the Lower Tester 1.

  The IUT sends a Friend Update message with the MD field set to 0x00 when there are no more cached messages.
All Cached Messages are received at the Lower Tester in the proper order.

All oldest messages sent that do not fit in the IUT’s cache are discarded in step 7 and not sent.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

**Fail verdict**

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

### 4.12.1.9 MESH/NODE/FRND/FN/BV-09-C [Friendship – Friend Node Cached Cleared on Poll Timeout]

- **Test Purpose**
  
  Verify that an IUT node acting as a Friend Node discards messages in its cache when friendship is lost because of a PollTimeout.

- **Reference**

  [3] Section 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

- **Initial Condition**

  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  
  - The Lower Tester 2 is a node supporting Configuration Client Model.
  
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  
  - The IUT and the Lower Tester 1 have run through the Friendship Establishment Procedure in test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].
  
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- **Test Procedure**

  1. The Lower Tester 2 sends a message to the IUT, emulating a different device attempting to send a message to the Lower Tester’s address emulating the LPN and expects the IUT to cache this message.
  
  2. The Lower Tester 1 does not send the Friend Poll message to the IUT for twice the PollTimeout time.
  
  3. The IUT should remove the Lower Tester 1 as a Friend.
  
  4. The Lower Tester sends Friend Poll messages and expects no response from IUT indicating that the Friendship between Lower Tester 1 and IUT has been removed.
  
  5. The Lower Tester 1 runs through the Friendship Establishment Procedure again in test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].
  
  6. The Lower Tester 1 sends a Friend Poll message to the IUT within the ReceiveDelay time. The Friend Poll Message must have a TTL of zero and be addressed to the IUT.
7. The Lower Tester 1 expects a Friend Update message from the IUT with the MD field set to 0x00.

Figure 4.38: MESH/NODE/FRND/FN/BV-09-C

- **Expected Outcome**
  
  **Pass verdict**
  
  In step 4, the IUT does not respond to Friend Poll messages indicating that Friendship between Lower Tester and IUT was removed.
  
  In step 7, the IUT sends a Friend Update message to the Lower Tester indicating there are no cached messages for the Lower Tester and new Friendship was established with empty cache.
  
  All Friend Poll and Friend Update messages are secured using the friendship security credentials.
  
  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.
  
  **Fail verdict**
  
  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.
4.12.1.10  MESH/NODE/FRND/FN/BV-10-C [Friendship – Friend Node Config Friend Set Received]

• Test Purpose
  Verify that an IUT node acting as a Friend Node clears all associated Friendship Relationships and all data associated with them when it receives the Config Friend Set message with “State” field set appropriately.

• Reference
  [3] Section 3.6.6, 4.3.2.56, 4.4.1.2.5

• Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - The Lower Tester 2 is a Configuration Client.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure
  1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester 2 sends a message to the IUT, emulating a different device attempting to send a message to the Lower Tester’s address emulating the LPN and expects the IUT to cache this message.
  3. The Lower Tester 2 sends the Config Friend Set message with the “State” parameter value equal to 0 to the IUT, the message is encrypted with the Device Key of the IUT.
  4. The Lower Tester 1 sends three Friend Poll messages to the IUT within the PollTimeout time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  5. The Lower Tester 1 expects no response from IUT indicating that the Friendship between Lower Tester 1 and IUT has been removed.
Expected Outcome

Pass verdict

The IUT does not respond to Friend Poll messages sent in step 3 indicating that Friendship between Lower Tester 1 and IUT was removed.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.


Test Purpose

Verify that an IUT node acting as a Friend Node sends the Friend Clear message to the previous Friend Node after the Friend Establishment Procedure to inform it of the new establishment and will retry sending if Friend Clear Confirmation is not received.

Reference

[3] Section 3.6.5.5, 3.6.5.6, 3.6.6, 3.6.6.3.3

Initial Condition

- The IUT is a node that supports the Friend Feature and the feature is enabled.
- The Lower Tester 1 is a node that supports the Low Power Feature.
- The Lower Tester 2 is a Friend Node and supports the Friend Feature.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- Test case MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node] has been executed.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure
  1. Lower Tester 1 completes the test case MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node] except the PreviousAddress value in step 1 is set to the Lower Tester 2’s address.
  2. The Lower Tester 2 expects the Friend Clear message from the IUT.
  3. The Lower Tester 2 waits for greater than the 1 second.
  4. The Lower Tester 2 expects the IUT to send the Friend Clear message again as in step 2.
  5. The Lower Tester 2 then sends the Friend Clear Confirm message to the IUT with the IUT’s address as the LPNAAddress field.
  6. The Upper Tester expects to receive positive acknowledgement from the IUT that it has received the Friend Clear Confirm message. If this is not received, then Lower Tester 2 retries sending the Friend Clear Confirm message.

Figure 4.40: MESH/NODE/FRND/FN/BV-11-C

• Expected Outcome

Pass verdict

The IUT sends the Friend Clear message to the Lower Tester 2 with the LPNAAddress value of the Lower Tester 1 address and a TTL value set to the maximum valid value in step 2.

The IUT sends a Friend Clear message again in step 4.

After step 6 is complete the IUT does not send any more Friend Clear Messages.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.
All Friend Clear and Friend Clear Confirm messages are secured using master security credentials.
All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

**Fail verdict**

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

### 4.12.1.12 MESH/NODE/FRND/FN/BV-12-C [Friend Clear and Friend Request Friendship Termination Scenarios]

- **Test Purpose**
  Verify that an IUT node acting as a Friend node terminates the friendship on receiving a Friend Clear message, or a Friend Request message from the Low Power node.

- **Reference**
  [3] Section 3.6.5.1, 3.6.5.3, 3.6.5.5, 3.6.5.6, 3.6.6, 3.6.6.3.1, 3.6.5.5, 3.6.6.4.2

- **Initial Condition**
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - The Lower Tester 2 is a Friend Node and supports the Friend Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- **Test Procedure**
  1. Lower Tester 1 completes the test case MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node] with the IUT.
  2. Lower Tester 2 sends a Friend Clear message to the IUT with the LPNAddress field set to the unicast address of the Lower Tester 1.
  3. Lower Tester 2 expects a Friend Clear Confirm message in response from the IUT.
  4. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. Lower Tester 1 expects no response from the IUT, and repeats the Friend Poll message two more times.
  5. Lower Tester 1 repeats step 1.
  6. Lower Tester 1 sends a Friend Request message using the same unicast address from the friendship establishment. The Lower Tester ignores any Friend Offer message from the IUT, if any.
  7. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. Lower Tester 1 expects no response from the IUT, and repeats the Friend Poll message two more times.
  8. Lower Tester 1 repeats step 1.
  9. Lower Tester 1 sends a Friend Clear message to the IUT, with the LPNAddress field set to its own unicast address.
10. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. Lower Tester 1 expects no response from the IUT, and repeats the Friend Poll message two more times.

Figure 4.41: MESH/NODE/FRND/FN/BV-12-C
• Expected Outcome
  
  **Pass verdict**
  
  In step 3, the IUT responds with a Friend Clear Confirm message with valid parameters.

  In steps 4, 7 and 10, the IUT does not respond to the Friend Poll message, confirming it has terminated the friendship.

  All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Friend Clear and Friend Clear Confirm messages are secured using the master security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  **Fail verdict**

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.


• Test Purpose
  
  Verify that an IUT node acting as a Friend Node calculates the correct Friend Offer Delay.

• Reference
  
  [3] Section 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.3.3

• Initial Condition
  
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  
  - The Lower Tester is a node that supports the Low Power Feature. The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure
  
  1. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode field set to 0x03, the DST field set to the all-friends address, a TTL field set to 0x00, Criteria field set to a value between 0x39 and 0x3F, ReceiveDelay field set to 0x0A, PollTimeout field set to 0x000064, a LPNCounter set to 0, NumElements field set to 0x01, and a PreviousAddress field set to 0x0000. Lower Tester then waits 100 ms.
  
  2. The Lower Tester starts a timer after sending the Friend Request.
  
  3. The Lower Tester then expects to receive a Friend Offer message. The Friend Offer message is expected to be received after the Friend Offer Delay which is calculated by using the RSSIFactor and ReceiveWindowFactor values.
4. When the Lower Tester receives the Friend Offer message from the IUT it checks that the value of the timer started in step 2 is equal to or greater than the calculated Friend Offer Delay.

<table>
<thead>
<tr>
<th>Lower Tester</th>
<th>IUT (Friend Node)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Friend Request</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>timer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Friend Offer</td>
</tr>
</tbody>
</table>

*Figure 4.42: MESH/NODE/FRND/FN/BV-13-C*

- **Expected Outcome**
  - **Pass verdict**
    
    The IUT sends the Friend Offer messages after the appropriate Friend Offer Delay. For the Lower Tester to calculate the Friend Offer Delay it will use the following formula.

    Local Delay is computed using the formula found in [3] Section 3.6.6.3.1. If the IUT sent the Friend Offer after this appropriate Friend Offer Delay, then test passes.

    Each Friend Request and Friend Offer messages are secured using the master security credentials.

    All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  - **Fail verdict**
    
    The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.


- **Test Purpose**
  
  Verify that an IUT node acting as a Friend Node can properly handle a Key Refresh procedure using Secure Network beacons.

- **Reference**

  [3] Section 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3, 3.6.6.4.1, 3.10.4

- **Initial Condition**
  
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending Secure Network Beacons.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Lower Tester 2 and IUT share device key of the IUT.
- Key Refresh procedure is not already in progress.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

**Test Procedure**

1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
3. The Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester and using the current IV Index, with the IV Update and Key Refresh flags set to 0 and the MD field set to 0x00.
4. The Lower Tester 2 sends Config NetKey Update message to IUT with new value of the NetKey thus starting Key Refresh procedure on IUT.
5. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
6. The Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester.
7. The Lower Tester 2 begins to broadcast a Secure Network Beacon, emulating a different device on the same network, secured with the new network key and KR flag set to 1.
8. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
9. The Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester, indicating the current IV Index, the MD field being set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 1. The Friend Update messages is secured using updated Friendship Security Credentials with the new NetKey.
10. The Lower Tester 2 begins to broadcast a Secure Network Beacon, emulating a different device on the same network, with the secured with new NetKey and KR flag set to 0.
11. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll Message will have a TTL of zero, be addressed to the IUT and secured with new NetKey.
12. The Lower Tester 1 expects a Friend Update message from the IUT indicating the IV Index, the MD field being set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 0. The Friend Update messages is secured using new NetKey.
Figure 4.43: MESH/NODE/FRND/FN/BV-14-C
Mesh Profile (MESH) / Test Suite

- Expected Outcome
  
  **Pass verdict**

  The IUT communicates the updated key to the Lower Tester with the MD field set to 0x00, the IV Update Flag set to 0, and the Key Refresh flag set to 1 in the Friend Update Message in step 9 after receiving the Secure Network Beacon with KR Bit Set to 1 in step 7. The IUT sends a Friend update message in step 12 with the IV Index, MD field being set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 0 secured with the new network key.

  All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  **Fail verdict**

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.


- Test Purpose

  Verify that an IUT node acting as a Friend Node can properly handle a Key Refresh procedure using Configuration Client messages.

- Reference

  [3] Section 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3, 3.6.6.4.1, 3.10.4

- Initial Condition

  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - The Lower Tester 2 is a Configuration Client.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Lower Tester 2 and IUT share device key of the IUT.
  - Key Refresh procedure is not already in progress.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
3. The Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester and indicating the current IV Index, the IV Update and Key Refresh flags set to 0 and the MD field set to 0x00.

4. The Lower Tester 2 sends Config NetKey Update message to IUT with new value of the NetKey thus starting Key Refresh procedure on IUT.

5. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.

6. The Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester.

7. The Lower Tester 2 sends a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition field set to 0x02 and expects a Config Key Refresh Phase Status message from the IUT in response with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the NetKey, and the Phase set to 0x02.

8. The Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.

9. The Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester, indicating the current IV Index, the MD field set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 1. The Friend Update messages is secured using updated Friendship Security Credentials with the new NetKey.

10. The Lower Tester 2 sends a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition field set to 0x03 and expects a Config Key Refresh Phase Status message from the IUT in response with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the NetKey, and the Phase set to 0x00.

11. The Lower Tester 1 sends a Friend Poll Message to the IUT within the negotiated PollTimeout. The Friend Poll Message will have a TTL of zero, be addressed to the IUT and secured with new NetKey.

12. The Lower Tester 1 expects a Friend Update message from the IUT indicating the IV Index, the MD field set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 0. The Friend Update messages is secured using new NetKey.

• Expected Outcome

Pass verdict

The IUT communicates the updated key to the Lower Tester with the MD field set to 0x00 and, the IV Update Flag set to 0, and the Key Refresh flag set to 1 in the Friend Update Message in step 9 after receiving the Secure Network Beacon with KR Bit Set to 1 in step 7.

The IUT sends a Friend update message in step 12 with the IV Index, the MD field set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 0 secured with the new network key.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.
4.12.1.16  MESH/NODE/FRND/FN/BV-16-C [Friendship – Friend Node - FSN Field Not Changed]

- **Test Purpose**
  
  Verify that an IUT node acting as a Friend Node handles a Friend Poll message properly if the FSN field is not toggled.

- **Reference**
  
  [3] Section 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

- **Initial Condition**

  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- **Test Procedure**

  1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, RFU bits set to 0, and FSN bit toggled. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
  3. The Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester.
  4. The Lower Tester 2 sends an Unsegmented Access Message to Lower Tester 1 and expects the IUT to cache this message.
  5. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01, RFU set to 0, and FSN bit toggled. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  6. The Lower Tester 1 expects to receive the cached message from Lower Tester 2 sent in step 4. The Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the message sent in step 4 by the Lower Tester 2 with the expected contents.
  7. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01, RFU set to 0, and FSN bit not toggled. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  8. The Lower Tester 1 expects to receive the same message from the IUT as it did in step 6.
  9. The Lower Tester 1 repeats steps 7 and 8 two times to ensure the same message is always delivered from the IUT.
  10. The Lower Tester 2 then sends an Unsegmented Access Message to the Lower Tester 1 and expects the IUT to cache this message.
  11. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x00, RFU set to 0, and FSN bit toggled. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
12. The Lower Tester 1 expects to receive the cached message from Lower Tester 2 sent in Step 4. The Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the message sent in Step 4 by the Lower Tester 2 with the expected contents.

13. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01, RFU set to 0, and FSN bit toggled. The Friend Poll message will have a TTL of zero and be addressed to the IUT.

14. The Lower Tester 1 expects to receive the cached message from Lower Tester 2 sent in Step 10. The Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the message sent in Step 10 by the Lower Tester 2 with the expected contents.

15. The Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01, RFU set to 0, and FSN bit toggled. The Friend Poll message will have a TTL of zero and be addressed to the IUT.

16. The Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester.

Figure 4.44: MESH/NODE/FRND/FN/BV-16-C
• Expected Outcome

**Pass verdict**

The IUT sends the same message in step 8 as it did in step 6. The FSN bit must be 1 every time this same message is sent.

The IUT sends the same message two more times in step 9.

The IUT sends the cached message from Step 4 in Step 12.

The IUT sends the cached message from Step 10 in Step 14. The IUT sends the Friend Update Message to the Lower Tester 1 indicating it has zero messages for it in step 16.

The IUT sends each response within the [ReceiveDelay, ReceiveDelay + ReceiveWindow] time interval from the Friend Poll Message. The IUT sends a Friend Update message with the MD field set to 0x00 when there are no more cached messages.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

**Fail verdict**

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.


• Test Purpose

Verify that an IUT node acting as a Friend Node handles a SAR transaction properly when the segment acknowledgment is duplicated in the Friend Node cache.

• Reference

[3] Section 3.5.3, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

• Initial Condition

- The IUT is a node that supports the Friend Feature and the feature is enabled.
- The Lower Tester 1 is a node that supports the Low Power Feature.
- Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- **Test Procedure**

1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. The Lower Tester 2 sends a Segment Acknowledgement message with the Lower Tester 1’s address and BlockAck field set to 0x00000001. The IUT is expected to cache this message.
3. Lower Tester 1 does not send a Friend Poll yet and the Lower Tester 2 sends another Segment Acknowledgement message to the Lower Tester 1’s address with the same SeqZero value and BlockAck field set to 0x00000003. The IUT is expected to cache this message and discard the older Segment Acknowledgement message.
4. Lower Tester 1 then sends the Friend Poll.
5. The IUT sends the cached message to the Lower Tester 1.
6. The Lower Tester 2 sends a message as in test MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast] except the unicast destination address is that of Lower Tester 1 and in Step 1 the Lower Tester 2 does not complete the entire SAR transaction by not sending all segments of the message.
7. The Lower Tester 1 then sends a Friend Poll message to the IUT within the PollTimeout time, the Transport Control Message Opcode field is set to 0x01, the RFU bits set to 0, TTL field set to 0x00, and FSN bit toggled for each new Friend Poll message if the previous Friend Poll message has triggered a response.
8. The Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester.

![Diagram](image-url)
• **Expected Outcome**

**Pass verdict**

The IUT sends the second Segment Acknowledgement Message to the Lower Tester 1 during step 6 and does not receive two Segment Acknowledgement messages from the IUT.

The IUT sends the Friend Update message in Step 8 with the MD field set to 0x00.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

**Fail verdict**

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

**4.12.1.18 MESH/NODE/FRND/FN/BV-18-C [Message Caching based on TTL – Friend Node]**

• **Test Purpose**

Verify that an IUT acting as a Friend Node only caches messages with a TTL greater than or equal to 2.

• **Reference**

[3] Section 3.5.2.1, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

• **Initial Condition**

- The IUT is a node that supports the Friend Feature and the feature is enabled.
- The Lower Tester 1 is a node that supports the Low Power Feature.
- Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1

• **Test Procedure**

1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. The Lower Tester 2 sends a message to the IUT with the DST field set to the unicast address of Lower Tester 1 and with the TTL field set to a random valid value, greater than or equal to 2, expecting the IUT to cache the message.
3. The Lower Tester 1 sends a Friend Poll message to the IUT.
4. The Lower Tester 1 expects the IUT to respond with the message cached in step 2.
5. The Lower Tester 2 sends another message to the IUT with the DST field set to the unicast address of Lower Tester 1 and with the TTL field set to 1.
6. The Lower Tester 1 sends a Friend Poll message to the IUT.
7. The Lower Tester 1 expects the IUT to respond with a Friend Update message indicating it has not cached the message sent in step 5.

8. Repeat steps 5–7 using a TTL field value of 0 in step 5.

- Expected Outcome

  **Pass verdict**

  In step 4, the IUT sends the message received in step 2, with the MD bit set to 0 and the TTL field value decremented by 1. All other fields are unchanged.

  In each iteration through step 6, the IUT sends a Friend Update message, with the MD field set to 0x00.

  The Friend Poll and Friend Update messages and the retransmitted message are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.
Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.


- **Test Purpose**
  - Verify that an IUT acting as a Friend Node only places message segments in the Friend Cache when the complete segmented message has been received.

- **Reference**
  - [3] Section 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1, 3.6.6.5, 3.6.6.5.1

- **Initial Condition**
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- **Test Procedure**
  1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester 2 executes only steps 1-2 of the procedure described in MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast] setting the destination address to the unicast address of Lower Tester 1.
  3. The Lower Tester 1 sends a Friend Poll message to the IUT.
  4. The Lower Tester 1 expects the IUT to respond with a Friend Update message, indicating that it has not cached any of the segments received in step 2.
  5. The Lower Tester 2 executes step 3 of the procedure started in step 2.
  6. The Lower Tester 1 sends a Friend Poll message to the IUT.
  7. The Lower Tester 1 expects the IUT to respond with a cached message, containing a segment of the segmented message send by Lower Tester 2.
  8. The Lower Tester 1 repeats steps 6–7 until the IUT responds with a Friend Update message andreassembles the complete message.
Expected Outcome

Pass verdict

In step 4, the IUT sends a Friend Update message.

In each but the last iteration through step 7, the IUT sends a message containing a segment of the same segmented message. In the last iteration through step 7, the IUT sends a Friend Update message with the MD field set to 0x00.

The complete message reassembled by Lower Tester 1 is the same as the message sent by Lower Tester 2.
The Friend Poll and Friend Update messages and the retransmitted messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

**Fail verdict**

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

### 4.12.120 MESH/NODE/FRND/FN/BV-20-C [Adding another Friend Update – Friend Node]

- **Test Purpose**
  
  Verify that an IUT acting as a Friend node adds two Friend Update messages in the Friend Queue.

- **Reference**
  
  [3] Section 3.5.2.1, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

- **Initial Condition**
  
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester 1 supports the Configuration Client Model and has access to the device key of the IUT.
  - The Lower Tester 2 is a node that supports the Low Power feature.
  - The IUT and the Lower Testers share previously established network security credentials. The IUT has been in Normal Operation state of the IV Update procedure for at least 96 hours. There is no Key Refresh procedure in progress on the IUT.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- **Test Procedure**
  
  1. The Lower Tester 2 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester 1 updates the network key on the IUT by sending a Config NetKey Update message, containing a new value for the network key. The IUT responds with a Config NetKey Status message, confirming the reception of the new key value.
  3. The Lower Tester 1 sends a Secure Network beacon to the IUT, containing the new IV Index value, setting the IV Update flag to 1 and setting the Key Refresh flag to 0.
  4. The Lower Tester 1 sends another Secure Network beacon to the IUT, containing the new IV Index value, setting the IV Update flag to 1 and setting the Key Refresh flag to 1.
  5. The Lower Tester 2 sends a Friend Poll message to the IUT.
  6. The Lower Tester 2 expects the IUT to respond with a Friend Update message containing the new IV Index value, the IV Update flag set to 1, the Key Refresh flag set to 0 and the MD field set to 0x01.
  7. The Lower Tester 2 sends another Friend Poll message to the IUT, with a toggled FSN field value.
8. The Lower Tester 2 expects the IUT to respond with a Friend Update message containing the new IV Index value, the IV Update flag set to 1, the Key Refresh flag set to 1 and the MD field set to 0x00.

![Diagram](image)

**Figure 4.48: MESH/NODE/FRND/FN/BV-20-C**

- **Expected Outcome**
  - **Pass verdict**
    
    In step 6, the IUT sends a Friend Update message with the new IV Index value, the IV Update flag set to 1, the Key Refresh flag set to 0 and the MD field set to 0x01.
    
    In step 8, the IUT sends a Friend Update message with the new IV Index value, the IV Update flag set to 1, the Key Refresh flag set to 1 and the MD field set to 0x00.
    
    The Friend Poll and Friend Update messages and the retransmitted message are secured using the friendship security credentials.
    
    All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.
    
  - **Fail verdict**
    
    The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

• Test Purpose

Verify that an IUT node acting as a Friend Node responds to the Config Low Power Node PollTimeout Get message with the correct PollTimeout for each LPN it has a Friend relationship with.

• Reference

[3] Section 4.3.2.67, 4.3.2.68, 4.4.1.2.17, 4.4.1.2.18, 4.4.2.2.17

• Initial Condition

- The IUT is a node that supports the Friend Feature and the feature is enabled.
- The Lower Tester 1 is a node that supports the Low Power Feature.
- The Lower Tester 2 is a Configuration Client. It has access to the device key of the IUT defined in the IXIT file.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure

1. The Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. The Lower Tester 2 sends the Config Low Power Node PollTimeout Get message with the LPNAddress field value equal to the address of Lower Tester 1.
3. The Lower Tester 2 expects to receive the Config Low Power Node PollTimeout Status message from the IUT with the LPNAddress field equal to that of the Lower Tester 1 address and the PollTimeout field equal to the PollTimeout of the Lower Tester 1 that was stored during Step 1.
4. The Lower Tester 2 sends the Config Low Power Node PollTimeout Get message with the LPNAddress parameter value equal to any address that isn’t the address of Lower Tester 1.
5. The Lower Tester 2 expects to receive the Config Low Power Node PollTimeout Status message from the IUT with the LPNAddress field equal to that of the address sent by the Lower Tester 2 in Step 4 and the PollTimeout field equal to the 0x000000.
Figure 4.49: MESH/NODE/FRND/FN/BV-21-C

- **Expected Outcome**

  **Pass verdict**

  In step 3, the IUT sends a Config Low Power Node PollTimeout Status message to the Lower Tester 2 with the LPNAddress field equal to that of the Lower Tester 1 address and the PollTimeout field equal to the PollTimeout of the Lower Tester 1 that was stored during Step 1.

  In step 5, the IUT sends a Config Low Power Node PollTimeout Status message to the Lower Tester 2 with the LPNAddress field equal to the address sent in Step 4 from Lower Tester 2 and the PollTimeout field equal to 0x000000All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  The Config Low Power Node PollTimeout Get and Config Low Power Node PollTimeout Status messages are encrypted using the device key of the IUT.

  **Fail verdict**

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

• Test Purpose
  Verify that an IUT node acting as a Friend Node ignores a Control message with an invalid Transport Control Opcode.

• Reference
  [3] Section 3.6.5, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

• Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester is a node that supports the Low Power Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure
  1. The Lower Tester elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BI-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester sends a misformed Friend Poll message to the IUT within the PollTimeout time, with the Transport Control Opcode set to 0x0C, RFU set to 0, and FSN bit shall be toggled for every Friend Update Message received. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  3. The Lower Tester expects no Friend Update message from the IUT and no response to this invalid message.

![Diagram](image.png)

*Figure 4.50: MESH/NODE/FRND/FN/BI-01-C*
• Expected Outcome

Pass verdict

The IUT ignores the misformed Friend Poll message with an invalid Transport Control OpCode and does not send any Friend Update or any other messages in return.

All Friend Poll and Friend Update messages and the invalid message with Opcode equal to 0x0C are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.


• Test Purpose

Verify that an IUT node acting as a Friend node handles a Friend Poll message with incorrectly set Prohibited field.

• Reference

[3] Section 3.1.2, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

• Initial Condition

- The IUT is a node that supports the Friend feature and the feature is enabled.
- The Lower Tester is a node that supports the Low Power feature.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure

1. The Lower Tester elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.

2. The Lower Tester sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01 and the Prohibited field set to 0x7F. The Friend Poll message will have a TTL of zero and be addressed to the IUT.

3. The Lower Tester expects no response from the IUT.
Figure 4.51: MESH/NODE/FRND/FN/BI-02-C

- **Expected Outcome**
  
  **Pass verdict**

  The IUT sends no response to the Friend Poll message with an invalid Prohibited field value.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.


- **Test Purpose**

  Verify that an IUT node with Friendship feature handles a Friend Request message with incorrect set RFU bits.

- **Reference**

  [3] Section 3.6.5.1, 3.6.5.2, 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.3.1

- **Initial Condition**

  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester is a node that supports the Low Power Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- **Test Procedure**

  1. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set
to the all-friends address, a TTL of zero, MinCacheSizeLog value of 0b000 in the Criteria field, RSSIFactor value of 0b00, ReceiveWindowFactor value of 0b00, ReceiveDelay value of 0x0A, PollTimeout value of 0x000064, LPNCounter set to 0, NumElements value of 0x01, and a PreviousAddress value of 0x0000. Lower Tester then waits 100 ms.

2. The Lower Tester expects no response from the IUT.

3. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set to the all-friends address, a TTL of zero, MinCacheSizeLog value of 0b001 in the Criteria field, RSSIFactor value of 0b00, ReceiveWindowFactor value of 0b00, ReceiveDelay set to any random value between 0x00 – 0x09, PollTimeout value of 0x000064, LPNCounter set to 0, NumElements value of 0x01, and a PreviousAddress value of 0x0000. Lower Tester then waits 100 ms.

4. The Lower Tester expects no response from the IUT.

5. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set to the all-friends address, a TTL of zero, Criteria field RFU value of 0b1, RSSIFactor value of 0b00, ReceiveWindowFactor value of 0b00, ReceiveDelay value of 0x0A, PollTimeout value of 0x000064, a LPNCounter set to 0, NumElements value of 0x00, and a PreviousAddress value of 0x0000. Lower Tester then expects to receive a Friend Offer message. The Friend Offer Message should be sent after the Friend Offer Delay which is calculated by using the RSSI factor and Received Window Factor.
Figure 4.52: MESH/NODE/FRND/FN/BL-03-C
• Expected Outcome

Pass verdict

The IUT sends Friend Offer message to the Lower Tester’s unicast address ignoring RFU bit in Criteria field of the Friend Request message, with a TTL value of 0, and a Transport Control Message Opcode value of 0x04.

The IUT sends the Friend Offer messages after the calculated Friend Offer Delay.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

4.12.2 Low Power Node

This section contains tests related to behavior from the perspective of a node participating in Friendship procedures as a Low Power node.

4.12.2.1 MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]

• Test Purpose

Verify that an IUT node acting as a Low Power Node can perform the Friendship Establishment procedure.

• Reference

[3] Section 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 4.2.2.4

• Initial Condition

- The IUT is a node that supports the Low Power Feature.
- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester is a node that supports the Friend Feature.
- The Lower Tester supports the Configuration Client Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The IUT exposes some mechanism to initiate the Low Power Establishment operation.
- The IUT has not ran the Low Power Establishment operation with other nodes yet.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure

1. The Upper Tester orders the IUT to initiate the Low Power Establishment operation.
2. The Lower Tester expects to receive a Friend Request message from the IUT requesting to begin the Friendship Establishment operation.
3. The Lower Tester should wait the appropriate Friend Offer Delay computed using the formula found in [3] Section 3.6.6.3.1 from receiving the Friend Request message and then send Friend Offer message to the IUT’s unicast address, with a TTL value of 0, a Transport Control Message
Opcode value of 0x04, a ReceiveWindow field value equal to 0xFF, RSSI value of the measured RSSI of the IUT, and a FriendCounter field with value equal to the number of times the Lower Tester sent a Friend Offer message.

4. The Lower Tester starts a timer once the Friend Offer has been sent.

5. The Lower Tester expects to receive a Friend Poll from the IUT within 1 second of sending the Friend Offer.

6. Once the Lower Tester receives the Friend Poll from the IUT it responds with a Friend Update message with the Key Refresh flag field set to 0, the IV Update flag field set to 0 and the MD field set to 0x00.

![Diagram of MESH/NODE/FRND/LPN/BV-01-C](image)

**Figure 4.53: MESH/NODE/FRND/LPN/BV-01-C**

- **Expected Outcome**

  **Pass verdict**

  The IUT sends a Friend Request to the Lower Tester with the DST field set to the all-friends address, a TTL of 0, Criteria field value between 0x00 and 0x7F excluding values when MinCacheSizeLog is equal to 0b000, ReceiveDelay value between 0x0A and 0xFF, PollTimeout value between 0x00000A and 0x34BBFF, a LPNCounter set to 0, a NumElements field equal to that of the total number of elements on the device, and the PreviousAddress field value set to Unassigned Address 0x0000. The Transport Control Opcode field is set to 0x03.

  The IUT sends a Friend Poll message to the Lower Tester no more than 1 seconds after the Friend Offer was sent by the Lower Tester. The Transport Control Opcode field is set to 0x01, RFU field set to 0x00, FSN field set to 0, TTL field set to 0 and destination address set to the Lower Tester unicast address.

  The Friend Poll message is secured using the friendship security credentials.

  Each Friend Request message is secured using the master security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.
### 4.12.2.2 MESH/NODE/FRND/LPN/BV-02-C [Friendship Messaging – Low Power Node]

#### Test Purpose

Verify that an IUT node acting as a Low Power Node can request and retrieve cached messages from a Friend Node and can handle a Friendship Credentials Flag change.

#### Reference

[3] Section 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 3.6.6.4.2

#### Initial Condition

- The IUT is a node that supports the Low Power Feature.
- The Lower Tester is a node that supports the Friend Feature and the Configuration Client Model.
- The IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Lower Tester has access to the device key of the IUT.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

#### Test Procedure

1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
3. The following steps 4–7 shall be repeated for N messages where N =1, 2, 3, and 4.
4. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll Message to the Lower Tester. The Friend Poll Message will have a TTL of zero and be addressed to the Lower Tester. The Lower Tester records the FSN field value of each Friend Poll Messages to ensure each Friend Poll has an appropriate FSN field.
5. The Lower Tester sends a Friend Update Message to the IUT indicating it has zero messages for the IUT and expects the IUT to return to sleep.
6. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll Message to the Lower Tester. The Friend Poll will have a TTL of zero and be addressed to the Lower Tester.
7. The Lower Tester sends N message(s) to the IUT followed by a Friend Update message with the MD field set to 0x00. The Lower Tester expects an appropriate Friend Poll message from the IUT after sending each cached message, and no Friend Poll message after sending the Friend Update message.
8. Upper Tester orders the IUT to publish a Health Status message.
9. The Lower Tester expects the IUT to use the master security credentials when sending the Health Status message.
10. The Lower Tester caches a Config Model Publication Set message with the CredentialFlag field set to 1.
11. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll Message to the Lower Tester.
12. The Lower Tester sends the cached Config Model Publication Set message to the IUT and after another Friend Poll message from the IUT it sends a Friend Update message with the MD field set to 0x00.
13. Upper Tester orders the IUT to publish a Health Status message.
14. The Lower Tester expects the IUT to use the friendship security credentials when sending the Health Status message.

**Expected Outcome**

*Pass verdict*

The IUT sends a Friend Poll Messages to the Lower Tester within the PollTimeout in each cycle of the test procedure.
Each Friend Poll Message has an appropriate FSN value, RFU field set to 0, TTL value set to 0, and the Transport Control message, Opcode set to 0x01 and destination address set to the Lower Tester unicast address.

The IUT successfully receives all cached messages and sends all expected Friend Poll messages. The IUT stops sending Friend Poll messages when it receives a Friend Update message with the MD field set to 0x00.

The IUT sends the Health Status message using the friendship security credentials in Step 14.

Each Friend Poll message is secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

4.12.2.3 MESH/NODE/FRND/LPN/BV-03-C [Friend IV Update – Low Power Node]

• Test Purpose
Verify that an IUT node acting as a Low Power Node can receive and react to IV Index updates from the Friend Node.

• Reference
[3] Section 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 3.8.3

• Initial Condition
- The IUT is a node that supports the Low Power Feature.
- The Lower Tester is a node that supports the Friend Feature.
- The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure
1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use. The Lower Tester will send all Friend Update messages in this test procedure with the MD field set to 0x00.
3. The Upper Tester orders the IUT to ignore IV Update procedure time constrains.
4. The Lower Tester expects the IUT to send a Friend Poll Message within the PollTimeout. The Friend Poll Message will utilize the old IV Index and have a TTL of zero and be addressed to the Lower Tester.
5. The Lower Tester sends a Friend Update Message with the new IV Index and the IV Index flag set to 1, indicating that the IV Update procedure is in IV Update in Progress state.
6. The Lower Tester expects the IUT to send a Friend Poll Message within the PollTimeout. The Friend Poll Message will utilize the old IV Index and have a TTL of zero and be addressed to the Lower Tester.
7. The Lower Tester sends a Friend Update message with the new IV Index and the IV Index flag set to 0, indicating that the IV Update procedure is completed.

8. The Lower Tester expects the IUT to send a Friend Poll Message within the PollTimeout. The Friend Poll message will utilize the new IV Index and have a TTL of zero and be addressed to the Lower Tester.

9. In response to the Friend Poll message, the Lower Tester sends a Friend Update with the new IV Index and IV Index flag set to 0.

**Figure 4.55: MESH/NODE/FRND/LPN/BV-03-C**

- **Expected Outcome**

  **Pass verdict**

  The IUT sends a Friend Poll messages to the Lower Tester within the PollTimeout in each cycle of the test procedure.
Each Friend Poll message has an appropriate FSN value from the IUT, RFU field set to 0, TTL value set to 0, and the Transport Control message, Opcode set to 0x01 and destination address set to the Lower Tester unicast address.

The IUT begins to utilize the new IV Index once it receives the Friend Update message with new IV Index and IV Update flag set to 0 from the Lower Tester.

Each Friend Poll message is secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

4.12.2.4 MESH/NODE/FRND/LPN/BV-04-C [Friendship – Send Friend Subscription List Add Message]

• Test Purpose
  Verify that an IUT node acting as a Low Power Node can send the Friend Subscription List Add message to a relay.

• Reference
  [3] Section 3.6.6, 3.6.5.1, 3.6.5.2, 3.6.5.7, 3.6.5.9, 3.6.6.3.3, 3.6.6.4.3

• Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Lower Tester and IUT have previously established a friend relationship through test MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node].
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure
  1. The Upper Tester orders the IUT to send a Friend Subscription List Add message to the Lower Tester with the “AddressList” field containing values supplied in the IXIT [6] and the “TransactionNumber” field containing a valid value.
  2. The Lower Tester expects to receive a Friend Subscription List Add message.
Figure 4.56: MESH/NODE/FRND/LPN/BV-04-C

• Expected Outcome

Pass verdict

The IUT sends a Friend Subscription List Add message with values supplied in the IXIT [6].

The TransactionNumber in Step 1 is the value 0x00.

The Friend Subscription List Add message is secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

4.12.2.5 MESH/NODE/FRND/LPN/BV-05-C [Friendship – Send Friend Subscription List Remove Message]

• Test Purpose

Verify that an IUT node acting as a Low Power Node can send the Friend Subscription List Remove message to a relay.

• Reference

[3] Section 3.6.5.1, 3.6.5.2, 3.6.5.7, 3.6.5.8, 3.6.5.9, 3.6.6, 3.6.6.3.3

• Initial Condition

- The IUT is a node that supports the Low Power Feature.
- The Lower Tester is a node that supports the Friend Feature.
- The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Lower Tester and IUT have previously established a friend relationship and have executed the test procedure described in MESH/NODE/FRND/LPN/BV-04-C [Friendship – Send Friend Subscription List Add Message].
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
• Test Procedure

1. The Upper Tester orders the IUT to send a Friend Subscription List Remove message to the Lower Tester with the "AddressList" field containing values supplied in the IXIT [6] and the "TransactionNumber" field containing a valid value.
2. The Lower Tester expects to receive a Friend Subscription List Remove message.

Figure 4.57: MESH/NODE/FRND/LPN/BV-05-C

• Expected Outcome

Pass verdict

The IUT sends a Friend Subscription List Remove message with values supplied in the IXIT [6]. The Friend Subscription List Remove message is secured using the friendship security credentials. All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

4.12.2.6 MESH/NODE/FRND/LPN/BV-06-C [Friendship Updates Failed Friend Update Message – Low Power Node]

• Test Purpose

Verify that an IUT node acting as a Low Power Node can handle a missed Friend update message.

• Reference

[3] Section 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 3.6.6.4.2

• Initial Condition

- The IUT is a node that supports the Low Power Feature.
- The Lower Tester is a node that supports the Friend Feature.
- The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- **Test Procedure**
  1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
  2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  3. The Lower Tester expects the IUT to send a Friend Poll message within the PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the Lower Tester.
  4. The Lower Tester receives the Friend Poll message, waits and does not send the Friend Update message.
  5. The Lower Tester waits for the Friend Poll message to be sent again and checks to make sure the FSN field is the same as before since there was no acknowledgment of the Friend Poll.

![Diagram of MESH/NODE/FRND/LPN/BV-06-C](image)

*Figure 4.58: MESH/NODE/FRND/LPN/BV-06-C*

- **Expected Outcome**
  - **Pass verdict**

  The IUT sends a Friend Poll messages to the Lower Tester within the PollTimeout in each cycle of the test procedure.

  The IUT sends another Friend Poll message with the same FSN field value as the first in step 4.

  Each Friend Poll messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

4.12.2.7 MESH/NODE/FRND/LPN/BV-07-C [Friend Key Refresh – Low Power Node]

- **Test Purpose**
  - **Verify that an IUT node acting as a Friend Node can properly handle a Key Refresh procedure.**
• **Reference**

[3] Section 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3, 3.6.6.4.1, 3.10.4

• **Initial Condition**

  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - Key Refresh procedure is not already in progress.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
  - IV Update test mode is activated on the IUT.

• **Test Procedure**

  1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
  2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use. The Lower Tester will send all Friend Update messages in this test procedure with the MD field set to 0x00.
  3. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll Message to the Lower Tester. The Friend Poll message will have a TTL of zero and be addressed to the Lower Tester. The Lower Tester records the FSN field value of each Friend Poll message to ensure each Friend Poll has an appropriate FSN field.
  4. The Lower Tester sends Config NetKey Update message to IUT with new value of the NetKey thus starting Key Refresh procedure on IUT.
  5. The Lower Tester expects a Friend Poll message from the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT and is secured with the old NetKey.
  6. The Lower Tester sends a Friend Update message to the IUT indicating it has zero messages for the Lower Tester, indicating the current IV Index, the IV Update Flag equal to 0, and Key Refresh flag set to 1. The Friend Update messages are secured using updated Friendship Security Credentials with the new NetKey.
  7. The Lower Tester expects the IUT to send a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT and secured with the new NetKey.
  8. The Lower Tester sends a Friend Update message to the IUT indicating it has zero messages for the IUT, indicating the current IV Index, the IV Update Flag equal to 0, and Key Refresh flag set to 0. The Friend Update messages is secured using updated Friendship Security Credentials with the new NetKey.
Expected Outcome

Pass verdict

The IUT sends a Friend Poll message in Step 7 secured with the new network key.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.
4.12.2.8 MESH/NODE/FRND/LPN/BV-08-C [Low Power Node Send Friend Clear Message]

- **Test Purpose**
  Verify that an IUT node acting as a Low Power Node can send the Friend Clear message.

- **Reference**
  [3] Section 3.6.5.5, 3.6.6.4.2

- **Initial Condition**
  - The IUT is a node that supports the Low Power feature.
  - The Lower Tester acts as a Friend node.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Lower Tester and IUT have previously established a friend relationship through test MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node].
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- **Test Procedure**
  1. The Upper Tester orders the IUT to terminate the friendship using a Friend Clear message.
  2. The Lower Tester expects to receive a Friend Clear message from the IUT.

   ![Test Diagram](image)

   *Figure 4.60: MESH/NODE/FRND/LPN/BV-08-C*

- **Expected Outcome**
  **Pass verdict**

  In step 2, the IUT sends a Friend Clear message to the Lower Tester. The LPNAddress field is set to the unicast address of the IUT.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.
4.12.2.9 MESH/NODE/FRND/LPN/BI-01-C [Friendship Establishment Mode – Low Power Node Friend Poll Retries]

• Test Purpose
  Verify that an IUT node acting as a Low Power Node will retry Friend Polls during the Friendship Establishment period until the maximum number of retries.

• Reference
  [3] Section 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 3.6.6.4.2

• Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The IUT exposes some mechanism to initiate the Low Power Establishment operation.
  - The IUT has not ran the Low Power Establishment operation with other nodes yet.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure
  1. The Upper Tester orders the IUT to initiate the Low Power Establishment operation.
  2. The Lower Tester expects to receive a Friend Request message from the IUT requesting to begin the Friendship Establishment operation.
  3. The Lower Tester should wait the calculated Friend Offer Delay from receiving the Friend Request message and then send a Friend Offer message to the IUT’s unicast address, with a TTL value of 0, a Transport Control message, Opcode value of 0x04, a ReceiveWindow field value equal to 0xFF, RSSI value of the measured RSSI of the IUT, and a FriendCounter field with value equal to the number of times the Lower Tester sends a Friend Offer message.
  4. The Lower Tester expects to receive the Friend Poll message no more than 1 second after sending the Friend Offer.
  5. The Lower Tester waits and does not send a Friend Update message.
  6. The Lower Tester expects the IUT to resend the Friend Poll message.
  7. The Lower Tester responds with a Friend Update message with the MD field set to an invalid value in the interval 0x02-0xFF.
  8. The Lower Tester expects the IUT to resend the Friend Poll message.
**Expected Outcome**

*Pass verdict*

The IUT sends a Friend Request message in step 2 with the DST field set to the all-friends address, a TTL of 0, Criteria field value between 0x00 and 0x7F excluding values when MinCacheSizeLog is equal to 0b000, ReceiveDelay value between 0x0A and 0xFF, PollTimeout value between 0x00000A and 0x34BBFF, a LPNCounter set to 0, a NumElements field equal to that of the total number of elements on the device, and the PreviousAddress field value set to Unassigned Address.

The IUT sends a Friend Poll message in steps 4, 6, and 8.

All Friend Poll messages are secured using the friendship security credentials.

All Friend Request messages are greater than 1.1 seconds apart from each other.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

**4.12.2.10 MESH/NODE/FRND/LPN/BI-02-C [Invalid Transport Control OpCode – Low Power Node]**

*Test Purpose*

Verify that an IUT node acting as a Low Power Node can handle a message properly that contains an RFU Transport Control Opcode.

*Reference*

[3] Section 3.6.5, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

*Initial Condition*

- The IUT is a node that supports the Low Power Feature.
- The Lower Tester is a node that supports the Friend Feature.
The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

**Test Procedure**

1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
3. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll message to the Lower Tester. The Friend Poll message will have a TTL of zero and be addressed to the Lower Tester. The Lower Tester records the FSN field value of each Friend Poll message.
4. The Lower Tester sends a message to the IUT with the Transport Control Opcode field set to an RFU value, setting the SRC field to its own unicast address (the Lower Tester’s unicast address used during friendship establishment).
5. The Lower Tester expects to receive the next Friend Poll message with the FSN field value toggled.

![Diagram](Image)

*Figure 4.62: MESH/NODE/FRND/LPN/Bl-02-C*

- **Expected Outcome**
  - *Pass verdict*

The IUT ignores the message with an RFU Transport Control Opcode value but sends the Friend Poll message in step 5 with an alternating FSN value.

Each Friend Poll message is secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.
4.12.2.11 MESH/NODE/FRND/LPN/BI-03-C [Friendship Low Power Node – Friend Offer Invalid RFU Bits]

• Test Purpose

Verify that an IUT node acting as a Low Power Node can handle a Friend Offer message with invalid set RFU bits.

• Reference

[3] Section 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1

• Initial Condition

- The IUT is a node that supports the Low Power Feature.
- The Lower Tester is a node that supports the Friend Feature.
- The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The IUT exposes some mechanism to initiate the Low Power Establishment operation.
- The IUT has not ran the Low Power Establishment operation with other nodes yet.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure

1. The Upper Tester orders the IUT to initiate the Low Power Establishment operation.
2. The Lower Tester expects to receive a Friend Request message from the IUT requesting to begin the Friendship Establishment operation.
3. The Lower Tester should wait the appropriate Friend Offer Delay computed using the formula found in [3] Section 3.6.6.3.1 from receiving the Friend Request message and then send a Friend Offer messages to the IUT’s unicast address, with a TTL value of 0, a Transport Control message, Opcode value of 0x04, a ReceiveWindow field value equal to 0x00, RSSI value of the measured RSSI of the IUT, and a FriendCounter field with value equal to the number of times the Lower Tester sends a Friend Offer message.
4. The Lower Tester starts a timer once the Friend Offer has been sent.
5. The Lower Tester does not expect the IUT to send a Friend Poll message and the Friendship should not be established.
Expected Outcome

Pass verdict

The IUT sends a Friend Request to the Lower Tester with the DST field set to the all-friends address, a TTL of zero, Criteria field value between 0x00 and 0x7F excluding values when MinCacheSizeLog is equal to 0b000, ReceiveDelay value between 0x0A and 0xFF, Poll7imeout value between 0x00000A and 0x34BBFF, a LPNCounter set to 0, a NumElements field equal to that of the total number of elements on the device, and the PreviousAddress field value set to Unassigned Address. The Transport Control Opcode field should be set to 0x03.

The IUT does not send a Friend Poll message in response to the Friend Offer.

The Friend Request message is secured using the master security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

### 4.13 Proxy Protocol

The test objective is to verify functionality of the proxy protocol which can be used over several connection oriented channels.

#### 4.13.1 MESH/SR/PROX/BV-01-C [Receive White List Filter Setup]

- **Test Purpose**
  
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can receive the Proxy Filter for setup for a white list filter.

- **Reference**
  
  [3] Section 6.1, 6.4, 6.4.1, 6.5, 6.6

- **Initial Condition**
  
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
- The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).

- Lower Tester is a Proxy Client.

**Test Procedure**

1. Lower Tester sends a Proxy PDU to the IUT with the SAR field set to b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value set to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x00 (White List Filter).

2. The Lower Tester expects the IUT to send a Proxy Configuration Message of “Filter Status” back to the Lower Tester with a FilterType field set to 0x00 (White List) and a ListSize field set to [0x00].

3. Lower Tester sends a Proxy PDU to the IUT with the SAR field set to b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value set to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray field set with 1 address equal to the of the Lower Tester.

4. The Lower Tester expects the IUT to send a Proxy Configuration Message of “Filter Status” back to the Lower Tester with a FilterType value equal to 0x00 (White List) and a ListSize equal to [1].

**Figure 4.64: MESH/SR/PROX/BV-01-C**

**Expected Outcome**

**Pass verdict**

The IUT sends a single Proxy Configuration Message with 0x03 “Filter Status” as the Proxy Configuration Status OpCode, a FilterType field value of 0x00 (White List), and a ListSize field value of [0] in step 2.

The IUT sends a single Proxy Configuration Message with 0x03 “Filter Status” as the Proxy Configuration Status OpCode, a FilterType field value of 0x00 (White List), and a ListSize field value of [1] in step 4.
Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address 0x0000.

4.13.2 MESH/SR/PROX/BV-02-C [Complete Message Network PDU Relay to Proxy Client]

- **Test Purpose**
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can send a Network PDU to the Proxy Client where one complete message fits in the PDU.

- **Reference**
  [3] Section 6.3, 6.5, 6.6

- **Initial Condition**
  - The IUT has just performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - The address of Lower Tester 1 has been added to the IUT's White list by running test case MESH/SR/PROX/BV-01-C [Receive White List Filter Setup].

- **Test Procedure**
  1. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address of the Lower Tester 1. The Proxy PDU does not require reassembly at the IUT.
  2. The Lower Tester 1 expects the IUT to send a Proxy PDU with SAR field set to 00 (“Complete higher level message”), MessageType equal to 0x00 (“Network PDU”), and the complete mesh message to the Lower Tester 1.

![Diagram](image-url)  
*Figure 4.65: MESH/SR/PROX/BV-02-C*
• Expected Outcome

Pass verdict

The IUT sends a single Proxy PDU to the Lower Tester 1 with the Access message data the same as sent from Lower Tester 2 to the IUT. The TTL value should be decremented by 1 for the Proxy hop.

4.13.3 MESH/SR/PROX/BV-03-C [Segmented Message Network PDU Relay to Proxy Client]

• Test Purpose

Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can send a Network PDU to the Proxy Client where one complete message does not fit into the PDU.

• Reference

[3] Section 6.3, 6.5, 6.6

• Initial Condition

- The IUT has performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.

- The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).

- Lower Tester 1 is a Proxy Client.

- Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.

- The address of Lower Tester 1 has been added to the IUT’s White list by running test case MESH/SR/PROX/BV-01-C [Receive White List Filter Setup].

• Test Procedure

1. Lower Tester 2 sends a mesh message to the IUT larger than the negotiated MTU for that connection and with the destination address of the Lower Tester 1.

2. The Lower Tester 1 expects the IUT to send a Proxy PDU to the Lower Tester 1 with SAR field set to b01 (“Data field contains the first segment of a message”), MessageType equal to 0x00 (“Network PDU”), and Data field containing the first segment of the mesh message to the Lower Tester 1.

3. If the message size in step 2 was greater than 2 times the negotiated MTU for the Proxy connection then IUT then sends a Proxy PDU to the Lower Tester 1 with SAR field set to b10 (“Data field contains a continuation segment of a message”), MessageType equal to 0x00 (“Network PDU”), and Data field containing the next complete segment of the mesh message to the Lower Tester 1.

4. Step 3 is repeated for as many times until the last segment is to be sent.

5. If the message size in step 2 was not greater than 2 times the negotiated MTU for that Proxy connection then steps 3 and 4 will not occur and the IUT then sends a Proxy PDU to the Lower Tester 1 with SAR field set to b11 (“Data field contains the last segment of a message”), MessageType equal to 0x00 (“Network PDU”), and the Data field containing the last segment of the mesh message to the Lower Tester 1.

6. Lower Tester 1 receives all Proxy PDUs from the IUT with the exact data contained in the entire message sent from the Lower Tester 2 to the IUT.
**Expected Outcome**

**Pass verdict**

The IUT sends all the correct Proxy PDUs in steps 2 – 5 with the SAR field set correctly for each associated step, Message Type equal to 0x00.

The Lower Tester 1 receives the entire mesh message data sent by the Lower Tester 2.

4.13.4 MESH/SR/PROX/BV-04-C [Send a Complete Message Network PDU to Node from Proxy Client]

**Test Purpose**

Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can send a Network PDU from the Proxy Client to the Mesh Node where one complete message does fit into the PDU.

**Reference**

[3] Section 6.3, 6.5, 6.6

**Initial Condition**

- The IUT has performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.
- The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
- Lower Tester 1 is a Proxy Client.
- Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
- The address of Lower Tester 1 has been added to the IUT’s White list by running test case MESH/SR/PROX/BV-01-C [Receive White List Filter Setup].
• Test Procedure

1. Lower Tester 1 sends a single Proxy PDU to the IUT smaller than the negotiated proxy MTU with SAR field set to 00 (“Complete higher level message”), MessageType equal to 0x00 (“Network PDU”), and with the DST address of Lower Tester 2. The Proxy PDU does not require reassembly at the IUT.

2. The Lower Tester 2 expects the IUT to send a Proxy PDU with SAR field set to 00 (“Complete higher level message”), Type equal to 0x00 (“Mesh Data”), and the complete mesh data to the Lower Tester 2.

3. Lower Tester 2 receives a mesh message from the IUT with the exact data contained in the Proxy PDU sent from Lower Tester 1 to the IUT.

• Expected Outcome

Pass verdict

The IUT sends a single mesh message to the Lower Tester 2 with the exact data contained in the Proxy PDU sent from Lower Tester 1 to the IUT.

4.13.5 MESH/SR/PROX/BV-05-C [Send a Segmented Message Network PDU to Node from Proxy Client]

• Test Purpose

Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can send a Network PDU from the Proxy Client to the Mesh Node where one complete message does not fit into the PDU.

• Reference

[3] Section 6.3, 6.5, 6.6

• Initial Condition

- The IUT has performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.

- The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).

- Lower Tester 1 is a Proxy Client.

- Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.

- The address of Lower Tester 1 has been added to the IUT’s White list by running test case MESH/SR/PROX/BV-01-C [Receive White List Filter Setup].

• Test Procedure

1. Lower Tester 1 sends a mesh message to the IUT larger than the negotiated MTU for that connection and with the destination address of the Lower Tester 2.

2. The Lower Tester 1 first sends a Proxy PDU to the IUT with SAR field set to b01 (“Data field contains the first segment of a message”), MessageType equal to 0x00 (“Network PDU”), and the Data field contains the first portion of segmented mesh data to the IUT.

3. If the message size in step 1 was greater than 2 times the negotiated MTU for the Proxy connection the Lower Tester 1 then sends a Proxy PDU to the IUT with SAR field set to b10
4. Step 2 is repeated for as many times until the last mesh data segment is to be sent.
5. If the message size in step 1 was not greater than 2 times the negotiated MTU for that Proxy connection then steps 2 and 3 will not occur and the Lower Tester 1 then sends a Proxy PDU to the IUT with SAR field set to b11 (“Data field contains the last segment of a message”), MessageType equal to 0x00 (“Network PDU”), and the Data field contains the last segment of mesh data.
6. The Lower Tester 2 expects the IUT to send a mesh message containing the reassembled message sent to the IUT in step 1, the IUT performs segmentation as defined in test MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast] if needed.
7. The Lower Tester 1 then sends another Proxy PDU to the IUT with SAR field set to b01 (“Data field contains the first segment of a message”), MessageType equal to 0x00 (“Network PDU”), and the Data field contains the first portion of segmented mesh data to the IUT.
8. Lower Tester 1 waits for more than 20 seconds.
9. The Lower Tester 1 expects the IUT to terminate the connection.

Figure 4.67: MESH/SR/PROX/BV-05-C

- Expected Outcome
  
  Pass verdict

  The IUT sends the entire mesh message that Lower Tester 1 sends through steps 1–4 to the Lower Tester 2 in step 6. In Step 9, the IUT terminates the connection.
4.13.6 MESH/SR/PROX/BV-06-C [Filter White List Addresses]

- **Test Purpose**
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly filter messages for the appropriate white list set.

- **Reference**
  [3] Section 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- **Initial Condition**
  - The IUT has performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network. Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - The address of Lower Tester 1 has been added to the IUT's White list by running test case MESH/SR/PROX/BV-01-C [Receive White List Filter Setup].

- **Test Procedure**
  1. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address equal to the Lower Tester 1. The Proxy PDU does not require reassembly at the IUT.
  2. The Lower Tester 1 expects the IUT to send a Proxy PDU with SAR field set to 00 (“Complete higher level message”), MessageType equal to 0x00 (“Network PDU”), and Data field containing the complete mesh data to the Lower Tester 1.
  3. Lower Tester 1 receives a mesh message from the IUT with the exact mesh data contained in the Proxy PDU sent from the Lower Tester 2 to the IUT.
  4. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address that is not the address of the IUT and also not the address of Lower Tester 1. The mesh message does not require reassembly at the IUT.
  5. The Lower Tester 1 expects the IUT should not send any Proxy PDUs.

- **Expected Outcome**
  **Pass verdict**
  The IUT sends a single Proxy PDU to the Lower Tester 1 with the SAR field set to 00 (“Complete higher level message”), MessageType equal to 0x00 (“Network PDU”), and Data field containing the complete mesh data sent from Lower Tester 2 to the IUT in step 2.

  The IUT does not send any Proxy PDUs to the Lower Tester 1 in step 5.

  Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address 0x0000.
4.13.7 MESH/SR/PROX/BV-07-C [Receive Black List Filter Setup]

• Test Purpose
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can receive the Proxy Filter for setup of a black list filter.

• Reference
  [3] Section 6.2.1, 6.4, 6.4.1, 6.5, 6.6

• Initial Condition
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester is a Proxy Client.

• Test Procedure
  1. Lower Tester sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x01 (Black List Filter).
  2. The Lower Tester expects the IUT to send a Proxy Configuration Message of “Filter Status” with a FilterType value equal to 0x01 (Black List) and a ListSize equal to [0].
  3. The Lower Tester sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray field set with 1 address equal to that of the Lower Tester.
  4. The Lower Tester expects the IUT to send a Proxy Configuration Message of “Filter Status” with a FilterType value equal to 0x01 (Black List) and a ListSize equal to [1].
  5. The Lower Tester sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray field set with 1 address equal to that of the Lower Tester.
  6. The Lower Tester expects the IUT to send a Proxy Configuration Message of “Filter Status” with a FilterType value equal to 0x01 (Black List) and a ListSize equal to [1].
  7. The Lower Tester sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray field set with 1 address equal to that of the unassigned address.
  8. The Lower Tester expects the IUT to send a Proxy Configuration Message of “Filter Status” with a FilterType value equal to 0x01 (Black List) and a ListSize equal to [1].
• Expected Outcome

Pass verdict

The IUT sends a Proxy PDU with MessageType field equal to 0x02 "Proxy Configuration Message" with OpCode 0x03 "Filter Status", a FilterType field value of 0x01 (Black List), and a ListSize field value of [0] in step 2.

The IUT sends a Proxy PDU with MessageType field equal to 0x02 "Proxy Configuration Message" with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x01 (Black List), and a ListSize field value of [1] in step 4.

The IUT sends a Proxy PDU with MessageType field equal to 0x02 "Proxy Configuration Message" with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x01 (Black List), and a ListSize field value of [1] in step 6.
The IUT sends a Proxy PDU with MessageType field equal to 0x02 “Proxy Configuration Message” with 0x03 “Filter Status” as the Proxy Configuration Status OpCode, a FilterType field value of 0x01 (Black List), and a ListSize field value of [1] in step 8.

Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address.

4.13.8 MESH/SR/PROX/BV-08-C [Filter Black List Addresses]

- **Test Purpose**
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly filter messages for the appropriate black list set.

- **Reference**
  [3] Section 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- **Initial Condition**
  - The IUT has performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network. Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - Test case MESH/SR/PROX/BV-07-C [Receive Black List Filter Setup] has been executed.

- **Test Procedure**
  1. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address of Lower Tester 1. The mesh message does not require reassembly at the IUT.
  2. The Lower Tester 1 expects the IUT should not send any Proxy PDU.
  3. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address that is not the address of the IUT but a valid mesh address. The Proxy PDU does not require reassembly at the IUT.
  4. The IUT sends a Proxy PDU with SAR field set to 00 (“Complete higher level message”), MessageType equal to 0x0 (“Network PDU”), and the complete mesh data to the Lower Tester 1.
  5. Lower Tester 1 expects a Proxy PDU from the IUT with the exact data contained in the Proxy PDU sent from the Lower Tester 2 to the IUT.

- **Expected Outcome**
  - **Pass verdict**

  The IUT does not send any Proxy PDUs to the Lower Tester 1 in step 2.

  The IUT sends a single Proxy PDU to the Lower Tester 1 with the exact data contained in the mesh message sent from Lower Tester 2 to the IUT in step 3, except TTL will be decremented by 1.
Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address.

4.13.9 MESH/SR/PROX/BV-09-C [Remove Address from White List]

- **Test Purpose**
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly remove addresses from the white list address list.

- **Reference**
  [3] Section 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- **Initial Condition**
  - The IUT has performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network. Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - The address of Lower Tester 1 has been added to the IUT's White list by running test case MESH/SR/PROX/BV-01-C [Receive White List Filter Setup].

- **Test Procedure**
  1. Lower Tester 1 sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x02 (Remove Addresses from Filter) and the AddressArray field set equal to the Lower Tester 1.
  2. The Lower Tester 1 expects the IUT to send a Proxy Configuration Message of “Filter Status” with a FilterType value equal to 0x00 (White List) and a ListSize equal to [0].
  3. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address of Lower Tester 1. The mesh message does not require reassembly at the IUT.
  4. The Lower Tester 1 expects the IUT not to send any Proxy PDUs.

- **Expected Outcome**
  **Pass verdict**
  The IUT sends a single Proxy Configuration Message with 0x03 “Filter Status” as the Proxy Configuration Status OpCode, a FilterType field value of 0x00 (White List), and a ListSize field value of [0] in step 2.
  The IUT does not send any Proxy PDUs to the Lower Tester 1 in step 4.
Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address.

4.13.10 MESH/SR/PROX/BV-10-C [Remove Address from Black List]

- **Test Purpose**
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly remove addresses from the black list address list.

- **Reference**
  [3] Section 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- **Initial Condition**
  - The IUT has performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network. Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - Test case MESH/SR/PROX/BV-07-C [Receive Black List Filter Setup] has been executed right before this test.

- **Test Procedure**
  1. Lower Tester 1 sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x02 (Remove Addresses from Filter) and the AddressArray field set to the address of Lower Tester 1.
  2. The Lower Tester 1 expects the IUT to send a Proxy Configuration Message of “Filter Status” with a FilterType value equal to 0x01 (Black List) and a ListSize equal to 0.
  3. Lower Tester 2 sends a mesh message to the IUT that will fit into one single Proxy PDU with the DST address set to the Lower Tester 1’s address.
  4. The Lower Tester 1 expects the IUT to send a Proxy PDU with SAR field set to 00 (“Complete higher level message”), MessageType equal to 0x00 (“Network PDU”), and the complete mesh data to the Lower Tester 1.

- **Expected Outcome**
  **Pass verdict**

  The IUT sends a single Proxy Configuration Message with 0x03 “Filter Status” as the Proxy Configuration Status OpCode, a FilterType field value of 0x01 (Black List), and a ListSize field value of 0 in step 2.

  The IUT sends the entire mesh message from Lower Tester 2 to the Lower Tester 1 in the Proxy PDU in step 4, with TTL value decremented by 1.
Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address.

4.13.11  MESH/SR/PROX/BV-11-C [Send Secure Network Beacon When Proxy Connection Established]

- Test Purpose
  Verify the IUT can send Secure Network beacons when a proxy connection is established.

- Reference
  [3] 3.3.2, 6.2, 6.6

- Initial Condition
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester is a Proxy Client.

- Test Procedure
  1. The Lower Tester establishes a connection with the IUT.
  2. The IUT sends a Secure Network beacon to the Lower Tester. The beacon has the Beacon Type set to 0x01 (“Secure Network beacon”), the Key Refresh Flag and IV Update Flag in the Flags field set to 0, the Network ID set to the current Network ID, the IV Index containing the current IV Index, and the Authentication Value set to a valid value computed from the NetKey and beacon field values.

- Expected Outcome
  Pass verdict
  The IUT sends a Secure Network beacon to the Lower Tester after connecting.
  The Lower Tester receives valid message data from the IUT.

4.13.12  MESH/SR/PROX/BV-12-C [Send Secure Network Beacon When IV Index Updated]

- Test Purpose
  Verify the IUT can send Secure Network beacons when the IV Index is updated.

- Reference
  [3] 3.3.2, 6.2, 6.6

- Initial Condition
  - The IUT is in a proxy connection with Lower Tester 1.
- The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
- Lower Tester 1 is the Proxy Client.
- Lower Tester 2 is another node in the mesh network.
- The IV Update Test Mode is activated on the IUT.

**Test Procedure**

1. The Upper Tester orders the IUT to send a Secure Network beacon to the Lower Tester 1. The beacon has the Beacon Type set to 0x01 (“Secure Network beacon”), the Key Refresh Flag (Flags bit 0) and IV Update Flag (Flags bit 1) set to 0, the Network ID set to the current Network ID, the IV Index containing the current IV Index, and the Authentication Value set to a valid value computed from the NetKey and beacon field values.
2. Lower Tester 2 sends a Secure Network beacon to the IUT, signaling an IV Index update. The beacon has the IV Update Flag (Flags bit 1) set to 1, and the IV Index is the new value used by the network.
3. The IUT sends a Secure Network beacon to the Lower Tester 1 to relay the IV Index value update. The beacon has the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 1, the Network ID set to the current Network ID, the IV Index set to the new IV Index, and the Authentication Value set to the computed value.

**Expected Outcome**

Pass verdict

The IUT sends a Secure Network beacon to the Lower Tester 1 after connecting.

After receiving the IV Index update from Lower Tester 2, the IUT begins advertising Secure Network beacons with the IV Update Flag set to 1.

The Lower Tester receives both Secure Network beacons with the correct values.

**4.13.13 MESH/SR/PROX/BV-13-C [Send Secure Network Beacon with Previous IV Index Value]**

**Test Purpose**

Verify the IUT sends Secure Network beacons after receiving Secure Network beacons with the previous IV Index value.

**Reference**

[3] 3.3.2, 6.2, 6.6

**Initial Condition**

- The IUT has performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.
- The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
- Lower Tester 1 is a Proxy Client.
- Lower Tester 2 is a node in the mesh network.
- Test Procedure

1. The Upper Tester orders the IUT to send Secure Network beacons. The beacon has the Key Refresh Flag (Flags bit 0) and IV Update Flag (Flags bit 1) disabled, the Network ID set to the current Network ID, the IV Index containing the current IV Index, and the Authentication Value set to a valid value computed from the NetKey and beacon field values.
2. Lower Tester 1 receives the Secure Network beacon advertised by the IUT and validates the data in its fields.
3. Lower Tester 2 sends a Secure Network beacon to the IUT with the IV Index value set to the previous IV Index (e.g., if the current value is 0x00000002, then the beacon sent by Lower Tester 2 contains 0x00000001).
4. The Upper Tester orders the IUT to send Secure Network beacons.
5. The IUT continues advertising the Secure Network beacon with the current IV Index value and all values as listed in step 1.
6. Lower Tester 1 receives the Secure Network beacon from the IUT and validates the data in its fields.
7. Lower Tester 1 sends a single Mesh Data Proxy Message to the IUT smaller than the negotiated MTU. The Proxy Message does not require reassembly at the IUT. The IV Index of the Proxy PDU is equal to the current IV Index value – 1.
8. The IUT sends a Proxy Message with SAR field set to 00 (“Complete higher level message”), RFU set to zero, Type equal to 0x0 (“Mesh Data”), and the complete mesh data to the Lower Tester 2.
9. Lower Tester 2 receives a Proxy Message from the IUT with the exact data contained in the Proxy Message sent from Lower Tester 1 to the IUT.

Figure 4.69: MESH/SR/PROX/BV-13-C
• Expected Outcome

Pass verdict

The IUT sends a Secure Network beacon to the Lower Tester 1 after connecting.

The IUT continues sending Secure Network beacons with the current IV Index after receiving a Secure Network beacon from Lower Tester 2 with the previous IV Index.

Lower Tester 1 receives each of the Secure Network beacons from the IUT with the correct values.

The IUT transmits the Proxy Messages sent by Lower Tester 1 to the Lower Tester 2.

4.13.14 MESH/SR/PROX/BV-14-C [IV Update Initiated by Proxy Client]

• Test Purpose

Verify that an IUT acting as Proxy Server can process an IV Update initiated by a Proxy Client.

• Reference

[3] Section 3.3.2, 3.10.5, 3.10.5.1, 6.6

• Initial Condition

- The IUT is in a proxy connection with Lower Tester 1.
- The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
- Lower Tester 1 is the Proxy Client.
- Lower Tester 2 is another node in the mesh network.
- The IV Update Test Mode is activated on the IUT.

• Test Procedure

1. The Lower Tester 2 generates a Secure Network beacon using the same network credentials and current IV Index as the IUT and begins advertising this beacon periodically. The IV Update Flag and the Key Refresh Flag are set to 0.
2. The Lower Tester 2 expects to receive Secure Network beacons with the same values from the IUT.
3. The Lower Tester 1 increases its IV Index by one, then sends a Proxy PDU containing a new Secure Network beacon with the new IV Index value, the IV Update Flag set to 1 and the Key Refresh Flag set to 0.
4. The Lower Tester 2 expects to receive the new Secure Network beacon containing the increased IV Index value and the IV Update Flag set to 1.
5. The Lower Tester 1 sends a Proxy PDU containing a new Secure Network beacon with the new IV Index value, the IV Update Flag set to 0 and the Key Refresh Flag set to 0.
6. The Lower Tester 2 expects to receive the new Secure Network beacon containing the increased IV Index value and the IV Update Flag set to 0.
- Expected Outcome
  
  **Pass verdict**

  In step 2, the IUT sends the same Secure Network beacon as Lower Tester 2.

  In step 5, the IUT sends the new Secure Network beacon with the increased IV Index and the IV Update Flag set to 1.

  In step 7 the IUT sends the new Secure Network beacon with the increased IV Index and the IV Update Flag set to 0.

---

**4.13.15 MESH/SR/PROX/BI-01-C [Ignore Invalid Message Type]**

- Test Purpose
  
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly handle a Proxy PDU with an invalid Message PDU Type.

- Reference
  
  [3] Section 6.2, 6.5, 6.6

- Initial Condition
  
  - The IUT has just performed a proxy connection to the Lower Tester 1 through a connection oriented channel from another test case.

  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).

  - Lower Tester 1 is a Proxy Client.

  - Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.

  - There have been no Addresses added to any Proxy Address List.
• Test Procedure
  1. Lower Tester 2 sends a single Proxy PDU to the IUT smaller than the negotiated MTU with the
destination address of the Lower Tester 1 and the “MessageType” field of the Proxy PDU set to a
random RFU value.
  2. The IUT ignores the Proxy PDU from Lower Tester 2 and does not send any mesh message or
Proxy PDU in response.

• Expected Outcome
  Pass verdict

  In step 2, the IUT does not send any mesh message or Proxy PDU in response.

4.13.16  MESH/CL/PROX/BV-01-C [Receive Complete Message from Proxy Server]
• Test Purpose
  Verify that an IUT can receive a message from a Proxy Server where the complete message fits into
one PDU.

• Reference

• Initial Condition
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented
channel from another test case.
  - The IUT is the Proxy Client.
  - Lower Tester is a Proxy Server.
  - The IUT has a mechanism to add its own unicast address to the Proxy white list.

• Test Procedure
  1. Lower Tester sends a single mesh message to the IUT smaller than the negotiated MTU and with
the DST address of the IUT. The Proxy PDU does not require reassembly at the IUT with SAR
field set to 00 (“Complete higher level message”), MessageType equal to 0x00 (“Mesh Data”),
and Data field contains the complete mesh data to the IUT.
  2. IUT receives a Proxy PDU from the Lower Tester.

• Expected Outcome
  Pass verdict

  The IUT receives the Proxy PDU and reports to the Upper Tester that it has received the mesh
message contained in the Proxy PDU from the Lower Tester.

4.13.17  MESH/CL/PROX/BV-02-C [Receive Segmented Message from Proxy Server]
• Test Purpose
  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can receive a message
from a Proxy Server where the complete message does not fit into one PDU.
• Reference

• Initial Condition
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.
  - Lower Tester is a Proxy Server.

• Test Procedure
  1. The Lower Tester sends a Proxy PDU to the IUT with SAR field set to b01 (“Data field contains the first segment of a message”), MessageType equal to 0x00 (“Network PDU”), and the Data field contains the first segment of the mesh data to the IUT.
  2. The Lower Tester then sends a Proxy PDU to the IUT with SAR field set to b11 (“Data field contains the last segment of a message”), MessageType equal to 0x00 (“Network PDU”), and Data field contains the last segment of the mesh data to the IUT.
  3. The Lower Tester then sends another Proxy PDU to the IUT with SAR field set to b01 (“Data field contains the first segment of a message”), MessageType equal to 0x00 (“Network PDU”), and the Data field contains the first segment of the mesh data to the IUT.
  4. Lower Tester then waits greater than 20 seconds.
  5. The Lower Tester expects the IUT to terminate the connection.

• Expected Outcome
  Pass verdict

The IUT receives all Proxy PDUs from the Lower Tester and reports to the Upper Tester that it has received the complete mesh message from Lower Tester.

In Step 5, the IUT terminates the connection.

4.13.18 MESH/CL/PROX/BV-03-C [Send Complete Message to Proxy Server]

• Test Purpose
  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send a message to a Proxy Server where the complete message fits into one PDU.

• Reference

• Initial Condition
  - The IUT has just performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.
  - Lower Tester is a Proxy Server.
  - There have been no Addresses added to any Proxy Address Filter List.
  - The IUT has a method to initiate sending a non-segmented message to the Lower Tester.
• Test Procedure
  1. The Upper Tester orders the IUT to send a single Proxy PDU to the Lower Tester smaller than the negotiated MTU with the destination address of the Lower Tester. The Proxy PDU does not require reassembly.
  2. The Lower Tester receives the Proxy PDU from the IUT.

• Expected Outcome
  **Pass verdict**
  The IUT sends a single Proxy PDU to Lower Tester with SAR field set to b00 ("Complete higher level message"), MessageType equal to 0x00 ("Network PDU"), and Data field contains the complete mesh data.

4.13.19 MESH/CL/PROX/BV-04-C [Send Segmented Message to Proxy Server]

• Test Purpose
  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send a message to a Proxy Server where the complete message does not fit into one PDU.

• Reference

• Initial Condition
  - The IUT has just performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.
  - Lower Tester is a Proxy Server.
  - There have been no Addresses added to any Proxy Address List.

• Test Procedure
  1. The Upper Tester orders the IUT to send a mesh message to the Lower Tester through the Proxy PDU to the IUT which has a larger payload size than the negotiated MTU for that connection and with the destination address of the Lower Tester.
  2. The IUT sends a Proxy PDU to the IUT with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x00 ("Network PDU"), and Data field contains the first segment of the mesh data.
  3. If the message size in step 1 was greater than 2 times the negotiated MTU for the Proxy connection then the IUT sends a Proxy PDU to the IUT with SAR field set to b10 ("Data field contains the next segment of a message"), MessageType equal to 0x00 ("Network PDU"), and Data field contains the next segment of the mesh data.
  4. Step 3 is repeated until one before the last segment is received.
  5. If the message size in step 1 was not greater than 2 times the negotiated MTU for that Proxy connection then steps 3 and 4 will not occur and the IUT then sends a Proxy PDU to the Lower Tester with SAR field set to b11 ("Data field contains the last segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the last part of the mesh data to the Lower Tester.
6. Lower Tester receives all Proxy PDUs from the IUT with the exact data contained in the entire message.

![Diagram of MESH/CL/PROX/BV-04-C](image)

**Figure 4.71: MESH/CL/PROX/BV-04-C**

- **Expected Outcome**
  - **Pass verdict**

The IUT sends the Proxy PDUs in steps 2 with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x0 ("Network PDU"), and the Data field containing first segment of the mesh message.

The IUT sends all the Proxy PDUs in steps 3 – 4 with SAR field set to b10 ("Data field contains the next segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the next of the mesh message.

The IUT sends the last Proxy PDUs in step 4 with SAR field set to b11 ("Data field contains the last segment of a message"), RFU set to zero, MessageType equal to 0x0 ("Network PDU"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the final segment of the mesh message.

All the Proxy PDU data from the mesh message is reassembled.

### 4.13.20 MESH/CL/PROX/BV-05-C [Send Set Filter Type Proxy Configuration Message]

- **Test Purpose**
  - Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send a Set Filter Type Proxy Configuration Message.
• Reference

[3] Section 6.2, 6.3, 6.5, 6.7

• Initial Condition
- The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
- The IUT is the Proxy Client.
- Lower Tester is a Proxy Server.

• Test Procedure
  1. The Upper Tester orders the IUT to send a Set Filter Type Proxy PDU to the Lower Tester.
  2. The Lower Tester expects to receive a Set Filter Type Proxy PDU with the DST field with the unassigned address, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x00 (White List Filter) or 0x01 (Black List Filter).

• Expected Outcome

Pass verdict

The IUT sends a Set Filter Type Proxy PDU with the DST field set to the unassigned address, SRC address field set to the unicast address of its primary element, SEQ field set to the sequence number of its primary element, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x00 (White List Filter) or 0x01 (Black List Filter).

4.13.21 MESH/CL/PROX/BV-06-C [Send Add Addresses to Filter Proxy Configuration Message]

• Test Purpose

Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send an Add Addresses to Filter Proxy Configuration Message.

• Reference

[3] Section 6.2, 6.3, 6.4, 6.5, 6.7

• Initial Condition
- The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
- The IUT is the Proxy Client.
- Lower Tester is a Proxy Server.

• Test Procedure
  1. The Upper Tester orders the IUT to send a Proxy Configuration message with OpCode 0x01 (Add Addresses to Filter) to the Lower Tester.
  2. The Lower Tester expects to receive an Add Addresses to Proxy PDU with the DST field set to the unassigned address, a CTL field value set to 1, and a TTL field value set to 0. The Proxy
Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray equal to n * length of an address.

- Expected Outcome

**Pass verdict**

The IUT sends an Add Addresses Filter Proxy PDU with the DST field set to the unassigned address, SRC address field set to the unicast address of its primary element, SEQ field set to the sequence number of its primary element, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray equal to n * length of an address.

4.13.22  MESH/CL/PROX/BV-07-C [Send Remove Addresses from Filter Proxy Configuration Message]

- Test Purpose

Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send a Remove Addresses from Filter Proxy Configuration Message.

- Reference

[3] Section 6.2, 6.3, 6.4, 6.5, 6.7

- Initial Condition

  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  
  - The IUT is the Proxy Client.
  
  - Lower Tester is a Proxy Server.

- Test Procedure

  1. **The Upper Tester orders the IUT send a Proxy Configuration message with OpCode 0x02 (Remove Addresses from Filter) to the Lower Tester.**
  2. **The Lower Tester expects to receive a Remove Addresses to Proxy PDU with the DST field set to the unassigned address, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x02 (Remove Addresses to Filter) and the AddressArray equal to n * length of an address.**

- Expected Outcome

**Pass verdict**

The IUT sends a Remove Addresses Filter Proxy PDU with the DST field set to the unassigned address, SRC address field set to the unicast address of its primary element, SEQ field set to the sequence number of its primary element, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x02 (Remove Addresses to Filter) and the AddressArray equal to n * length of an address.
4.13.23 MESH/CL/PROX/BV-08-C [Receive Secure Network Beacon When Proxy Connection Established]

- **Test Purpose**
  Verify the Proxy Client IUT can receive Secure Network beacons when a proxy connection is established.

- **Reference**
  [3] 3.3.2, 6.2, 6.7

- **Initial Condition**
  - The IUT is a Proxy Client.
  - The Lower Tester is a Proxy Server.

- **Test Procedure**
  1. The Upper Tester orders the IUT to connect to the Lower Tester.
  2. The Lower Tester sends a Secure Network beacon to the IUT. The beacon has the Key Refresh Signaling bit and IV Update Signaling bit of the SecurityFlags field set to 0, the Network ID field containing the eight least significant bytes of the Device UUID, the IV Index containing the current IV Index, and a valid MIC for authenticating the packet contents. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set to 0x01 ("Secure Network beacon").

- **Expected Outcome**
  Pass verdict
  The IUT receives valid Secure Network beacon data from the Lower Tester.

4.13.24 MESH/CL/PROX/BV-09-C [Send Secure Network Beacon When IV Index Updated]

- **Test Purpose**
  Verify the IUT can receive Secure Network beacons when the IV Index is updated.

- **Reference**
  [3] 3.3.2, 6.2, 6.7

- **Initial Condition**
  - The IUT has just performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is a Proxy Client.
  - Lower Tester is a Proxy Server.

- **Test Procedure**
  1. The Lower Tester sends a Secure Network beacon to the IUT. The beacon has the Key Refresh Signaling bit and IV Update Signaling bit of the SecurityFlags field set to 0, the Network ID field containing the eight least significant bytes of the Device UUID, the IV Index containing the current
IV Index, and a valid MIC for authenticating the packet contents. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set correctly.

2. The Lower Tester sends a Secure Network beacon to the IUT to relay the IV Index value update. The beacon has the Key Refresh bit set to 0 and the IV Update bit set to 1 in the Flags field, the Network ID field containing the eight least significant bytes of the Device UUID, the IV Index containing the new IV Index value, and a valid MIC for authenticating the packet contents. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set correctly.

3. The IUT clears its oldest previously accepted IV index and saves a replay protection list for the new IV index.

• Expected Outcome

Pass verdict

After receiving the IV Index update from Lower Tester, the IUT updates its IV Index.

4.13.25 MESH/CL/PROX/BV-10-C [Send Secure Network Beacon with Previous IV Index Value]

• Test Purpose

Verify the Proxy Client IUT behaves appropriately when receiving Secure Network beacons with a previous IV Index value.

• Reference

[3] 3.3.2, 6.2, 6.7

• Initial Condition

- The IUT has just performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.

- The IUT is a Proxy Client.

- Lower Tester is a Proxy Server.

• Test Procedure

1. The Lower Tester sends a Secure Network beacon to the IUT. The beacon has the Key Refresh bit and IV Update bit of the Flags field set to 0, the Network ID field containing the eight least significant bytes of the Device UUID, the IV Index containing the current IV Index, and a valid MIC for authenticating the packet contents. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set correctly.

2. The Lower Tester sends a Secure Network beacon to the IUT to relay an IV Index value update. The beacon has the Key Refresh Signaling bit set to 0 and the IV Update Signaling bit set to 1 in the Security Flags field, the Network ID field containing the eight least significant bytes of the Device UUID, the IV Index containing the new IV Index value, and a valid MIC for authenticating the packet contents. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set correctly.

3. Lower Tester sends a single Mesh Data Proxy Message to the IUT smaller than the negotiated MTU. The Proxy Message does not require reassembly at the IUT. The IV Index of the Mesh Packet is equal to the current IV Index value – 1.
• Expected Outcome
  Pass verdict

The IUT receives the packet with the previous IV Index from the Lower Tester.

4.13.26  MESH/CL/PROX/BI-01-C [Ignore Invalid Message Type]

• Test Purpose
  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Client role can properly handle a Proxy PDU with an invalid Message PDU Type.

• Reference

• Initial Condition
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.
  - Lower Tester is a Proxy Server.

• Test Procedure
  1. Lower Tester sends a single Proxy PDU to the IUT smaller than the negotiated MTU and with the DST address of IUT. The Proxy PDU does not require reassembly at the IUT with SAR field set to 00 (“Complete higher level message”), RFU set to zero, Type equal to a random RFU value.
  2. IUT ignores the Proxy PDU due to the invalid MessageType.

• Expected Outcome
  Pass verdict

  IUT ignores the Proxy PDU due to the invalid MessageType.

4.14 Mesh Proxy Service

The test group objective is to verify the Mesh Proxy Service.

4.14.1 MESH/SR/MPXS/BV-01-C [Advertise Mesh Proxy Service with Network ID]

• Test Purpose
  Verify that an IUT supporting the GATT Bearer Server role can properly advertise the Mesh Proxy Service with the Network ID.

• Reference
  [3] 6.2, 7.2.2.1, 7.2.2.2, 7.2.2.2.1

• Initial Condition
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
- The IUT supports the GAP Broadcaster role.
- The Lower Tester is a GAP Observer.
- The IUT and the Lower Tester share previously established network security credentials.

**Test Procedure**

1. The Upper Tester orders the IUT to start advertising the Mesh Proxy Service with Network ID.
2. The Lower Tester expects the IUT to send Advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 9 octets of service data.

**Expected Outcome**

Pass verdict

In step 2, the IUT sends advertising packets with the Advertising Data field containing the required AD Types, formatted correctly and including the «Mesh Proxy Service» and the 17-octet service data. The first octet of the service data is equal to 0x00, and the remaining 8 octets are equal to the Network ID derived with the k3 function from the shared network key. The Flags AD indicates that the IUT is in the GAP General Discoverable Mode.

### 4.14.2 MESH/SR/MPXS/BV-02-C [Expose Mesh Proxy Service]

**Test Purpose**

Verify that an IUT supporting the GATT Bearer Server role can expose the Mesh Proxy Service in the GATT database.

**Reference**

[3] 6.2, 7.2.2.1, 7.2.2.2

**Initial Condition**

- The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
- The Lower Tester is a GATT Bearer Client.
- The IUT and the Lower Tester support GAP Peripheral and GAP Central roles, respectively.
- Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

**Test Procedure**

• Expected Outcome

Pass verdict

In step 1, the IUT returns a single attribute handle range. The starting handle and the ending handle correspond to the Mesh Proxy Service definition in the GATT database of the IUT.

In step 2, the IUT returns no instance of the Mesh Provisioning Service.

4.14.3 MESH/SR/MPXS/BV-03-C [Expose Mesh Proxy Data Out Characteristic]

• Test Purpose

Verify that an IUT supporting the GATT Bearer Server role can expose the Mesh Proxy Data Out Characteristic in the GATT database.

• Reference

[3] 7.2.1, 7.2.2, 7.2.3.2

• Initial Condition

- The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
- The Lower Tester is a GATT Bearer Client.
- Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
- The procedure described in
- MESH/SR/MPXS/BV-02-C [Expose Mesh Proxy Service] has been executed and the Lower Tester retained the handle range of the Mesh Proxy Service.

• Test Procedure

The Lower Tester executes either of the procedures included in GATT.TS [5]: Discover All Characteristic of a Service, GATT/SR/GAD/BV-04-C, with the handle range of the Mesh Proxy Service, or Discover Characteristic by UUID, GATT/CL/GAD/BV-05-C, with the handle range of the Mesh Proxy Service and UUID set to «Mesh Proxy Data Out».

• Expected Outcome

Pass verdict

The IUT returns a single characteristic with the UUID equal to «Mesh Proxy Data Out» in the specified handle range. The characteristic properties value has the Notifications flag set to 1.


• Test Purpose

Verify that an IUT supporting the GATT Bearer Server role can expose the Mesh Proxy Data Out Client Characteristic Configuration descriptor in the GATT database.
Mesh Profile (MESH) / Test Suite

- Reference
  [3] 7.2.1, 7.2.2, 7.2.3.2

- Initial Condition
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The procedure described in MESH/SR/MPXS/BV-03-C [Expose Mesh Proxy Data Out Characteristic] has been executed and the Lower Tester retained the handle range of the Mesh Proxy Data Out characteristic.

- Test Procedure
  The Lower Tester executes one pass of the procedure included in GATT.TS [5]: Discover all Characteristic Descriptors – from Server, GATT/SR/GAD/BV-06-C using the handle range of the Mesh Proxy Data Out characteristic.

- Expected Outcome
  Pass verdict
  The IUT returns a descriptor with the UUID equal to «Client Characteristic Configuration» in the specified handle range.


- Test Purpose
  Verify that an IUT supporting the GATT Bearer Server role supports GATT Notifications on the Mesh Proxy Data Out characteristic.

- Reference
  [3] 7.2.1, 7.2.2, 7.2.3.2

- Initial Condition
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The procedure described in MESH/SR/MPXS/BV-04-C [Expose Mesh Proxy Data Out Client Characteristic Configuration Descriptor] has been executed and the Lower Tester retained the handle of the Mesh Proxy Data Out Client Characteristic Configuration Descriptor.
• Test Procedure
  1. The Lower Tester enables notifications on the Mesh Proxy Data Out by executing the preamble procedure in GATT.TS [5] Section 4.2.2.1, using the handle of the Mesh Proxy Data Out Client Characteristic Configuration Descriptor.
  2. The Lower Tester expects the IUT to send an ATT Handle Value Notification packet containing a Secure Network beacon.

• Expected Outcome

  Pass verdict

In step 1, the IUT accept the procedure and enables the notifications on the Mesh Proxy Data Out characteristic.

In step 2, the IUT sends an ATT Handle Value Notification packet with the Attribute Handle field set to the handle of the Mesh Proxy Data Out characteristic value, and the Attribute Value field containing a Secure Network beacon with correct format and valid field values.

4.14.6 MESH/SR/MPXS/BV-06-C [Expose Mesh Proxy Data In Characteristic]

• Test Purpose

Verify that an IUT supporting the GATT Bearer Server role can expose the Mesh Proxy Data In Client Characteristic Configuration descriptor in the GATT database.

• Reference

[3] 7.2.1, 7.2.2, 7.2.3.1

• Initial Condition

- The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
- The Lower Tester is a GATT Bearer Client.
- Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
- The procedure described in
  - MESH/SR/MPXS/BV-02-C [Expose Mesh Proxy Service] has been executed and the Lower Tester retained the handle range of the Mesh Proxy Service.

• Test Procedure

The Lower Tester executes either of the procedures included in GATT.TS [5]: Discover All Characteristic of a Service, GATT/SR/GAD/BV-04-C, with the handle range of the Mesh Proxy Service, or Discover Characteristic by UUID, GATT/CL/GAD/BV-05-C, with the handle range of the Mesh Proxy Service and UUID set to «Mesh Proxy Data In».

• Expected Outcome

  Pass verdict

The IUT returns a single characteristic with the UUIT equal to «Mesh Proxy Data In» in the specified handle range. The characteristic properties value has the Write Without Response flag set to 1.
4.14.7 MESH/SR/MPXS/BV-07-C [Support Write Without Response to Mesh Proxy Data In Characteristic]

- **Test Purpose**
  Verify that an IUT supporting the GATT Bearer Server role supports GATT Write Without Response on the Mesh Proxy Data In characteristic.

- **Reference**
  [3] 7.2.1, 7.2.2, 7.2.3.1

- **Initial Condition**
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The procedure described in MESH/SR/MPXS/BV-06-C [Expose Mesh Proxy Data In Characteristic] has been executed and the Lower Tester retained the handle of the Mesh Proxy Data In characteristic value.
  - The procedure described in MESH/SR/MPXS/BV-05-C [Support Notifications for Mesh Proxy Data Out Characteristic] has been executed.

- **Test Procedure**
  1. The Lower Tester sends an ATT Write Command packet with the Attribute Handle field set to the handle of the Mesh Proxy Data In characteristic value, and the Attribute Value set to a properly formatted Set Filter Type proxy configuration message.
  2. The Lower Tester expects the IUT to respond with an ATT Handle Value Notification packet with the Attribute Handle field set to the handle of the Mesh Proxy Data Out characteristic value, and the Attribute Value field set to a Filter Status proxy configuration message with correct format and parameters.

- **Expected Outcome**
  **Pass verdict**

  In step 2, the IUT responds with a Filter Status message, confirming that it has received the ATT Write Command packet from the Lower Tester in step 1.

4.14.8 MESH/SR/MPXS/BV-09-C [Advertise Mesh Proxy Service with Node Identity]

- **Test Purpose**
  Verify that an IUT supporting the GATT Bearer Server role can properly advertise the Mesh Proxy Service with Node Identity.

- **Reference**
  [3] 7.2.1, 7.2.2
Mesh Profile (MESH) / Test Suite

- Initial Condition
  - The IUT exposes the Mesh Proxy Service.
  - The IUT supports the GAP Broadcaster role.
  - The Lower Tester is a GAP Observer.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure
  1. The Upper Tester orders the IUT to start advertising the Mesh Proxy Service with Node Identity.
  2. The Lower Tester expects the IUT to send advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 17 octets of service data.

- Expected Outcome
  Pass verdict

In step 2, the IUT sends advertising packets with the Advertising Data field containing the required AD Types, formatted correctly and including the «Mesh Proxy Service» and the 17-octet service data. The first octet of the service data is equal to 0x01, the next 8 octets contain a Random value and the last 8 octets contain a Hash value properly computed over the Random value with the IdentityKey of the IUT.

4.14.9 MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server]

- Test Purpose
  Verify that an IUT supporting the GATT Bearer Client role can find and connect to a GATT Bearer Server.

- Reference
  [3] 7.2.2

- Initial Condition
  - The IUT is a GATT Bearer Client.
  - The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure
  1. The Upper Tester orders the IUT to begin scanning for GATT Bearer Server advertisements and automatically connect to devices advertising the Mesh Proxy Service.
  2. The Lower Tester begins sending connectable advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 9 octets of service data. The first octet is 0x00 and the next 8 octets contain the Network ID.
3. The Lower Tester expects the IUT to establish a GATT connection upon receiving the advertisements.

   • Expected Outcome

     Pass verdict

     In step 3, the IUT establishes a GATT connection with the Lower Tester.

4.14.10 MESH/CL/MPXS/BV-02-C [Discover Mesh Proxy Service]

   • Test Purpose

     Verify that an IUT supporting the GATT Bearer Client role can discover the Mesh Proxy Service on a GATT Bearer Server.

   • Reference

     [3] 7.2.2

   • Initial Condition

     - The IUT is a GATT Bearer Client.
     - The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
     - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
     - The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.

   • Test Procedure

     1. The Upper Tester orders the IUT to discover the Mesh Proxy Service on the Lower Tester either by executing the procedure in GATT.TS [5] Discover Primary Services by Service UUID – By Client, GATT/CL/GAD/BV-02-C, with the Service UUID of «Mesh Proxy Service» or the procedure in GATT.TS [5] Discover All Primary Services – By Client, GATT/CL/GAD/BV-01-C.

     2. The Upper Tester expects the IUT to report the correct attribute handle range of the Mesh Proxy Service from the GATT database on Lower Tester.

   • Expected Outcome

     Pass verdict

     In step 1, the IUT executes the discovery procedure to obtain the attribute handle range of the Mesh Proxy Service on the Lower Tester.

     In step 2, the IUT reports the correct attribute handle range of the Mesh Proxy Service to the Upper Tester of the Mesh Proxy Service.

4.14.11 MESH/CL/MPXS/BV-03-C [Discover Mesh Proxy Data Out Characteristic]

   • Test Purpose

     Verify that an IUT supporting the GATT Bearer Client role can discover the Mesh Proxy Data Out characteristic on a GATT Bearer Server.
• Reference

[3] 7.2.2, 7.2.3.2

• Initial Condition

- The IUT is a GATT Bearer Client.
- The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
- The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
- The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.

• Test Procedure

1. The Upper Tester orders the IUT to execute either of the procedures included in GATT.TS [5]: Discover All Characteristics of a Service – by Client, GATT/CL/GAD/BV-04-C, with the specified handle range for the instantiation of the Mesh Proxy Service, or Discover Characteristics by UUID – by Client, GATT/CL/GAD/BV-05-C, with the specified handle range for the instantiation of the Mesh Proxy Service and UUID set to «Mesh Proxy Data Out». In the selected procedure, only one pass is needed with the server database defined in Initial Condition.
2. The Upper Tester expects the IUT to report the correct handle range for the Mesh Proxy Data Out characteristic from the GATT database on the IUT.

• Expected Outcome

Pass verdict

In step 1, the IUT executes the discovery procedure to obtain the attribute handle range of the Mesh Proxy Data Out characteristic on the Lower Tester.

In step 2, the IUT reports the correct attribute handle range to the Upper Tester.


• Test Purpose

Verify that an IUT supporting the GATT Bearer Client role can discover the Mesh Proxy Data Out Characteristic Configuration Descriptor on a GATT Bearer Server and can enable and process GATT Notifications for that characteristic.

• Reference

[3] 7.2.2, 7.2.3.2

• Initial Condition

- The IUT is a GATT Bearer Client.
- The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
- The IUT and the Lower Tester share previously established network security credentials.
- The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
- The procedure described in Mesh Profile (MESH) / Test Suite / Bluetooth SIG Proprietary Page 237 of 432

- The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.

  • Test Procedure
    1. The Upper Tester orders the IUT to execute one pass of the procedure included in GATT.TS [5]: Discover all Characteristic Descriptors, GATT/CL/GAD/BV-06-C using the specified handle range.
    2. The Upper Tester expects the IUT to report the correct attribute handle for the Mesh Proxy Data Out Client Characteristic Configuration Descriptor from the GAT database on the IUT.
    3. The Upper Tester orders the IUT to enable notifications on the Mesh Proxy Data Out characteristic.
    4. The Lower Tester expects the IUT to execute the procedure described in GATT.TS [5]: GATT/CL/GAW/BV-08-C [Write Characteristic Descriptors – by Client] with the handle of the Client Characteristic Configuration Descriptor and the value 0x0001.
    5. The Lower Tester sends an ATT Handle Value Notification packet to the IUT with the Attribute Handle field set to the handle of the Mesh Proxy Data Out characteristic value and the Attribute Value field containing a Secure Network beacon with correct format and valid field values.
    6. The Upper Tester expects the IUT to report the received Secure Network beacon.

  • Expected Outcome
    Pass verdict

    In step 1, the IUT executes the discovery procedure to obtain the handle of the Mesh Proxy Data Out Client Characteristic Configuration Descriptor on the Lower Tester.

    In step 2, the IUT reports the correct attribute handle to the Upper Tester.

    In step 4, the IUT executes the write procedure to enable notifications on the Mesh Proxy Data Out characteristics.

    In step 6, the IUT reports it has received a valid Secure Network beacon.


  • Test Purpose
    Verify that an IUT supporting the GATT Bearer Client role can discover the Mesh Proxy Data In characteristic on a GATT Bearer Server.

  • Reference
    [3] 7.2.2, 7.2.3.1

  • Initial Condition
    - The IUT is a GATT Bearer Client.
    - The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
    - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
    - The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.
**Test Procedure**

1. The Upper Tester orders the IUT to execute either of the procedures included in GATT.TS [5]: Discover All Characteristic of a Service, GATT/SR/GAD/BV-04-C, with the specified handle range for the instantiation of the Mesh Proxy Service, or Discover Characteristic by UUID, GATT/SR/GAD/BV-05-C, with the specified handle range for the instantiation of the Mesh Proxy Service and UUID set to «Mesh Proxy Data In». In the selected procedure, only one pass is needed with the server database defined in Initial Condition.

2. The Upper Tester expects the IUT to report the correct handle range for the Mesh Proxy Data In characteristic from the GAT database on the IUT.

**Expected Outcome**

*Pass verdict*

In step 1, the IUT executes the discovery procedure to obtain the attribute handle range of the Mesh Proxy Data In characteristic on the Lower Tester.

In step 2, the IUT reports the correct attribute handle range to the Upper Tester.

---


**Test Purpose**

Verify that an IUT supporting the GATT Bearer Client role can execute GATT Write Without Response on the Mesh Proxy Data In characteristic.

**Reference**

[3] 7.2.2, 7.2.3, 7.2.3.1

**Initial Condition**

- The IUT is a GATT Bearer Client.
- The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
- The IUT and the Lower Tester share previously established network security credentials.
- The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.
- The procedure described in MESH/CL/MPXS/BV-05-C [Discover Mesh Proxy Data In Characteristic] has been executed.

**Test Procedure**

1. The Upper Tester orders the IUT to send an ATT Write Command packet with the Attribute Handle field set to the handle of the Mesh Proxy Data In characteristic value and the Attribute Value field containing a proxy configuration message.

2. The Lower Tester expects the IUT to send the proper ATT Write Command packet.

**Expected Outcome**

*Pass verdict*

In step 2, the IUT sends an ATT Write Command packet with the proper fields.
4.15 Configuration Model

The test objective is to verify functionality specific to the Configuration Model for both the Configuration Server Model and the Configuration Client Model.

4.15.1 Secure Network Beacon Procedures

This section is for validating the Secure Network Beacon procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.1.1 MESH/NODE/CFG/SNBP/BV-01-C [Respond to Config Beacon Set/Get Messages]

• Test Purpose
  Verify that an IUT node supporting the Configuration Server Model can respond to Config Beacon Set and Get messages.

• Reference
  [3] Section 3.9.3, 4.2.10, 4.3.2.1, 4.3.2.2, 4.3.2.3, 4.4.1.2, 4.4.1.2.1, 4.4.2.2.1

• Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.

• Test Procedure
  1. The Lower Tester sends a Config Beacon Set message to the IUT with a Beacon field value of 0x01.
  2. The Lower Tester expects to receive a Config Beacon Status message from the IUT with the Beacon field set to 0x01.
  3. The Lower Tester expects the IUT to start broadcasting Secure Network beacons within 20 seconds.
  4. The Lower Tester sends a Config Beacon Get message to the IUT.
  5. The Lower Tester expects to receive a Config Beacon Status message from the IUT with the Beacon field set to 0x01.
  6. The Lower Tester sends a Config Beacon Set message to the IUT with a Beacon field value of 0x00 (The node is not beaconing a Secure Network Beacon).
  7. The Lower Tester expects to receive a Config Beacon Status message from the IUT with the Beacon field set to 0x00.
  8. The Lower Tester scans for 20 seconds to verify that the IUT stops broadcasting Secure Network beacons.
  9. The Lower Tester sends a Config Beacon Get message to the IUT.
  10. The Lower Tester expects to receive a Config Beacon Status message from the IUT with the Beacon field set to 0x00.
Figure 4.72: MESH/NODE/CFG/SNBP/BV-01-C

- **Expected Outcome**

  **Pass verdict**

  In step 2 and step 5, the IUT sends a Config Beacon Status message with the Beacon field set to 0x01.

  In step 3, the IUT sends at least one Secure Network beacon.

  In step 8, the IUT does not send Secure Network beacons.

  In step 10, the IUT sends a Config Beacon Status message with the Beacon field set to 0x00.

  All messages are encrypted using the device key of the IUT.
4.15.1.2 MESH/NODE/CFG/SNBP/BI-01-C [Ignore invalid Config Beacon Set/Get Messages]

- Test Purpose
  Verify that an IUT node supporting the Configuration Server Model ignores invalid Config Beacon Set and Get messages.

- Reference
  [3] Section 3.9.3, 4.2.10, 4.3.2.1, 4.3.2.2, 4.3.2.3, 4.4.1.2.1, 4.4.2.2.1

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- Test Procedure
  1. The Lower Tester sends a Config Beacon Get message to the IUT.
  2. The Lower Tester expects the IUT to respond with a Config Beacon Status message.
  3. The Lower Tester sends a Config Beacon Set message to the IUT with a Beacon field value of 0x02 (value is Reserved for Future Use).
  4. The Lower Tester expects the IUT to ignore the Config Beacon Set message and not respond with any Config Beacon Status message.
  5. The Lower Tester sends a Config Beacon Get message to the IUT.
  6. The Lower Tester expects the IUT to respond with a Config Beacon Status message.

![Diagram of test procedure](image_url)
• Expected Outcome

Pass verdict

In step 4, the IUT does not send any Config Beacon Status message.

In step 6, the Beacon field in the Config Beacon Status message has the same value as the one obtained in step 2.

All configuration messages are encrypted using the device key of the IUT.

4.15.1.3 MESH/CFGCL/CFG/SNBP/BV-01-C [Send Config Beacon Set/Get Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config Beacon Get messages to a Configuration Server.

• Reference

[3] Section 4.2.10, 4.3.2.1, 4.3.2.2, 4.4.1.2.1, 4.4.2.2.1

• Initial Condition

- The IUT supports the Configuration Client Model.
- The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure

1. The Upper Tester orders the IUT to send a Config Beacon Get message to the Lower Tester.
2. The Lower Tester expects to receive a Config Beacon Get message and responds with a Config Beacon Status message.
3. The Upper Tester orders the IUT to send a Config Beacon Set message to the Lower Tester with the Beacon field value set to 0x00.
4. The Lower Tester expects to receive a Config Beacon Set message and responds by sending a Config Beacon Status message with the Beacon field value set to 0x00.
5. The Upper Tester orders the IUT to send a Config Beacon Set message to the Lower Tester with the Beacon field value set to 0x01.
6. The Lower Tester expects to receive a Config Beacon Set message and responds by sending a Config Beacon Status message with the Beacon field value set to 0x01.
Figure 4.74: MESH/CFGCL/CFG/SNBP/BV-01-C

- Expected Outcome
  
  Pass verdict

  In step 2, the IUT sends a properly formatted Config Beacon Get message.

  In step 4, the IUT sends a properly formatted Config Beacon Set message with the Beacon field value set to 0x00.

  In step 6, the IUT sends a properly formatted Config Beacon Set message with the Beacon field value set to 0x01.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.15.2 Composition Data Procedures

This section is for validating the Composition Data procedures for both the Configuration Server Model and the Configuration Client Model.
4.15.2.1 MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages]

• Test Purpose
  Verify that an IUT node supporting the Configuration Server Model can respond to Config Composition Data Get messages and that the IUT supports the mandatory models.

• Reference
  [3] Section 4.2.1, 4.2.1.1, 4.3.2.4, 4.3.2.5, 4.4.1.2, 4.4.1.2.2, 4.4.2.2.2

• Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT and the number of pages of the IUT’s Composition Data defined in the IXIT [6] parameters.

• Test Procedure
  1. The Lower Tester sends a Config Composition Data Get message to the IUT with the Page field value set to 0xFF.
  2. The Lower Tester expects the IUT to respond with a Config Composition Data Status message with the Page field set to the maximum page number defined in IXIT and the Data field set to the information contained by that Composition Data Page.
  3. While the page number received in step 2 is not 0x00, repeat steps 1-2, setting the Page field in step 1 to the most recently received page number minus one.
  4. The Lower Tester parses the Composition Data returned by the IUT in all iterations through step 2 and checks that all fields are valid and consistent and that the mandatory root models are supported.
• **Expected Outcome**

  **Pass verdict**

  In all iterations through step 2, the IUT sends complete and correctly formatted Composition Data. The Configuration Model and the Health Model are supported on the primary Element. All fields from the Composition Data information contain valid values.

  All configuration messages are encrypted using the device key of the IUT.

4.15.2.2 MESH/CFGCL/CFG/COMP/BV-01-C [Send Composition Data Get Messages]

• **Test Purpose**

  Verify that an IUT node supporting the Configuration Client Model can send Config Composition Data Get messages to a Configuration Server.

• **Reference**

  [3] Section 4.3.2.4, 4.4.2.2.2

• **Initial Condition**

  - The IUT supports the Configuration Client Model.
- The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

**Test Procedure**

1. The Upper Tester orders the IUT to obtain the device composition data of the Lower Tester.
2. The Lower Tester expects to receive a Config Composition Data Get message from the IUT and responds with a Config Composition Data Status message with the Page field value set to 0x00.

![Diagram](image)

*Figure 4.76: MESH/CFGCL/CFG/COMP/BV-01-C*

**Expected Outcome**

*Pass verdict*

In step 2, the IUT sends a properly formatted Config Composition Data Get message to the Lower Tester.

All configuration messages are encrypted using the device key of the Lower Tester.

### 4.15.3 Default TTL Procedure

This section is for validating the Default TTL procedures for both the Configuration Server Model and the Configuration Client Model.

#### 4.15.3.1 MESH/NODE/CFG/DTTL/BV-01-C [Respond to Config Default TTL Set/Get Messages]

**Test Purpose**

Verify that an IUT node supporting the Configuration Server Model can respond to Config Default TTL Set and Config Default TTL Get messages and the configuration messages are encrypted using the device key used in the received messages.

**Reference**

[3] Section 3.7.4.3, 4.3.2.6, 4.3.2.7, 4.3.2.8, 4.4.1.2, 4.4.1.2.3

**Initial Condition**

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports the Configuration Client Model.
The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Lower Tester sends a Config Default TTL Set message to the IUT with the TTL field set to a value of 0x00.
  2. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a TTL field value of 0x00.
  3. The Lower Tester sends a Config Default TTL Get message to the IUT.
  4. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a TTL field value of 0x00.
  5. The Lower Tester sends a Config Default TTL Set message to the IUT with the TTL field set to a value of 0x7F.
  6. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a TTL field value of 0x7F.
  7. The Lower Tester sends a Config Default TTL Get message to the IUT.
  8. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a TTL field value of 0x7F.

*Figure 4.77: MESH/NODE/CFG/DTTL/BV-01-C*
• **Expected Outcome**

  **Pass verdict**

  In steps 2 and 4, the IUT responds with a Config Default TTL Status message with the TTL field value equal to 0x00.

  In steps 6 and 8, the IUT responds with a Config Default TTL Status message with the TTL field value equal to 0x7F.

  All configuration messages are encrypted using the device key of the IUT.

4.15.3.2 MESH/NODE/CFG/DTTL/BI-01-C [Respond to Config Default TTL Set Message (On Secondary Element or With Invalid TTL)]

• **Test Purpose**

  Verify that an IUT node supporting the Configuration Server Model ignores a Config Default TTL Set message with invalid fields or if a Config Default TTL Get message is sent to the secondary Element of the IUT.

• **Reference**

  [3] Section 3.7.4.3, 4.3.2.6, 4.3.2.7, 4.3.2.8, 4.4.1.1, 4.4.1.2.3

• **Initial Condition**

  - The IUT supports the Configuration Server Model on the primary Element.
  - The IUT supports a secondary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key defined in the IXIT [6] parameters.

• **Test Procedure**

  1. The Lower Tester sends a Config Default TTL Get message to the IUT.
  2. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a valid TTL value.
  3. The Lower Tester sends a Config Default TTL Set message to the IUT with the TTL field set to a value of 0x80.
  4. The Lower Tester expects the IUT to ignore the message from step 3 and not respond with a Config Default TTL Status message.
  5. The Lower Tester sends a Config Default TTL Get message to the secondary Element of the IUT.
  6. The Lower Tester expects the IUT to ignore the message from step 5 and not respond with a Config Default TTL Status message.
  7. The Lower Tester repeats steps 3-4, setting the TTL field to 0x01 in step 3.
  8. The Lower Tester repeats steps 1-2 to confirm that the value of the Default TTL state has not changed on the IUT.
Figure 4.78: MESH/NODE/CFG/DTTL/Bi-01-C

- **Expected Outcome**

  **Pass verdict**

  In both iterations through step 2, the IUT responds with a Config Default TTL Status message containing a valid value in the TTL field.

  In both iterations through step 4 and in step 6, the IUT does not respond to the Config Default TTL Set message with the invalid TTL field value.

  In step 7, the IUT resends the same message from step 2.

  All configuration messages are encrypted using the device key of the IUT.
4.15.3.3 MESH/CFGCL/CFG/DTTL/BV-01-C [Send Config Default TTL Set/Get Messages]

- **Test Purpose**
  
  Verify that an IUT node supporting the Configuration Client Model can send Config Default TTL Get messages to a Configuration Server, and the configuration messages are encrypted using destination device’s device key.

- **Reference**
  
  [3] Section 3.7.4.3, 4.3.2.6, 4.3.2.7, 4.3.2.8, 4.4.1.2.3, 4.4.2.2.3

- **Initial Condition**
  
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- **Test Procedure**

  1. The Upper Tester orders the IUT to send a Config Default TTL Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config Default TTL Get message.
  3. The Lower Tester responds with a Config Default TTL Status message with a valid TTL field value.
  4. The Upper Tester orders the IUT to send a Config Default TTL Set message to the Lower Tester with a valid TTL field value.
  5. The Lower Tester expects to receive a Config Default TTL Set message with the requested TTL field value.
  6. The Lower Tester responds with a Config Default TTL Status message with the TTL field value equal to the value from step 4.

![Diagram](image_url)
• Expected Outcome

   Pass verdict

   In step 2, the IUT sends a Config Default TTL Get message to the Lower Tester.

   In step 5, the IUT sends a Config Default TTL Set message to the Lower Tester, with the TTL field value set to the value requested in step 4.

   All configuration messages are encrypted using the device key of the Lower Tester.

4.15.4 GATT Proxy Procedures

This section is for validating the GATT Proxy procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.4.1 MESH/NODE/CFG/GPXY/BV-01-C [Respond to Config GATT Proxy Set/Get Messages]

• Test Purpose

   Verify that an IUT node supporting the Configuration Server Model can respond to Config GATT Proxy Set and Get messages.

• Reference

   [3] Section 4.2.1.1, 4.2.11, 4.3.2.10, 4.3.2.11, 4.4.1.2, 4.4.1.2.4, 4.4.2.2.4, 7.2.2.2

• Initial Condition

   - The IUT supports the Configuration Server Model on the Primary Element.
   - The Lower Tester supports the Configuration Client Model.
   - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

• Test Procedure

   1. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x00 (“The Mesh Proxy Service is stopped”).
   2. If the IUT supports the Mesh Proxy Service, the Lower Tester expects to receive a Config GATT Proxy Status message in response from the IUT with a GATTProxy field value of 0x00 (“The Mesh Proxy Service is stopped”). Otherwise, the Lower Tester expects a GATTProxy field value of 0x02 (“The Mesh Proxy Service is not supported”).
   3. The Lower Tester sends a Config GATT Proxy Get message to the IUT.
   4. If the IUT supports the Mesh Proxy Service, the Lower Tester expects to receive a Config GATT Proxy Status message in response from the IUT with a GATTProxy field value of 0x00 (“The Mesh Proxy Service is stopped”). Otherwise, the Lower Tester expects a GATTProxy field value of 0x02 (“The Mesh Proxy Service is not supported”).
   5. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01 (“The Mesh Proxy Service is running”).
   6. If the IUT supports the Mesh Proxy Service, the Lower Tester expects to receive a Config GATT Proxy Status message in response from the IUT with a GATTProxy field value of 0x01 (“The
Mesh Proxy Service is running”). Otherwise, the Lower Tester expects a GATTProxy field value of 0x02 (“The Mesh Proxy Service is not supported”).

**Figure 4.80: MESH/NODE/CFG/GPXY/BV-01-C**

- **Expected Outcome**
  
  **Pass verdict**

  In steps 2 and 4, the IUT responds with a Config GATT Proxy Status message with the GATTProxy field value set to 0x00, if the IUT supports the Mesh Proxy Service, or to 0x02, if the IUT does not support the Mesh Proxy Service.
In step 6, the IUT responds with a Config GATT Proxy Status message with the GATTProxy field value set to 0x01, if the IUT supports the Mesh Proxy Service, or to 0x02, if the IUT does not support the Mesh Proxy Service.

All configuration messages are encrypted using the device key of the IUT.

4.15.4.2 MESH/NODE/CFG/GPXY/BV-02-C [Network ID Advertising on Multiple Subnets]

- Test Purpose
  Verify that a Configuration Server IUT can trigger Network ID advertising based on the GATT Proxy state change and can interleave advertising for multiple Subnets.

- Reference
  [3] Section 4.2.12, 4.3.2.9, 4.3.2.11, 4.4.1.2, 4.4.1.2.12

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT supports the Mesh Proxy Service.
  - The GATT Proxy state is set to 0x00.

- Test Procedure
  1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT.
  2. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01.
  3. The Lower Tester expects the IUT to respond with a Config GATT Proxy Status message and to begin sending interleaved Network ID advertisings for both Subnets it supports.
Figure 4.81: MESH/NODE/CFG/GPXY/BV-02-C

- **Expected Outcome**

  **Pass verdict**

  In step 3, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x01. The IUT then begins advertising with Network ID with interleaved packets for both Subnets it supports.

4.15.4.3 MESH/NODE/CFG/GPXY/BI-01-C [Respond to Config GATT Proxy Set Messages Invalid Parameters]

- **Test Purpose**

  Verify that an IUT node supporting the Configuration Server Model ignores Config GATT Proxy Set messages with invalid field values.

- **Reference**

  [3] Section 3.1.2, 4.2.1.1, 4.3.2.10, 4.3.2.11, 4.4.1.2, 4.4.1.2.4, 4.4.2.2.4, 7.2.2.2

- **Initial Condition**

  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- **Test Procedure**

  1. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to an RFU value (in the range 0x03-0xFF).
2. The Lower Tester expects the IUT to ignore the message and not send any response.

![Diagram of message flow]

Figure 4.82: MESH/NODE/CFG/GPXY/BI-01-C

- Expected Outcome
  
  **Pass verdict**

  In step 2, the IUT does not send any message in response to the Config GATT Proxy Set message with invalid parameters sent by the Lower Tester in step 1.

  All configuration messages are encrypted using the device key of the IUT.

**4.15.4.4 MESH/CFGCL/CFG/GPXY/BV-01-C [Send Config GATT Proxy Set/Get Messages]**

- Test Purpose
  
  Verify that an IUT node supporting the Configuration Client Model can send Config GATT Proxy Get and Config GATT Proxy Set messages to a Configuration Server.

- Reference
  
  [3] Section 4.2.1.1, 4.3.2.10, 4.3.2.11, 4.4.1.2, 4.4.1.2.4, 4.4.2.2.4, 7.2.2.2

- Initial Condition
  
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure
  
  1. The Upper Tester orders the IUT to send a Config GATT Proxy Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config GATT Proxy Get message.
  3. The Lower Tester sends a Config GATT Proxy Status message in response to the IUT with a valid GATTProxy field value.
  4. The Upper Tester orders the IUT to send a Config GATT Proxy Set message to the Lower Tester with a GATTProxy field value of either 0x00 or 0x01.
  5. The Lower Tester expects to receive a Config GATT Proxy Set message with the GATTProxy field value set to the value requested in step 4.
6. The Lower Tester sends a Config GATT Proxy Status message in response to the IUT with the GATTProxy field value matching the value from the request.

![Diagram of Mesh Profile (MESH) Test Suite](image)

**Figure 4.83: MESH/CFGCL/CFG/GPXY/BV-01-C**

- **Expected Outcome**
  
  **Pass verdict**

  In step 2, the IUT sends a Config GATT Proxy Get message to the Lower Tester.

  In step 5, the IUT sends a Config GATT Proxy Set message to the Lower Tester with the GATTProxy field value set to the value requested in step 4.

  All configuration messages are encrypted using the device key of the Lower Tester.

### 4.15.5 Friend Procedures

This section is for validating the Friend procedures for both the Configuration Server Model and the Configuration Client Model.

#### 4.15.5.1 MESH/NODE/CFG/CFGF/BV-01-C [Respond to Config Friend Get/Set Messages]

- **Test Purpose**

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Friend Get and Config Friend Set messages.

- **Reference**

  [3] Section 3.7.4.3, 4.2.1, 4.2.13, 4.3.2.55, 4.3.2.56, 4.3.2.57, 4.4.1.2, 4.4.1.2.5

- **Initial Condition**
  
  - The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.
- IUT support for the Friend feature is indicated in the ICS table 7 (Node Capabilities – Upper Transport Layer). If the Friend feature is supported, the feature is disabled.

- **Test Procedure**
  1. The Lower Tester sends a Config Friend Get message to the IUT.
  2. The Lower Tester expects to receive a Config Friend Status message in response from the IUT with the Friend field set to 0x00, if the Friend feature is supported, or to 0x02, if the Friend feature is not supported.
  3. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x01.
  4. The Lower Tester expects to receive a Config Friend Status message in response from the IUT with the Friend field set to 0x01, if the Friend feature is supported, or to 0x02, if the Friend feature is not supported.
  5. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x00.
  6. The Lower Tester expects to receive a Config Friend Status message in response from the IUT with the Friend field set to 0x00, if the Friend feature is supported, or to 0x02, if the Friend feature is not supported.
**Expected Outcome**

**Pass verdict**

In steps 2 and 6, the IUT responds with a Config Friend Status message with the Friend field value set to 0x00, if the IUT supports the Friend feature, or to 0x02, if the IUT does not support the Friend feature.
In step 4, the IUT responds with a Config Friend Status message with the Friend field value set to 0x01, if the IUT supports the Friend feature, or to 0x02, if the IUT does not support the Friend feature.

All configuration messages are encrypted using the device key of the IUT.

**4.15.5.2 MESH/NODE/CFG/CGFG/BI-01-C [Respond to Config Friend Set Messages Invalid Parameters]**

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Server Model ignores Config Friend Set messages with invalid field values.

- **Reference**
  [3] Section 3.1.2, 4.2.1.1, 4.3.2.56, 4.4.1.2.4, 4.4.2.2.4, 7.2.2.2

- **Initial Condition**
  - The IUT supports the Configuration Server Model.
  - The IUT supports the Friend feature, and the feature is enabled.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to an RFU value (in the range 0x02-0xFF).
  2. The Lower Tester expects the IUT to ignore the message and not send any response.

- **Expected Outcome**
  **Pass verdict**
  In step 2, the IUT does not send any message in response to the Config Friend Set message with invalid parameters sent by the Lower Tester in step 1.

  All configuration messages are encrypted using the device key of the IUT.
4.15.5.3 MESH/CFGCL/CFG/CFGF/BV-01-C [Send Config Friend Get/Set Messages]

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Client Model can send Config Friend Get and Config Friend Set messages to a Configuration Server.

- **Reference**
  [3] Section 4.2.1, 4.3.2.55, 4.3.2.56, 4.3.2.57, 4.4.1.2.5, 4.4.2.2.5

- **Initial Condition**
  - The IUT supports the Configuration Client Model on the Primary Element.
  - The Lower Tester supports the Configuration Server Model.
  - The Lower Tester supports the Friend feature and the feature is enabled.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Upper Tester orders the IUT to send a Config Friend Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config Friend Get message.
  3. The Lower Tester sends a Config Friend Status message in response to the IUT with a valid Friend field value.
  4. The Upper Tester orders the IUT to send a Config Friend Set message to the Lower Tester with a Friend field value of either 0x00 or 0x01.
  5. The Lower Tester expects to receive a Config Friend Set message with the Friend field value set to the value requested in step 4.
  6. The Lower Tester sends a Config Friend Status message in response to the IUT with the Friend field value matching the value from the request or 0x02 if the Friend feature is not supported.

---

![Diagram](Figure 4.86: MESH/CFGCL/CFG/CFGF/BV-01-C)
• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Friend Get message to the Lower Tester.

In step 5, the IUT sends a Config Friend Set message to the Lower Tester with the Friend field value set to the value requested in step 4.

All configuration messages are encrypted using the device key of the Lower Tester.

4.15.6 Relay Procedures

This section is for validating the Relay procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.6.1 MESH/NODE/CFG/CFGR/BV-01-C [Respond to Config Relay Get/Set Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to Config Relay Get and Config Relay Set messages.

• Reference

[3] Section 3.7.4.3, 4.3.2.12, 4.3.2.13, 4.3.2.14, 4.4.1.2, 4.4.1.2.6, 4.4.2.2.6

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.
- IUT support for the Relay feature is indicated in the IXIT [6] parameters. If the Relay feature is supported, the feature is disabled.

• Test Procedure

1. The Lower Tester sends a Config Relay Get message to the IUT.
2. The Lower Tester expects to receive a Config Relay Status message in response from the IUT with the Relay field set to 0x00, if the Relay feature is supported, or to 0x02, if the Relay feature is not supported, and the RelayRetransmitCount field value set to 0b000, and the RelayRetransmitIntervalSteps field value set to 0b00000 in both cases.
3. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01, the RelayRetransmitCount field set to 0b001, and the RelayRetransmitIntervalSteps field set to 0b00001.
4. The Lower Tester expects to receive a Config Relay Status message in response from the IUT with the Relay field set to 0x01, the RelayRetransmitCount field value set to 0b001, and the RelayRetransmitIntervalSteps field value set to 0b00001, if the Relay feature is supported, or to 0x02, the RelayRetransmitCount field value set to 0b000, and the RelayRetransmitIntervalSteps field value set to 0b00000, if the Relay feature is not supported.
5. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x00, the RelayRetransmitCount field set to 0b000, and the RelayRetransmitIntervalSteps field set to 0b00000.

6. The Lower Tester expects to receive a Config Relay Status message in response from the IUT with the Relay field set to 0x00, if the Relay feature is supported, or to 0x02, if the Relay feature is not supported, and the RelayRetransmitCount field value set to 0b000, and the Publish RelayRetransmitIntervalSteps field value set to 0b00000 in both cases.

Figure 4.87: MESH/NODE/CFG/CFGR/BV-01-C
• Expected Outcome

Pass verdict

In steps 2 and 6, the IUT responds with a Config Relay Status message with the Relay field value set to 0x00, if the IUT supports the Relay feature, or to 0x02, if the IUT does not support the Relay feature, and the RelayRetransmitCount field value set to 0b00, and the RelayRetransmitIntervalSteps field value set to 0b00000 in both cases.

In step 4, the IUT responds with a Config Relay Status message with the Relay field value set to 0x01, the RelayRetransmitCount field value set to 0b001, and the RelayRetransmitIntervalSteps field value set to 0b00001, if the IUT supports the Relay feature, or to 0x02, the RelayRetransmitCount field value set to 0b0000, and the RelayRetransmitIntervalSteps field value set to 0b00000, if the IUT does not support the Relay feature.

All configuration messages are encrypted using the device key of the IUT.

4.15.6.2 MESH/NODE/CFG/CFGR/BI-01-C [Respond to Config Relay Set Messages Invalid Parameters]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model ignores Config Relay Set messages with invalid field values.

• Reference

[3] Section 3.1.2, 4.3.2.13, 4.3.2.14, 4.4.1.2.6, 4.4.2.2.6

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

• Test Procedure

1. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to an RFU value (in the range 0x03-0xFF), the RelayRetransmitCount field value set to 0b000, and the RelayRetransmitIntervalSteps field value set to 0b00000.
2. The Lower Tester expects the IUT to ignore the message and not send any response.

Figure 4.88: MESH/NODE/CFG/CFGR/BI-01-C
• Expected Outcome

Pass verdict

In step 2, the IUT does not send any message in response to the Config Relay Set message with invalid parameters sent by the Lower Tester in step 1.

All configuration messages are encrypted using the device key of the IUT.

4.15.6.3 MESH/CFGCL/CFG/CFGR/BV-01-C [Send Config Relay Get/Set Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config Relay Get and Config Relay Set messages to a Configuration Server.

• Reference

[3] Section 3.7.4.3, 4.2.8, 4.3.2.12, 4.3.2.13, 4.3.2.14, 4.4.1.2.6, 4.4.2.2.6

• Initial Condition

- The IUT supports the Configuration Client Model on the Primary Element.
- The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure

1. The Upper Tester orders the IUT to send a Config Relay Get message to the Lower Tester.
2. The Lower Tester expects to receive a Config Relay Get message.
3. The Lower Tester sends a Config Relay Status message in response to the IUT with a valid Relay field value, a valid RelayRetransmitCount field value, and a valid RelayRetransmitIntervalSteps field value.
4. The Upper Tester orders the IUT to send a Config Relay Set message to the Lower Tester with a Relay field value of either 0x00 or 0x01, the RelayRetransmitCount field set to 0b000, and the RelayRetransmitIntervalSteps field set to 0b00000.
5. The Lower Tester expects to receive a Config Relay Set message with the Relay field value set to the value requested in step 4.
6. The Lower Tester sends a Config Relay Status message in response to the IUT with the Relay field value matching the value from the request.
Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Relay Get message to the Lower Tester.

In step 5, the IUT sends a Config Relay Set message to the Lower Tester with the Relay field value set to the value requested in step 4.

All configuration messages are encrypted using the device key of the Lower Tester.

4.15.7 Model Publication Procedures

This section is for validating the Model Publication procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.7.1 MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages]

Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Publication Set, Config Model Publication Get and Config Model Publication Virtual Address Set messages.

Reference

[3] Section 3.1.2, 4.2.2.3, 4.3.2, 4.3.2.15, 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2, 4.4.1.2.7, 4.4.2.2.7

Initial Condition

- The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- IXIT [6] provides ModelID that supports publication with periodic and state change publishing.
• Test Procedure

1. The Lower Tester sends an Config AppKey Add message to the IUT with a NetKeyIndex value of 0, an AppKeyIndex of 0, (so the NetKeyIndex and AppKeyIndex field is set to 0x000000) and a valid 16-octet AppKey value.

2. The Lower Tester expects to receive an Config AppKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex and AppKeyIndex field set to value used in step 1.

3. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to zero (0).

4. The Lower Tester expects to receive a Config Model App Status message from the IUT with the Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex set to 0 (zero), and the Model Identifier as referenced in step 3. The Lower Tester stores the information for later use.

5. The Lower Tester sends a Config Model Publication Set message to the IUT with the ElementAddress, Model Identifier, and AppKeyIndex fields set to the values stored from step 4, the PublishAddress field set to the address of the Lower Tester, the CredentialFlag field set to 0b0, the RFU field set to 0b111, the Publish TTL field set to 0xFF (use device default TTL), and the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to valid values.

6. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field value equal to 0x00, the ElementAddress field value equal to the IUT’s unicast address, the PublishAddress field value equal to the address of the Lower Tester, the AppKeyIndex field value equal to the value stored in step 4, the CredentialFlag field value equal to 0b0, the RFU field value set to 0b111, the Publish TTL field value set to 0xFF, and the Publish Period, Publish Retransmit Count, Publish Retransmit Interval Steps and Model Identifier fields set to the values received in step 5.

7. The Lower Tester sends a Config Model Publication Get message with the ElementAddress field value equal to that sent in step 5 and a Model Identifier field value equal to that used to set the Publication in step 5.

8. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with all the fields the same as that in step 6.

9. The Lower Tester sends a Config Model Publication Set message to the IUT with the ElementAddress, Model Identifier, and AppKeyIndex fields set to values stored from step 4, a PublishAddress equal to the address of the Lower Tester, CredentialFlag field set to 0b0, RFU field set to 0b111, Publish TTL field set to a random number between 0x00 and 0x7F, and the Publish Period, Publish Retransmit Count, Publish Retransmit Interval Steps and Model Identifier fields set to the values received in step 9.

10. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00, the ElementAddress field equal to the IUT’s unicast address, PublishAddress field value set to Publish Address for the Model, an AppKeyIndex field value equal to the value stored in step 4, CredentialFlag field value set to 0b0, RFU field value set to 0b000, Publish TTL field value equal to 0xFF, and the Publish Period, Publish Retransmit Count, Publish Retransmit Interval Steps and Model Identifier fields set to the values received in step 9.

11. The Lower Tester sends a Config Model Publication Virtual Address Set message to the IUT with the Element Address equal to that in step 4, the PublishAddress set to a full 128-bit UUID, the AppKeyIndex field set to 0, the RFU field set to 0b000, the Publish TTL value set to 0x3F, the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to valid values and the Model Identifier value as in step 4.
12. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00, the RFU field value set to 0b000 and all other field values equal to those sent in step 11.
13. Repeat step 9, but with PublishAddress set to the unassigned address 0x0000.
14. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00.
15. The Lower Tester sends a Config Model Publication Get message to the IUT with the ElementAddress field value equal to that sent in step 5 and a Model Identifier field value equal to that used to set the Publication in step 5.
16. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00, ElementAddress and ModelIdentifier fields are set to the values of the incoming message, and all other fields are set to 0x00.
Figure 4.90: MESH/NODE/CFG/MP/BV-01-C
• Expected Outcome

Pass verdict

In step 6 and step 8, the IUT sends the Config Model Publication Status message with the Status field set to 0x00, the ElementAddress field set to IUT’s unicast address, PublishAddress field set to Publish Address for the Model, AppKeyIndex field value equal to the value stored in step 4, CredentialFlag field set to 0b0, RFU field set to 0b000, Publish TTL field set to 0xFF, the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to the ones requested in step 5, and the Model Identifier fields set to the Model Identifier as in step 4.

In step 10, the IUT sends the Config Model Publication Status message with the Status field set to 0x00, the ElementAddress field set to IUT’s unicast address, PublishAddress field set to Publish Address for the Model, an AppKeyIndex field value equal to the value stored in step 4, CredentialFlag field set to 0b0, RFU field set to 0b000, Publish TTL field set to the value as in step 9, the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to the ones requested in step 9, and the Model Identifier field set to the Model Identifier as in step 9.

In step 12, the IUT sends a Config Model Publication Status message with the Status field set to 0x00, the RFU field value set to 0b000, and all other field values matching the values sent in step 11.

In steps 14 and 16, the IUT sends a Config Model Publication Status message with the Status field set to 0x00.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.7.2 MESH/NODE/CFG/MP/BV-02-C [Disabling Publishing – AppKey/Config NetKey Deleted]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model disables the publishing of the Status message when the AppKey or NetKey is deleted.

• Reference

[3] Section 3.7.4.3, 3.7.6.1.2, 3.7.6.1.3, 4.4.1.2.7, 4.4.1.2.9, 4.4.1.2.10, 4.4.2.2.7

• Initial Condition

- The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.

- The Lower Tester supports the Configuration Client Model.

- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- IXIT [6] provides ModelID that supports publication with periodic and state change publishing.

- The IUT has at least two network keys.

• Test Procedure

1. The Lower Tester runs step 1 through step 4 of test case MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set
2. The Lower Tester sends a Config Model Publication Set message to the IUT with the ElementAddress, Model Identifier, and AppKeyIndex fields set to the values stored from step 1, the PublishAddress field set to the address of the Lower Tester, CredentialFlag field set to 0b0, RFU field set to 0b000, the Publish TTL field set to 0xFF (use device default TTL), Publish Period field set to 0x05, the Publish Retransmit Count field set to 0b000, and the Publish Retransmit Interval Steps field set to 0b00000.

3. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00 and the other fields set to the values received in step 2.

4. Follow the instructions in test MESH/NODE/CFG/AKL/BV-03-C [Respond to Config AppKey Delete Message] to delete the AppKey of the Model.

5. The Lower Tester sends a Config Model Publication Get message with the ElementAddress field value equal to that sent in step 2 and a Model Identifier field value equal to that used to set the Publication in step 2.

6. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00 and the PublishAddress field set to the unassigned address.

7. Repeat steps 1–6, but in step 4, follow the instructions in another test MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message] to delete the NetKey that the AppKey is bound to (from step 1).

8. Repeat steps 1–6, but in step 4, execute the procedure described in MESH/NODE/CFG/MAKL/BV-02-C [Respond to Config Model App Unbind and SIG/Config Vendor Model App Get Messages] to unbind the AppKey from the Model.
Expected Outcome

Pass verdict

In step 3, the IUT sends a Config Model Publication Status message with the Status field set to 0x00 and the other fields set to the values received in step 2.
In step 6, the IUT sends a Config Model Publication Status message with the Status field set to 0x00 and the PublishAddress field set the unassigned address.

All configuration messages are encrypted using the device key of the IUT.

4.15.7.3 MESH/NODE/CFG/MP/BI-01-C [Respond to Config Model Publication Get/Set/Virtual Address Set Messages (Invalid ElementAddress)]

- Test Purpose
  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Publication Get, Set, and Virtual Address Set messages with an invalid ElementAddress field.

- Reference
  [3] Section 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2.7, 4.4.2.2.7

- Initial Condition
  - The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Element list of the IUT defined in the IXIT [6] parameters.

- Test Procedure
  1. The Lower Tester sends a Config Model Publication Set message to the IUT with an ElementAddress value within the range of the field but not one of the Elements listed in the IUT’s IXIT [6] table and all other fields set to valid values.
  2. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e. the identically named fields) of the incoming message, and all other fields set to 0.
  3. The Lower Tester sends a Config Model Publication Virtual Address Set message to the IUT with an ElementAddress value within the range of the field but not one of the Elements listed in the IUT’s IXIT [6] table.
  4. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e. the identically named fields) of the incoming message, and all other fields set to 0.
  5. The Lower Tester sends a Config Model Publication Get message to the IUT with an ElementAddress value within the range of the field but not one of the Elements listed in the IUT’s IXIT [6] table.
  6. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e. the identically named fields) of the incoming message, and all other fields set to 0.
• **Expected Outcome**

   **Pass verdict**

   In steps 2, 4, and 6, the IUT sends a Config Model Publication Status message with a Status field set to 0x01, the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.

   All configuration messages are encrypted using the device key of the IUT.

---

### 4.15.7.4 MESH/NODE/CFG/MP/BI-02-C [Respond to Config Model Publication Get/Set/Virtual Address Set Message (Invalid Model Identifier)]

• **Test Purpose**

   Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Publication Get/Set/Virtual Address Set messages with an invalid Model Identifier field.

• **Reference**

   [3] Section 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2.7, 4.4.2.2.7

• **Initial Condition**

   - The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
   - The Lower Tester supports the Configuration Client Model.
   - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
   - The Lower Tester has access to the Model list of the IUT defined in the IXIT [6] parameters.
• **Test Procedure**

1. The Lower Tester sends a Config Model Publication Set message to the IUT with a Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT’s IXIT [6] table and all other fields set to valid values.

2. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

3. The Lower Tester sends a Config Model Publication Virtual Address Set message to the IUT with a Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT’s IXIT [6] table.

4. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

5. The Lower Tester sends a Config Model Publication Get message to the IUT with a Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT’s IXIT [6] table.

6. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

![Diagram](image)

*Figure 4.93: MESH/NODE/CFG/MP/BI-02-C*

• **Expected Outcome**

**Pass verdict**

In steps 2, 4, and 6, the IUT sends a Config Model Publication Status message with a Status field set to 0x02, the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.

All configuration messages are encrypted using the device key of the IUT.
4.15.7.5 MESH/NODE/CFG/MP/BI-03-C [Respond to Config Model Publication Set/Virtual Address Set Messages (Invalid AppKeyIndex)]

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Publication Set/Virtual Address Set messages with an invalid AppKeyIndex field.

- **Reference**
  [3] Section 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2.7, 4.4.2.2.7

- **Initial Condition**
  - The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the AppKey list of the IUT defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Lower Tester sends a Config Model Publication Set message to the IUT with an AppKeyIndex value within the range of the field but not one of the AppKeys listed in the IUT’s IXIT [6] table and all other fields set to valid values.
  2. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x03 (Invalid AppKey).
  3. The Lower Tester sends a Config Model Publication Virtual Address Set message to the IUT with an AppKeyIndex value within the range of the field but not one of the AppKeys listed in the IUT’s IXIT [6] table and all other fields set to valid values.
  4. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x03 (Invalid AppKey).

![Diagram](Figure 4.94: MESH/NODE/CFG/MP/BI-03-C)
• Expected Outcome

Pass verdict

In step 2 and step 4, the IUT sends a Config Model Publication Status message with a Status field set to 0x03, the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.

All configuration messages are encrypted using the device key of the IUT.

4.15.7.6 MESH/CFGCL/CFG/MP/BV-01-C [Send Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config Model Publication Get, Config Model Publication Set and Config Model Publication Virtual Address Set messages to a Configuration Server.

• Reference

[3] Section 3.1.2, 3.7.4.3, 4.2.2.1, 4.2.2.3, 4.3.2, 4.3.2.15 – 4.3.2.18, 4.4.1.2.7, 4.4.2.2.7

• Initial Condition

- The IUT supports the Configuration Client Model.
- The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure

1. The Upper Tester orders the IUT to send a Config Model Publication Get message to the Lower Tester with the Lower Tester’s ElementAddress and Model Identifier.
2. The Lower Tester expects to receive a Config Model Publication Get message containing ElementAddress and Model Identifier field values equal to those sent in the Config Model Publication Get message in step 1.
3. The Lower Tester sends a Config Model Publication Status message in response to the IUT with a Status field set to 0x00, the ElementAddress field value as sent in step 2, the AppKeyIndex field set to 0, the Publish TTL value between 0x00—0x3F, the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to valid values, and the expected Model Identifier as sent in step 2.
4. The Upper Tester orders the IUT to send a Config Model Publication Set message to the Lower Tester with the ElementAddress value as in step 1, the PublishAddress set to its own address, the AppKeyIndex field set to 0, the CredentialFlag field set to 0b0, the RFU field set to 0b000, the Publish TTL value set to 0x3F, the Publish Period field set with Step Resolution set to 0b01 and Number of Steps set to 0x0F, the Publish Retransmit Count field set to 0b000, the Publish Retransmit Interval Steps field set to 0b00000 and the Model Identifier value as in step 1.
5. The Lower Tester expects to receive a Config Model Publication Set message containing the values as sent in step 4.
6. The Lower Tester sends a Config Model Publication Status message in response to the IUT with a Status field set to 0x00, and all other field values equal to those sent in step 4.
7. The Upper Tester orders the IUT to send a Config Model Publication Virtual Address Set message to the Lower Tester with the ElementAddress value as in step 1, the PublishAddress set to a full 128-bit UUID, the AppKeyIndex field set to 0, the CredentialFlag field set to 0b0, the RFU field set to 0b000, the Publish TTL value set to 0x3F, the Publish Period field set with Step Resolution set to 0b01 and Number of Steps set to 0x0F, the Publish Retransmit Count field set to 0b000, the Publish Retransmit Interval Steps field set to 0b00000 and the Model Identifier value as in step 1.

8. The Lower Tester sends a Config Model Publication Status message in response to the IUT with a Status field set to 0x00, and all other field values equal to those sent in step 7.

9. Repeat step 4–6 with the PublishAddress field set to a group address.

Figure 4.95: MESH/CFGCL/CFG/MP/BV-01-C

- **Expected Outcome**
  
  **Pass verdict**

  In step 1, the IUT sends a Config Model Publication Get message to the Lower Tester with the Lower Tester’s ElementAddress and Model Identifier.
In step 4, the IUT sends a Config Model Publication Set message to the Lower Tester with the ElementAddress value as in step 1, the PublishAddress set to its own address, the AppKeyIndex field set to 0, the CredentialFlag field set to 0b0, the RFU field set to 0b000, the Publish TTL field value set to 0x3F, the Publish Period field with the Step Resolution field set to 0b01 and the Number of Steps field set to 0x0F, the Publish Retransmit Count field set to 0b000, the Publish Retransmit Interval Steps field set to 0b00000 and the Model Identifier field value as in step 1.

In step 7, the IUT sends a Config Model Publication Virtual Address Set message with the ElementAddress field value as in step 1, the PublishAddress field set to a full 128-bit UUID, the AppKeyIndex field set to 0, the CredentialFlag field set to 0b0, the RFU field set to 0b000, the Publish TTL field value set to 0x3F, the Publish Period field with the Step Resolution field set to 0b01 and the Number of Steps field set to 0x0F, the Publish Retransmit Count field set to 0b000, the Publish Retransmit Interval Steps field set to 0b00000 and the Model Identifier value as in step 1.

In step 9, the IUT sends a Config Model Publication Set message to the Lower Tester with the PublishAddress field set to a group address and all other fields with the same values as in step 4.

All configuration messages are encrypted and authenticated using the device key of the Lower Tester.

### 4.15.8 Subscription List Procedures

This section is for validating the Subscription List procedures for both the Configuration Server Model and the Configuration Client Model.

#### 4.15.8.1 MESH/NODE/CFG/SL/BV-01-C [Respond to Config Model Subscription Messages]

- **Test Purpose**
  
  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Subscription Add/Delete/Delete All/Overwrite messages.

- **Reference**
  
  [3] Section 3.4.2, 3.7.4.3, 4.2.3, 4.3.2.19 - 4.3.2.30, 4.4.1.2, 4.4.1.2.7, 4.4.1.2.8, 4.4.2.2.8

- **Initial Condition**
  
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The IUT has no current subscriptions.

- **Test Procedure**
  
  1. The Lower Tester obtains and stores the ElementAddress, and the Model Identifier of a valid Model on the IUT from IXIT [6].
  2. The Lower Tester sends a Config Model Subscription Add message to the IUT with the ElementAddress and Model Identifier values from step 1 and the Address value of a group address to be added to the Subscription List.
3. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the ElementAddress field, the Address field and the Model Identifier field set to that in step 2.

4. Repeat steps 2–3 to verify the IUT responds properly when trying to add the same address twice.

5. If the Model ID is a 2-octet SIG Model ID, then the Lower Tester sends a Config SIG Model Subscription Get with the ElementAddress field the unicast address of the Element and the Model Identifier field the SIG Model ID. If the Model ID is a 4-octet Vendor Model ID, then the Lower Tester sends a Config Vendor Model Subscription Get with the ElementAddress field the unicast address of the Element and the Model Identifier field the Vendor Model ID.

6. The Lower Tester expects to receive a Config SIG Model Subscription List message (if the message sent in step 5 is Config SIG Model Subscription Get) or a Config Vendor Model Subscription List message (If the message sent in step 5 is a Config Vendor Model Subscription Get) in response.

7. The Lower Tester sends a Config Model Subscription Delete message to the IUT with the ElementAddress, the Model Identifier, and the Address as in step 2.

8. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as in step 7.

9. Repeat step 5 (the Lower Tester sending a Config SIG Model Subscription Get or a Config Vendor Model Subscription Get) to confirm the record is deleted.

10. The Lower Tester expects to receive a Config SIG Model Subscription List message in response from the IUT with a Status Field set to 0x00 and confirm the record is deleted.

11. Repeat steps 7–8 to verify that the IUT properly responds when trying to delete the same address twice.

12. Repeat steps 2–3 to add another address to the Subscription List.

13. The Lower Tester sends a Config Model Subscription Overwrite message to the IUT with the ElementAddress and the Model Identifier as in step 2, but a new Address (referred below as new 3) not in step 12.

14. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as in step 13.

15. Repeat step 5 (the Lower Tester sending a Config SIG Model Subscription Get or a Config Vendor Model Subscription Get) to confirm that the identified Subscription List is cleared, and the address new 3 in step 13 is added.

16. Repeat step 2 to add a new address (say, new 4) not the same as that in step 13 to the Subscription List. (Now there are two addresses in the Subscription List.)

17. The Lower Tester sends a Config Model Subscription Delete All message to the IUT with the ElementAddress and Model Identifier field values as in step 16.

18. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with the Status Field set to 0x00, the ElementAddress field and the Model Identifier field set to the incoming message, and the Address field set to unassigned address value 0x0000.

19. Repeat step 5 to confirm that the identified Subscription List is cleared.

20. The Lower Tester expects to receive a Config SIG Model Subscription List message (if request is Config SIG Model Subscription Get) or a Config Vendor Model Subscription List (if request is Config Vendor Model Subscription Get) in response from the IUT with the Status Field set to 0x00, the ElementAddress field and the Model Identifier field set to the incoming message, and an empty Addresses field.
Figure 4.96: MESH/NODE/CFG/SL/BV-01-C
• Expected Outcome

Pass verdict

In each iteration through step 3, the IUT sends a Config Model Subscription Status message with a Status field value of 0x00, the ElementAddress field, and the Address field set to the values in step 2.

In step 5, the IUT sends a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.

In each iteration through step 8, the IUT sends a Config Model Subscription Status message with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as defined by the incoming message in step 7.

In step 10, IUT sends a Config Model Subscription Status message with the Status Field set to 0x00.

In step 14, the IUT sends a Config Model Subscription Status message with a Status field value of 0x00, the Address, the ElementAddress, and the Model ID fields as defined by the incoming message in step 13.

In step 18, the IUT sends a Config Model Subscription Status message with the Status Field set to 0x00, the ElementAddress field and the Model ID fields set to the incoming message in step 17, and the Address field set to the unassigned address 0x0000.

In step 20, the IUT sends a Config SIG Model Subscription List or a Config Vendor Model Subscription List with an empty Addresses field.

All configuration messages are encrypted using the device key of the IUT.

4.15.8.2 MESH/NODE/CFG/SL/BV-02-C [Respond to Config Model Subscription Virtual Address Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Subscription Virtual Address Add/Delete/Overwrite messages.

• Reference

[3] Section 4.2.3, 4.3.2.19 - 4.3.2.30, 4.3.6.26, 4.4.1.2.7, 4.4.1.2.8, 4.4.2.2.8

• Initial Condition

- The IUT supports the Configuration Server Model on the primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The IUT has no current subscriptions.
Test Procedure

1. The Lower Tester obtains and stores the ElementAddress, and the Model Identifier of a valid Model on the IUT from IXIT [6].
2. The Lower Tester sends a Config Model Subscription Virtual Address Add message to the IUT with the ElementAddress and Model Identifier values from step 1 and the Address the Label UUID used as the new Address to be added to the Subscription List.
3. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the ElementAddress field, the Address field with the virtual address, and the Model Identifier field value set to that in step 2.
4. If the Model Identifier is a 2-octet SIG Model ID, then the Lower Tester sends a Config SIG Model Subscription Get with the ElementAddress field the unicast address of the Element and the Model Identifier field the SIG Model ID. If the Model Identifier is a 4-octet Vendor Model ID, then the Lower Tester sends a Config Vendor Model Subscription Get with the ElementAddress field set to the unicast address of the Element and the Model Identifier field set to the Vendor Model ID.
5. The Lower Tester expects to receive a Config SIG Model Subscription List message (if the message sent in step 4 is Config SIG Model Subscription Get) or a Config Vendor Model Subscription List message (If the message sent in step 4 is a Config Vendor Model Subscription Get) in response.
6. The Lower Tester sends a Config Model Subscription Virtual Address Delete message to the IUT with the ElementAddress, the Model Identifier, and the Address as in step 2.
7. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as defined by the incoming message in step 6.
8. Repeat steps 4 and 5 (the Lower Tester sending a Config SIG Model Subscription Get or a Config Vendor Model Subscription Get) to confirm the record is deleted.
9. Repeat step 2 to add a new Label UUID address (say, new 2) to the Subscription List.
10. The Lower Tester sends a Config Model Subscription Virtual Address Overwrite message to the IUT with the ElementAddress, and the Model Identifier as in step 2, but the value of a new Label UUID not in step 9.
11. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, ElementAddress, and Model Identifier fields as defined by the incoming message in step 10. The Address field is set to the virtual address which references the Label UUID used in step 10.
12. Repeat step 4 (the Lower Tester sending a Config SIG Model Subscription Get or a Config Vendor Model Subscription Get) to confirm that the identified Subscription List has one entry equal to Label UUID from step 10.
Figure 4.97: MESH/NODE/CFG/SL/BV-02-C
Mesh Profile (MESH) / Test Suite

- **Expected Outcome**

  **Pass verdict**

  In step 3 the IUT sends a Config Model Subscription Status message with the Status field value of 0x00, the ElementAddress field and the Address field set to the values in step 2.

  In step 4 the IUT sends a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.

  In step 5, the IUT sends a Config SIG Model Subscription List message (if the message sent in step 4 is Config SIG Model Subscription Get) or a Config Vendor Model Subscription List message (If the message sent in step 4 is a Config Vendor Model Subscription Get) in response with the current values of the identified Subscription List state (added in step 2), setting the ElementAddress and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00. The Addresses field is set to the value of the virtual address which references the Label UUID used in step 2.

  In step 7, the IUT sends a Config Model Subscription Status message with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as defined by the incoming message in step 6.

  In step 9, IUT sends a Config Model Subscription Status message with a Status Field set to 0x00.

  In step 11, the IUT sends a Config Model Subscription Status message with a Status field value of 0x00, the Address, the ElementAddress, and the Model Identifier fields as defined by the incoming message in step 11.

  All configuration messages are encrypted using the device key of the IUT.

4.15.8.3 MESH/NODE/CFG/SL/BV-03-C [Respond to Config Vendor Model Subscription Get Message]

- **Test Purpose**

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Vendor Model Subscription Get messages.

- **Reference**

  [3] Section 3.7.4.3, 4.2.3, 4.3.2.19, 4.3.2.23, 4.3.2.26, 4.3.2.29, 4.3.2.30, 4.4.1.2.8, 4.4.2.2.8

- **Initial Condition**

  - The IUT supports the Configuration Server Model.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester knows the ElementAddress and Model Identifier of a Vendor Model stored on the IUT.
- The IUT has one or more Vendor Models stored on the device, the 32-bit Vendor Model IDs specified in the IXIT [6], which supports subscription mechanism.

• Test Procedure

1. The Lower Tester sends a Config Model Subscription Overwrite message to the IUT with the ElementAddress field and the Model Identifier field set to a valid value, and the Address field set to a valid group address.
2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as in step 1.
3. The Lower Tester sends a Config Model Subscription Add message to the IUT with the same ElementAddress and Model Identifier values as in step 1 and the Address field set to another group address different from the one in used step 1.
4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the ElementAddress field, the Address field, and the Model Identifier field set to that in step 3.
5. The Lower Tester sends a Config Vendor Model Subscription Get message to the IUT with the ElementAddress and Model Identifier of the Vendor Model identified in the IXIT [6].
6. The Lower Tester expects to receive a Config Vendor Model Subscription List message in response from the IUT with the Status field set to 0x00, the ElementAddress and Model Identifier field values equal to the values sent in step 1, and the addresses of the two group addresses added in step 1 and step 3.

```
<table>
<thead>
<tr>
<th>Lower Tester</th>
<th>IUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config Model Subscription Overwrite</td>
<td></td>
</tr>
<tr>
<td>Config Model Subscription Status (Status=0x00)</td>
<td>Config Model Subscription Status (Status=0x00)</td>
</tr>
<tr>
<td>Config Model Subscription Add</td>
<td>Config Model Subscription Status (Status=0x00)</td>
</tr>
<tr>
<td>Config Vendor Model Subscription Get</td>
<td>Config Vendor Model Subscription List (Status=0x00)</td>
</tr>
</tbody>
</table>
```

Figure 4.98: MESH/NODE/CFG/SL/BV-03-C

• Expected Outcome

**Pass verdict**

The IUT sends a Config Vendor Model Subscription List message in response to the Lower Tester with the Status field set to 0x00, the ElementAddress and Model Identifier field values equal to the values sent in step 1, and the addresses of the two group addresses added in step 1 and step 3.

All configuration messages are encrypted using the device key of the IUT.
4.15.8.4 MESH/NODE/CFG/SL/BI-01-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid ElementAddress)]

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Add message or a Config Model Subscription Virtual Address Add message with an invalid ElementAddress field.

- **Reference**
  [3] Section 4.3.2.19, 4.3.2.20, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- **Initial Condition**
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Element list of the IUT defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Lower Tester sends a Config Model Subscription Add message to the IUT with a valid 16-bit Model Identifier value, the Address field set to a valid group address, and an ElementAddress value within the range of the field but not one of the Elements listed in the IUT’s IXIT [6] table.
  2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).
  3. The Lower Tester sends a Config Model Subscription Virtual Address Add message to the IUT with ElementAddress field and Model Identifier field set to the same as in step 1 and the Address field the Label UUID used as the new address to be added to the Subscription List.
  4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).
• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Model Subscription Status message with the Status field set to 0x01, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

In step 4, the IUT sends a Config Model Subscription Status message with the Status field set to 0x01, the Address field set to the unassigned address, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

All configuration messages are encrypted using the device key of the IUT.

4.15.8.5 MESH/NODE/CFG/SL/BI-02-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid Model Identifier)]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Add message or a Config Model Subscription Virtual Address Add message with an invalid Model Identifier field.

• Reference

[3] Section 4.3.2.19, 4.3.2.20, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

• Initial Condition

- The IUT supports the Configuration Server Model on the primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Model list of the IUT defined in the IXIT [6] parameters.

* Test Procedure

1. The Lower Tester sends a Config Model Subscription Add message to the IUT with a valid ElementAddress, the Address field set to a valid group address, and a Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT’s IXIT [6] table.
2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).
3. The Lower Tester sends a Config Model Subscription Virtual Address Add message to the IUT with the ElementAddress field and Model Identifier field set to the same as in step 1 and the Address field the Label UUID used as the new address to be added to the Subscription List.
4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

![Diagram](image)

* Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Model Subscription Status message with the Status field set to 0x02, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

In step 4, the IUT sends a Config Model Subscription Status message with the Status field set to 0x02, the Address field set to the unassigned address and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

All configuration messages are encrypted using the device key of the IUT.
4.15.8.6 MESH/NODE/CFG/SL/BI-03-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid ElementAddress)]

- **Test Purpose**
  Verify that the IUT can respond with the correct error status to a Config Model Subscription Delete message or a Config Model Subscription Virtual Address Delete message with an invalid ElementAddress value.

- **Reference**
  [3] Section 4.3.2.21, 4.3.2.22, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- **Initial Condition**
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Element list of the IUT defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Lower Tester sends a Config Model Subscription Delete message to the IUT with a valid 16-bit Model Identifier value, the Address field set to a valid group address, and an ElementAddress value within the range of the field but not one of the Elements listed in the IUT’s IXIT [6] table.
  2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).
  3. The Lower Tester sends a Config Model Subscription Virtual Address Delete message to the IUT with the ElementAddress field and Model Identifier field set to the same as in step 1 and the Address field the Label UUID used as the new address to be removed from the Subscription List.
  4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).

*Figure 4.101: MESH/NODE/CFG/SL/BI-03-C*
• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Model Subscription Status message with the Status field set to 0x01 (Invalid Address), and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

In step 4, the IUT sends a Config Model Subscription Status message with the Status field set to 0x01 (Invalid Address), the Address field set to the unassigned address and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

All configuration messages are encrypted using the device key of the IUT.

4.15.8.7 MESH/NODE/CFG/SL/BI-04-C [Respond to Config SIG Model Subscription Get Message (Invalid ElementAddress)]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to a Config SIG Model Subscription Get message with an invalid ElementAddress field.

• Reference

[3] Section 4.3.2.27, 4.3.2.28, 4.4.1.2.8, 4.4.2.2.8

• Initial Condition

- The IUT supports the Configuration Server Model on the primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Element list of the IUT defined in the IXIT [6] parameters.
- The IUT has one or more SIG Models stored on the device, the 16-bit SIG Model IDs specified in the IXIT [6].

• Test Procedure

1. The Lower Tester sends a Config SIG Model Subscription Get message to the IUT with a valid 16-bit SIG Model Identifier value, and an ElementAddress value within the range of the field but not one of the Elements listed in the IUT’s IXIT [6] table.
2. The Lower Tester expects to receive a Config SIG Model Subscription List message in response from the IUT with a Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier field values equal to the values sent in step 1, and the Addresses field set to an empty list.
**Expected Outcome**

- **Pass verdict**

The IUT sends a Config SIG Model Subscription List message with the Status field set to 0x01, the ElementAddress and the Model Identifier field values equal to the values sent in step 1, and the Addresses field set to an empty list.

All configuration messages are encrypted using the device key of the IUT.

**4.15.8.8 MESH/NODE/CFG/SL/BI-05-C [Respond to Config SIG Model Subscription Get Message (Invalid Model Identifier)]**

- **Test Purpose**

Verify that an IUT node supporting the Configuration Server Model can respond to a Config SIG Model Subscription Get message with an invalid Model Identifier field.

- **Reference**

[3] Section 4.3.2.27, 4.3.2.28, 4.4.1.2.8, 4.4.2.2.8

- **Initial Condition**

  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Model list of the IUT defined in the IXIT [6] parameters.
  - The IUT has one or more SIG Models stored on the device, the 16-bit SIG Model IDs specified in the IXIT [6].

- **Test Procedure**

  1. The Lower Tester sends a Config SIG Model Subscription Get message to the IUT with a valid ElementAddress value, and a 16-bit Model Identifier value not matching any SIG Model identified in the IXIT [6] a valid value.
  2. The Lower Tester expects to receive a Config SIG Model Subscription List message in response from the IUT with a Status field set to 0x02 (Invalid Model), an ElementAddress and the Model
Identifier field values equal to the values sent in step 1 and the Addresses field set to an empty list.

Lower Tester

\[\text{Config SIG Model Subscription Get} \]
\[(\text{ElementAddress=address of Element present on IUT, Model Identifier=invalid Model ID})\]

\[\text{Config SIG Model Subscription List} \]
\[(\text{Status=0x02, Addresses=empty list})\]

IUT

Figure 4.103: MESH/NODE/CFG/SL/BI-05-C

• Expected Outcome

Pass verdict

The IUT sends a Config Model Subscription List message with the Status field set to 0x02, the ElementAddress field and the Model Identifier field equal to that in step 1, and the Addresses field set to an empty list.

All configuration messages are encrypted using the device key of the IUT.

4.15.8.9 MESH/NODE/CFG/SL/BI-06-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid Model Identifier)]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Delete message or a Config Model Subscription Virtual Address Delete message with an invalid Model Identifier field.

• Reference

[3] Section 4.3.2.21, 4.3.2.22, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

• Initial Condition

- The IUT supports the Configuration Server Model on the primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Model list of the IUT defined in the IXIT [6] parameters.

• Test Procedure

1. The Lower Tester sends a Config Model Subscription Delete message to the IUT with a valid ElementAddress, the Address field set to a group address to be removed from IUT’s subscription list, and Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT’s IXIT [6] table.
2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model), an Address field equal to the group address set in step 1.

3. The Lower Tester sends a Config Model Subscription Virtual Address Delete message to the IUT with the same ElementAddress and Model Identifier as that in step 1 and the Address field set to a Label UUID which identifies the Address to be removed from the Subscription List.

4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

**Figure 4.104: MESH/NODE/CFG/SL/BI-06-C**

- **Expected Outcome**
  - **Pass verdict**

In step 2, the IUT sends a Config Model Subscription Status message with the Status field set to 0x02, the Address fields set to the group address received in step 1, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

In step 4, the IUT sends a Config Model Subscription Status message with the Status field set to 0x02, the Address fields set to the unassigned address, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

All configuration messages are encrypted using the device key of the IUT.

**4.15.8.10 MESH/NODE/CFG/SL/BI-07-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid ElementAddress)]**

- **Test Purpose**

Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Overwrite message or a Config Model Subscription Virtual Address Overwrite message with an invalid ElementAddress field.
• Reference

[3] Section 4.3.2.23, 4.3.2.24, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

• Initial Condition
  - The IUT supports the Configuration Server Model.
  - The Lower Tester supports the Configuration Client Model.
  - The Lower Tester has no current subscriptions.

• Test Procedure
  1. The Lower Tester sends a Config Model Subscription Overwrite message to the IUT with an ElementAddress field value not of an Element on the IUT, a valid 16-bit Model Identifier value, and the Address value of a valid group address.
  2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x01 (Invalid Address), an ElementAddress field with a value equal to that in step 1, Address field with a value of a valid group address, and a Model Identifier field value equal to that in step 1.
  3. The Lower Tester sends a Config Model Subscription Virtual Address Overwrite message to the IUT with the same ElementAddress and Model Identifier as that in step 1, and the Address field set to a Label UUID which identifies the Address to be added to the Subscription List.
  4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).

![Diagram](Figure 4.105: MESH/NODE/CFG/SL/BI-07-C)

• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Model Subscription Status message with the Status field value of 0x01, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.
In step 4, the IUT sends a Config Model Subscription Status message with the Status field value of 0x01, the Address field set to the unassigned address and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

All configuration messages are encrypted using the device key of the IUT.

4.15.8.11 MESH/NODE/CFG/SL/BI-08-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid Model Identifier)]

• Test Purpose
Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Overwrite message or a Config Model Subscription Virtual Address Overwrite message with an invalid Model Identifier field.

• Reference
[3] Section 4.3.2.23, 4.3.2.24, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

• Initial Condition
- The IUT supports the Configuration Server Model.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has no current subscriptions.

• Test Procedure
1. The Lower Tester sends a Config Model Subscription Overwrite message to the IUT with the ElementAddress field value of the primary Element on the IUT, a valid 16-bit Model Identifier value that is not valid for the IUT (or an invalid Model Identifier not defined, such as 0x0004, if the IUT supports all the defined Model Identifier), and the Address field set to a valid group address.
2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x02 (Invalid Model), and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.
3. The Lower Tester sends a Config Model Subscription Virtual Address Overwrite message to the IUT with the same parameters as that in step 1.
4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).
**Expected Outcome**

**Pass verdict**

In step 2, the IUT sends a Config Model Subscription Status message with the Status field value of 0x02, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

In step 4, the IUT sends a Config Model Subscription Status message with the Status field value of 0x02, the Address field set to the unassigned address and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

All configuration messages are encrypted using the device key of the IUT.

---

**4.15.8.12 MESH/NODE/CFG/SL/BI-09-C [Respond to Config Vendor Model Subscription Get Message (Invalid ElementAddress)]**

- **Test Purpose**
  
  Verify that an IUT node supporting the Configuration Server Model can return the correct Status/Error Code to a Config Vendor Model Subscription Get message with an ElementAddress field value that does not match any stored values.

- **Reference**
  
  [3] Section 4.3.2.29, 4.3.2.30, 4.4.1.2.8, 4.4.2.2.8

- **Initial Condition**
  
  - The IUT supports the Configuration Server Model.
  - The Lower Tester supports the Configuration Client Model.
  - The Lower Tester has previously discovered the primary Element address of the IUT and is using that Element Address for all communication.
- The Lower Tester knows the ElementAddress and Model Identifier of a Vendor Model stored on the IUT.
- The IUT has one or more Vendor Models stored on the device, the 32-bit Vendor Model IDs specified in the IXIT [6].

**Test Procedure**

1. The Lower Tester sends a Config Vendor Model Subscription Get message to the IUT with a valid Vendor Model a valid Model Identifier value, and an ElementAddress value not in the Element address range of the IUT identified in the IXIT [6].
2. The Lower Tester expects to receive a Config Vendor Model Subscription List message in response from the IUT with the Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier field values equal to the values sent in step 1, and the Addresses field set to an empty list.

![Diagram](image)

*Figure 4.107: MESH/NODE/CFG/SL/BI-09-C*

**Expected Outcome**

**Pass verdict**

The IUT sends a Config Vendor Model Subscription List message with the Status field set to 0x01, the ElementAddress field and the Model Identifier field value equal to that in step 1, and the Addresses field set to an empty list.

All configuration messages are encrypted using the device key of the IUT.

**4.15.8.13 MESH/NODE/CFG/SL/BI-10-C [Respond to Config Vendor Model Subscription Get Message (Invalid Model Identifier)]**

**Test Purpose**

Verify that an IUT node supporting the Configuration Server Model can respond to a Config Vendor Model Subscription Get message with an invalid Model Identifier field.

**Reference**

[3] Section 4.3.2.29, 4.3.2.30, 4.4.1.2.8, 4.4.2.2.8

**Initial Condition**

- The IUT supports the Configuration Server Model.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester knows the ElementAddress and Model Identifier of a Vendor Model stored on the IUT.
- The IUT has one or more Vendor Models stored on the device, the 32-bit Vendor Model IDs specified in the IXIT [6].

• Test Procedure
1. The Lower Tester sends a Config Vendor Model Subscription Get message to the IUT with an ElementAddress of an element in IUT and a valid Vendor Model Identifier not matching any Vendor Model identified in the IXIT [6].
2. The Lower Tester expects to receive a Config Vendor Model Subscription List message in response from the IUT with the Status field set to 0x02 (Invalid Model), and the ElementAddress and Model Identifier field values equal to the values sent in step 1 and the Addresses field set to an empty list.

Sample Diagram:

```
Lower Tester --> IUT

Config Vendor Model Subscription Get
(Vendor Model=value not matching any Model ID present on IUT,
ElementAddress=valid value)

Config Vendor Model Subscription List
(Status=0x02, Addresses=empty list)
```

Figure 4.108: MESH/NODE/CFG/SL/BI-10-C

• Expected Outcome

Pass verdict

The IUT sends a Config Vendor Model Subscription List message with the Status field set to 0x02, the ElementAddress field and the Model Identifier field equal to that in step 1, and the Addresses field set to an empty list.

All configuration messages are encrypted using the device key of the IUT.

4.15.8.14 MESH/CFGCL/CFG/SL/BV-01-C [Send Config Model Subscription Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config Model Subscription Add/Get/Delete/Delete All/Overwrite messages to a Configuration Server.

• Reference

[3] Section 3.7.4.3, 4.3.2.19 – 4.3.2.30, 4.4.2.2.8
• **Initial Condition**
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• **Test Procedure**
  1. The Upper Tester orders the IUT to send a Config Model Subscription Add message to the Lower Tester with the ElementAddress and Model Identifier field values as specified in the IXIT [6], and an Address value equal to a group address.
  2. The Lower Tester expects to receive a Config Model Subscription Add message.
  3. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
  4. The Upper Tester orders the IUT to send a Config Model Subscription Delete message to the Lower Tester with the ElementAddress, Address and Model Identifier field values from step 1.
  5. The Lower Tester expects to receive a Config Model Subscription Delete message.
  6. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
  7. The Upper Tester orders the IUT to send a Config Model Subscription Overwrite message to the Lower Tester with ElementAddress, Address and Model Identifier field values from step 1.
  8. The Lower Tester expects to receive a Config Model Subscription Overwrite message.
  9. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
 10. The Upper Tester orders the IUT to send a Config Model Subscription Delete All message to the Lower Tester with the ElementAddress as above and a valid Model Identifier.
 11. The Lower Tester expects to receive a Config Model Subscription Delete All message.
 12. The Lower Tester responds with a Config Model Subscription Status message, setting the ElementAddress and Model Identifier fields as defined by the incoming message, setting the Address field to 0x0000, and setting the Status field to 0x00.
 13. The Upper Tester orders the IUT to send a Config SIG Model Subscription Get message to the Lower Tester with the ElementAddress field set to the unicast address of the Lower Tester, and the Model Identifier field set to a 2-octet value which identifies the Model within the Element.
 14. The Lower Tester expects to receive a Config SIG Model Subscription Get message.
 15. The Lower Tester responds with a Config SIG Model Subscription List message, setting the ElementAddress and Model Identifier fields as defined by the incoming message, setting the Addresses field to an empty list, and setting the Status field to 0x00.
 16. The Upper Tester orders the IUT to send a Config Vendor Model Subscription Get message to the Lower Tester with the ElementAddress field set to the unicast address of the Lower Tester, and the Model Identifier field set to a 4-octet value which identifies the Model within the Element.
 17. The Lower Tester expects to receive a Config Vendor Model Subscription Get message.
 18. The Lower Tester responds with a Config Vendor Model Subscription List message, setting the ElementAddress and Model Identifier fields as defined by the incoming message, setting the Addresses field to an empty list, and setting the Status field to 0x00.
Figure 4.109: MESH/CFGCL/CFG/SL/BV-01-C
• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Model Subscription Add message to the Lower Tester with the ElementAddress and Model Identifier field values equal to those in step 1, the Address field equal to a group address.

In step 6, the IUT sends a Config Model Subscription Delete message to the Lower Tester with the ElementAddress equal to that of the Lower Tester’s, the Address field equal to a group address, and the Model Identifier equal to that of the model on the Lower Tester.

In step 9, the IUT sends a Config Model Subscription Overwrite message to the Lower Tester with the ElementAddress equal to that of the Lower Tester’s, the Address field equal to a group address, and the Model Identifier equal to that of the model on the Lower Tester.

In step 12, the IUT sends a Config Model Subscription Delete All message to the Lower Tester with the ElementAddress and the Model Identifier field values equal to those in step 1.

In step 15, the IUT sends a Config SIG Model Subscription Get message with the ElementAddress equal to that of the Lower Tester’s and the Model Identifier equal to the 2-octet value that identifies the Model within the Element on the Lower Tester.

In step 18, the IUT sends a Config Vendor Model Subscription Get message with the ElementAddress equal to that of the Lower Tester’s and the Model Identifier equal to the 4-octet value that identifies the Model within the Element on the Lower Tester.

All configuration messages are encrypted using the device key of the Lower Tester.

4.15.8.15 MESH/CFGCL/CFG/SL/BV-02-C [Send Config Model Subscription Virtual Address Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config Model Subscription Virtual Address Add/Delete/Overwrite messages to a Configuration Server.

• Reference

[3] Section 3.7.4.3, 4.3.2.20, 4.3.2.22, 4.3.2.24, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

• Initial Condition

- The IUT supports the Configuration Client Model.
- The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure

1. The Upper Tester orders the IUT to send a Config Model Subscription Virtual Address Add message to the Lower Tester with the ElementAddress field set to the Lower Tester’s unicast address and Model Identifier field values as specified in the IXIT [6], and an Address value equal to a valid 16-octet Label UUID.
2. The Lower Tester expects to receive a Config Model Subscription Virtual Address Add message.
3. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
4. The Upper Tester orders the IUT to send a Config Model Subscription Virtual Address Delete message to the Lower Tester with the ElementAddress, Model Identifier field, and the Address set to the same value as in step 1.
5. The Lower Tester expects to receive a Config Model Subscription Virtual Address Delete message.
6. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
7. The Upper Tester orders the IUT to send a Config Model Subscription Virtual Address Overwrite message to the Lower Tester with ElementAddress and Model Identifier field values from step 1, and an Address value equal to a valid 16-octet Label UUID.
8. The Lower Tester expects to receive a Config Model Subscription Virtual Address Overwrite message.
9. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.

Figure 4.110: MESH/CFGCL/CFG/SL/BV-02-C
• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Model Subscription Virtual Address Add message with the ElementAddress, the Model Identifier and the Address fields values equal to those in step 1.

In step 6, the IUT sends a Config Model Subscription Virtual Address Delete message with the ElementAddress, the Model Identifier and the Address fields values equal to those in step 4.

In step 9, the IUT sends a Config Model Subscription Virtual Address Overwrite message with the ElementAddress, the Model Identifier and the Address fields values equal to those in step 7.

All configuration messages are encrypted using the device key of the Lower Tester.

4.15.9 NetKey List Procedures
This section is for validating the NetKey List procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.9.1 MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to Config NetKey Add and Get Messages.

• Reference

[3] Section 3.8.5.1, 4.2.4, 4.3.1.1, 4, 4.3.2.31, 4.3.2.34, 4.3.2.35, 4.3.2.36, 4.4.1.2, 4.4.1.2.9, 4.4.2.2.9

• Initial Condition

- The IUT supports the Configuration Server Model on the primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT is configured only on the primary subnet.

• Test Procedure

1. The Lower Tester sends a Config NetKey Get message to the IUT.
2. The Lower Tester receives a NetKey List message from the IUT with the NetKeyIndexes field equal to 0x0000.
3. The Lower Tester sends a Config NetKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config NetKey Add message contains a NetKeyIdex value of 0x00 and the same NetKey value.
4. The Lower Tester receives a Config NetKey Status message from the IUT with the Status field set to 0x00 and the NetKeyIdex set to 0x000.
5. If the IUT does not support a second subnet, skip the following steps.
6. The Lower Tester sends a Config NetKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config NetKey Add message contains a NetKeyIndex value of 1 (one) and a new NetKey value.

7. The Lower Tester receives a Config NetKey Status message from the IUT with the Status field set to 0x00 and the NetKeyIndex set to the value used in step 6.

8. The Lower Tester sends a Config NetKey Get message to the IUT.

9. The Lower Tester receives a NetKey List message from the IUT using the generic segmentation and reassembly mechanism with the NetKeyIndexes field equal to the two NetKey Indexes present on the IUT.

• **Expected Outcome**

  **Pass Verdict**

  In step 2, the IUT sends a NetKey List message with the NetKeyIndexes field indicating only the NetKeyIndex value of 0x000.

  In steps 4 and 7, the IUT sends a Config NetKey Status message with a Status field set to 0x00 and expected NetKeyIndex value of 0x000 and 0x001, respectively.

---

*Figure 4.111: MESH/NODE/CFG/NKL/BV-01-C*
In step 9, the IUT sends a NetKey List message with the NetKeyIndexes field indicating both NetKey indexes present on the IUT.

All configuration messages are encrypted using the device key of the IUT.

4.15.9.2 MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message]

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Server Model can respond to a Config NetKey Update message.

- **Reference**
  [3] Section 4.3.1.1, 4.3.2.31, 4.3.2.32, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- **Initial Condition**
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on the primary subnet.
  - The IUT is in Key Refresh state 0x00.

- **Test Procedure**
  1. The Lower Tester sends for a Config NetKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0 and a new valid NetKey value.
  2. The Lower Tester receives a Config NetKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex field set to 0.
  3. Return the IUT to the original state (i.e., newly provisioned state).

---

**Figure 4.112: MESH/NODE/CFG/NKL/BV-02-C**
• Expected Outcome

Pass Verdict

The IUT sends a Config NetKey Status Messages with a Status field set to 0x00 and expected NetKeyIndex field value of 0.

All configuration messages are encrypted using the device key of the IUT.

4.15.9.3 MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to a Config NetKey Delete message.

• Reference

[3] Section 4.3.2.33, 4.3.2.34, 4.3.2.36, 4.4.1.2.9, 4.4.2.2.9

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT has a list of two NetKeys with NetKeyIndex of 0 and 1.

• Test Procedure

1. The Lower Tester sends a Config NetKey Delete message to the IUT with a NetKeyIndex value of 1.
2. The Lower Tester receives a Config NetKey Status message in response from the IUT with a Status field set to 0x00 and the NetKeyIndex field set to the value used in step 1.
3. The Lower Tester sends a Config NetKey Get message to the IUT.
4. The Lower Tester receives a NetKey List message from the IUT with the NetKeyIndexes field equal to 0.
5. The Lower Tester repeats steps 1–2, attempting to delete the same key again.
• Expected Outcome

**Pass Verdict**

In both iterations of step 2, the IUT sends a Config NetKey Status message with a Status field set to 0x00 and a NetKeyIndex field value of 1.

In step 4, the IUT sends a NetKey List message with the NetKeyIndexes field equal to 0.

All configuration messages are encrypted using the device key of the IUT.

4.15.9.4 MESH/NODE/CFG/NKL/BV-04-C [Receive Config NetKey Delete Message for Last NetKey]

• Test Purpose

Verify that the IUT supporting the Configuration Server Model with the last NetKey present can respond to a Config NetKey Delete message.

• Reference

[3] Section 4.2.4, 4.3.2.33, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- The IUT is configured only on the primary subnet.

- **Test Procedure**

  1. The Lower Tester sends a Config NetKey Delete message to the IUT with a NetKeyIndex value of 0.

  2. The Lower Tester receives a Config NetKey Status message from the IUT with the Status field set to 0x0C (Cannot Remove), and the NetKeyIndex field set to 0.

- **Expected Outcome**

  **Pass Verdict**

  In step 2, the Config NetKey Status message received from the IUT with the Status field set to 0x0C, and the NetKeyIndex field set to 0.

  All configuration messages are encrypted using the device key of the IUT.

**4.15.9.5 MESH/NODE/CFG/NKL/BI-01-C [Respond to Config NetKey Update Message (Invalid NetKeyIndex)]**

- **Test Purpose**

  Verify that the IUT can respond with the correct error status to a Config NetKey Update message with an invalid NetKeyIndex value.

- **Reference**

  [3] Section 4.3.1.1, 4.3.2.32, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- **Initial Condition**

  - The IUT supports the Configuration Server Model on the Primary Element.

  - The Lower Tester supports Configuration Client Model.

  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

  - The IUT is configured only on the primary subnet.
• Test Procedure
  1. The Lower Tester sends a Config NetKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 2 and a valid NetKey value.
  2. The Lower Tester expects to receive a Config NetKey Status message from the IUT. The message contains a Status field set to 0x04 (Invalid NetKey) and the NetKeyIndex set to value used in step 1.

![Diagram](image.png)

*Figure 4.115: MESH/NODE/CFG/NKL/BI-01-C*

• Expected Outcome
  **Pass Verdict**

The IUT sends a Config NetKey Status message with a Status field set to 0x04 and expected NetKeyIndex in the incoming Config NetKey Update message.

All configuration messages are encrypted using the device key of the IUT.

4.15.9.6 MESH/NODE/CFG/NKL/BI-02-C [Respond to Config NetKey Add Message (NetKeyIndex Already Stored)]

• Test Purpose
  Verify that the IUT can respond with the correct error status to a Config NetKey Add message with a NetKeyIndex already stored.

• Reference
  [3] Section 4.3.2.31, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

• Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on the primary subnet.
• Test Procedure

1. The Lower Tester sends a Config NetKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config NetKey Add message contains a NetKeyIndex value of 0 and a different NetKey value.

2. The Lower Tester receives a Config NetKey Status message from the IUT. The message contains the Status field set to 0x06 (Key Index Already Stored), and the NetKeyIndex field set to 0.

• Expected Outcome

Pass Verdict

The Config NetKey Add is expected to fail and IUT returns a Config NetKey Status message. The message contains the Status field set to 0x06, and the NetKeyIndex field set to 0.

All configuration messages are encrypted using the device key of the IUT.

4.15.9.7 MESH/NODE/CFG/NKL/BI-03-C [Respond to Config NetKey Add Message (Insufficient Resources)]

• Test Purpose

Verify that the IUT can respond with the correct error status to a Config NetKey Add message, if it doesn’t have sufficient resources to handle it.

• Reference

[3] Section 4.3.2.31, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT is configured with the maximum number of network keys supported as defined in the IXIT [6] parameters.
• Test Procedure
  1. The Lower Tester sends a Config NetKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config NetKey Add message contains a different NetKeyIndex value and a different NetKey value.
  2. The Lower Tester receives a Config NetKey Status message from the IUT. The message contains the Status field set to 0x05, and the NetKeyIndex field set to 1.

```
Lower Tester                     IUT
| NetKey Add                     |
| (NetKeyIndex=1,               |
|   NetKey=new NetKey value)    |
| NetKey Status                 |
| (Status=0x05,                 |
|   NetKeyIndex=1)             |
```

*Expected Outcome*

**Pass Verdict**

The Config NetKey Add in step 1 is expected to fail and IUT returns a Config NetKey Status message. The message contains the Status field set to 0x05, and the NetKeyIndex field set to 1.

All configuration messages are encrypted using the device key of the IUT.

4.15.9.8 MESH/CFGCL/CFG/NKL/BV-01-C [Send Config NetKey Add/Get Messages]

• Test Purpose
  Verify that an IUT node supporting the Configuration Client Model can send Config NetKey Add and Config NetKey Get messages to a Configuration Server.

• Reference
  [3] Section 4.3.2.31, 4.3.2.34, 4.3.2.35, 4.4.1.2.9, 4.4.2.2.9

• Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure
  1. The Upper Tester orders the IUT to send a Config NetKey Add message to the Lower Tester using the generic segmentation and reassembly mechanism. The message contains a valid NetKeyIndex and NetKey.
2. The Lower Tester receives a Config NetKey Add message using the generic segmentation and reassembly mechanism.
3. The Lower Tester sends a Config NetKey Status message in response to the IUT.
4. The Upper Tester orders the IUT to send a Config NetKey Get message to the Lower Tester.
5. The Lower Tester receives a Config NetKey Get message using the generic segmentation and reassembly mechanism.
6. The Lower Tester sends a NetKey List message in response to the IUT.

![Diagram]

**Figure 4.118: MESH/CFGCL/CFG/NKL/BV-01-C**

- **Expected Outcome**
  
  Pass verdict

  In step 1, the IUT sends a Config NetKey Add message with a 2-octet unique global NetKeyIndex and the 16-octet NetKey.

  In step 4, the IUT sends a Config NetKey Get message.

  The Config NetKey Add message is encrypted using the device key of the Lower Tester.

**4.15.9.9 MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages]**

- **Test Purpose**
  
  Verify that an IUT node supporting the Configuration Client Model can send Config NetKey Update messages to a Configuration Server.

- **Reference**
  
  [3] Section 4.3.2.32, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- **Initial Condition**
  
  - The IUT supports the Configuration Client Model.
The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure
  1. The Upper Tester orders the IUT to send a Config NetKey Update message to the Lower Tester using the generic segmentation and reassembly mechanism. The message contains a valid NetKeyIndex and a valid NetKey value.
  2. The Lower Tester receives a Config NetKey Update message using the generic segmentation and reassembly mechanism.
  3. The Lower Tester sends a Config NetKey Status message in response to the IUT.

• Expected Outcome
  Pass verdict
  The IUT sends a Config NetKey Update message with a 2-octet existing global NetKeyIndex and the 16-octet NetKey.
  The Config NetKey Update message is encrypted using the device key of the Lower Tester.

4.15.9.10  MESH/CFGCL/CFG/NKL/BV-03-C [Send Config NetKey Delete Messages]

• Test Purpose
  Verify that an IUT node supporting the Configuration Client Model can send Config NetKey Delete messages to a Configuration Server.

• Reference
  [3] Section 4.3.2.33, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

• Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- The Lower Tester has a list of two NetKeys with NetKeyIndex of 0 and 1.

**Test Procedure**

1. The Upper Tester orders the IUT to generate a Config NetKey Delete message to the Lower Tester using a NetKeyIndex value of 1.
2. The Lower Tester expects to receive a Config NetKey Delete message from the IUT.
3. The Lower Tester sends a Config NetKey Status message in response to the IUT.

**Expected Outcome**

Pass verdict

The IUT sends a Config NetKey Delete message with a 2-octet existing global NetKeyIndex value of 1.

The Config NetKey Delete message is encrypted using the device key of the Lower Tester.

### 4.15.10 AppKey List Procedures

This section is for validating the AppKey List procedures for both the Configuration Server Model and the Configuration Client Model.

#### 4.15.10.1 MESH/NODE/CFG/AKL/BV-01-C [Respond to Config AppKey Add/Get Message]

**Test Purpose**

Verify that an IUT node supporting the Configuration Server Model can respond to an Config AppKey Add and Get Messages.

**Reference**

[3] Section 4.3.2.37, 4.3.2.40, 4.3.2.41, 4.3.2.42, 4.4.1.2, 4.4.1.2.10, 4.4.2.2.10

**Initial Condition**

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- The IUT is configured only on a subnet with the NetKeyIndex set to 0x123.

• Test Procedure

1. The Lower Tester sends an Config AppKey Add message to the IUT with a NetKeyIndex value of 0x123, an AppKeyIndex of 0x456 and a 16-octet AppKey value using the generic segmentation and reassembly mechanism.

2. The Lower Tester expects to receive an Config AppKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex and AppKeyIndex field set to value used in step 1.

3. The Lower Tester sends an Config AppKey Get message to the IUT. The message contains a NetKeyIndex field set to 0x123.

4. The Lower Tester receives an AppKey List message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex field and the AppKeyIndexes field set to the values used in step 1.

5. The Lower Tester repeats steps 1–2, attempting to add the same application key with the same indexes.

Figure 4.121: MESH/NODE/CFG/AKL/BV-01-C

• Expected Outcome

Pass Verdict

In both iterations through step 2, the IUT sends an Config AppKey Status message with the Status field set to 0x00, the NetKeyIndex value set to 0x123, and the AppKeyIndex value set to 0x456.

In step 4, the IUT sends an AppKey List message with a Status field set to 0x00, the NetKeyIndex value set to 0x123, and the AppKeyIndex value set to 0x456.

All configuration messages are encrypted using the device key of the IUT.
4.15.10.2 MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]

• Test Purpose
Verify that an IUT node supporting the Configuration Server Model can respond to a Config AppKey Update message.

• Reference
[3] Section 4.3.2.38, 4.3.2.40, 4.3.2.42, 4.4.1.2.10, 4.4.2.2.10

• Initial Condition
- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT is configured only on a subnet with the NetKeyIndex set to 0x123.

• Test Procedure
1. The Lower Tester sends an Config AppKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config AppKey Update message contains a NetKeyIndex value of 0x123 and AppKeyIndex value of 0x456 in the NetKeyIndex and AppKeyIndex field, and a valid new AppKey value in the AppKey field.
2. The Lower Tester receives an Config AppKey Status message in response from the IUT with a Status field set to 0x00, a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.
3. The Lower Tester sends for a Config NetKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0x123 and a new valid NetKey value.
4. The Lower Tester expects to receive a Config NetKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex set to value used in step 3.
5. The Lower Tester sends an Config AppKey Update message to the IUT using the generic segmentation and reassembly mechanism. The Config AppKey Update message contains a NetKeyIndex value of 0x123 and AppKeyIndex value of 0x456 in the NetKeyIndex and AppKeyIndex field, and a valid new AppKey value.
6. The Lower Tester receives an Config AppKey Status message in response from the IUT with the Status field set to 0x00, a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.
7. The Lower Tester sends an Config AppKey Get message to the IUT. The message contains a NetKeyIndex field set to 0x123.
8. The Lower Tester receives an AppKey List message from the IUT. The message contains a Status field set to 0x00, a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.
Expected Outcome

Pass Verdict

In steps 2 and 6, the IUT sends an Config AppKey Status message with a Status field set to 0x00 and the NetKeyIndex and AppKeyIndex field set to 0x236145.

In step 8, the IUT sends an AppKey List message with the Status field set to 0x00, a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.

All configuration messages are encrypted using the device key of the IUT.
4.15.10.3 MESH/NODE/CFG/AKL/BV-03-C [Respond to Config AppKey Delete Message]

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Server Model can respond to a Config AppKey Delete message.

- **Reference**
  [3] Section 4.3.2.39, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- **Initial Condition**
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on a subnet with the NetKeyIndex set to 0x123.
  - The IUT has a list of at least two AppKeys (with AppKeyIndex field value set to 0x000 and 0x456) bound to the NetKey (with NetKeyIndex field value set to 0x123).

- **Test Procedure**
  1. The Lower Tester sends a Config AppKey Delete message to the IUT with the NetKeyIndex of the NetKey to which the AppKey is bound and an AppKeyIndex value of 0x123.
  2. The Lower Tester expects to receive a Config AppKey Status message in response from the IUT with a Status field set to 0x00 and the NetKeyIndex and AppKeyIndex field set to value used in step 1.
  3. The Lower Tester repeats steps 1-2, attempting to delete the same key.

![Diagram of test procedure](image-url)

*Figure 4.123: MESH/NODE/CFG/AKL/BV-03-C*
• Expected Outcome

*Pass Verdict*

In both iterations through step 2, the IUT sends the Config AppKey Status message with a Status field set to 0x00 and NetKeyIndex and AppKeyIndex values equal to those received in step 1.

All configuration messages are encrypted using the device key of the IUT.

### 4.15.10.4 MESH/NODE/CFG/AKL/BI-01-C [Respond to Config AppKey Add Message (Invalid NetKey Index)]

• **Test Purpose**

Verify that an IUT node supporting the Configuration Server Model can respond to a Config AppKey Add message with invalid Config AppKey Add parameters.

• **Reference**

[3] Section 4.3.1.1, 4.3.2.37, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

• **Initial Condition**

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- IUT does not have AppKeyIndex of 0x000 in AppKey list and does not have NetKeyIndex of 0xFFF in NetKey list.

• **Test Procedure**

1. The Lower Tester sends a Config AppKey Add message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0xFFF, an AppKeyIndex value of 0x000, and a valid AppKey value.
2. The Lower Tester receives a Config AppKey Status message from the IUT. The message contains a Status field set to 0x04 (Invalid NetKey) and the NetKeyIndex and AppKeyIndex value matching the value in step 1.

![Figure 4.124: MESH/NODE/CFG/AKL/BI-01-C](image-url)
Expected Outcome

Pass Verdict

In step 2, the IUT sends the Config AppKey Status message with a Status field set to 0x04 and the NetKeyIndex and AppKeyIndex values equal to those sent in step 1.

All configuration messages are encrypted using the device key of the IUT.

4.15.10.5 MESH/NODE/CFG/AKL/BI-02-C [Respond to Config AppKey Update Message (Invalid AppKey Index and NetKey Index)]

Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to an Config AppKey Update message with invalid Config AppKey Update parameters.

Reference

[3] Section 4.3.1.1, 4.3.2.38, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT has an AppKey with AppKeyIndex 0x000 bound to NetKeyIndex 0x000.
- If the IUT supports a second subnet, it is also configured on a subnet with the NetKeyIndex of 0x001, and has an AppKey with AppKeyIndex 0x001 bound to NetKeyIndex 0x001.
- The IUT does not have AppKeyIndex of 0xFFF in AppKey list.
- The IUT does not have NetKeyIndex of 0xFFF in NetKey list.

Test Procedure

1. The Lower Tester sends a Config NetKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0x000 and a new valid NetKey value.
2. The Lower Tester expects to receive a Config NetKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex set to value used in step 1.
3. The Lower Tester sends an Config AppKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0x000 and an AppKeyIndex value of 0xFFF.
4. The Lower Tester expects the IUT to send an Config AppKey Status message with a Status field set to 0x03 (Invalid AppKey) and the NetKeyIndex and AppKeyIndex field value matching the value in step 3.
5. The Lower Tester sends an Config AppKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0xFFF and an AppKeyIndex value of 0x000.
6. The Lower Tester expects the IUT to send an Config AppKey Status message with a Status field set to 0x04 (Invalid NetKey) and the NetKeyIndex and AppKeyIndex field value matching the value in step 5.

7. If the IUT does not support a second subnet, skip steps 8–9.

8. The Lower Tester sends an Config AppKey Update message to the IUT. The message contains a NetKeyIndex value of 0x000, an AppKeyIndex value of 0x001 and a valid AppKey.

9. The Lower Tester expects the IUT to send an Config AppKey Status message with a Status field set to 0x11 (Invalid Binding) and the NetKeyIndex and AppKeyIndex field value matching the value in step 8.

Figure 4.125: MESH/NODE/CFG/AKL/BI-02-C
• Expected Outcome

Pass Verdict

In step 4, the IUT sends an Config AppKey Status message with a Status field set to 0x03 and the NetKeyIndex and AppKeyIndex values equal to those sent in step 3.

In step 6, the IUT sends an Config AppKey Status message with a Status field set to 0x04 and the NetKeyIndex and AppKeyIndex values equal to those sent in step 5.

In step 9, the IUT sends an Config AppKey Status message with a Status field set to 0x11 and the NetKeyIndex and AppKeyIndex values equal to those sent in step 8.

All configuration messages are encrypted using the device key of the IUT.

4.15.10.6 MESH/NODE/CFG/AKL/BI-03-C [Respond to Config AppKey Get Message with Invalid NetKey Index]

• Test Purpose

Verify that a Configuration Server IUT can responds to an Config AppKey Get message with invalid values.

• Reference

[3] Section 4.3.2.41, 4.3.2.42, 4.4.1.2.10, 4.4.2.2.10

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- IUT does not have NetKeyIndex of 0xFFF in NetKey list.

• Test Procedure

1. The Lower Tester sends an Config AppKey Get message to the IUT with a NetKeyIndex value different from any of NetKeyindexes stored on the IUT (e.g., 0xFFF).
2. The Lower Tester expects to receive an AppKey List message in response from the IUT with the Status field set to 0x04 (Invalid NetKey), the NetKeyIndex field as defined by the incoming message, and the AppKeyIndexes field to a zero-length (empty) list.
• Expected Outcome

Pass verdict

The IUT sends an AppKey List message with the Status field set to 0x04, the NetKeyIndex field as defined by the incoming message, and the AppKeyIndexes field to a zero-length (empty) list.

All configuration messages are encrypted using the device key of the IUT.

4.15.10.7  MESH/NODE/CFG/AKL/BL-04-C [Respond to Config AppKey Add Message (NetKeyIndex and AppKeyIndex Already Stored)]

• Test Purpose

Verify that the IUT can respond to a Config AppKey Add message with NetKeyIndex and AppKeyIndex already stored.

• Reference

[3] Section 3.8.5, 4.3.2.37, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT is configured on the primary subnet.
- If the IUT supports a second subnet, it is also configured on a subnet with the NetKeyIndex of 0x001.

• Test Procedure

1. The Lower Tester sends a Config AppKey Add message to the IUT with a NetKeyIndex value of 0x000, an AppKeyIndex value of 0x000, and a valid 16-octet AppKey value using the generic segmentation and reassembly mechanism.
2. The Lower Tester expects the IUT to respond with an Config AppKey Status message with the Status field set to 0x00 and the NetKeyIndex and AppKeyIndex values equal to those sent in step 1.

3. Repeat step 1 with the same NetKeyIndex and AppKeyIndex but different AppKey value.

4. The Lower Tester expects the IUT to respond with an Config AppKey Status message with the Status field set to 0x06 (Key Index Already Stored), and the NetKeyIndex and AppKeyIndex values equal to those sent in step 3.

5. If the IUT does not support a second subnet, skip steps 6–7.

6. Repeat step 1 with the same AppKey, the same AppKeyIndex, but NetKeyIndex field set to 0x001.

7. The Lower Tester expects the IUT to respond with an Config AppKey Status message with the Status field set to 0x04 (Invalid NetKey) and the NetKeyIndex and AppKeyIndex values equal to those sent in step 5.

---

**Figure 4.127: MESH/NODE/CFG/AKL/BI-04-C**

- **Expected Outcome**
  - **Pass Verdict**

  In step 2, the IUT sends an Config AppKey Status message with the Status field set to 0x00, and the NetKeyIndex and AppKeyIndex values equal to those sent in step 1.
In step 4, the IUT sends an Config AppKey Status message with the Status field set to 0x06, and the NetKeyIndex and AppKeyIndex values equal to those sent in step 3.

In step 6, the IUT sends an Config AppKey Status message with the Status field set to 0x04 and the NetKeyIndex and AppKeyIndex values equal to those sent in step 5.

All configuration messages are encrypted using the device key of the IUT.

**4.15.10.8 MESH/CFGCL/CFG/AKL/BV-01-C [Send Config AppKey Add Messages]**

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Client Model can send Config AppKey Add messages to a Configuration Server.

- **Reference**
  [3] Section 4.3.2.37, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- **Initial Condition**
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Upper Tester orders the IUT to send an Config AppKey Add message to the Lower Tester with valid NetKeyIndex and AppKeyIndex and a valid 16-octet AppKey value. The message is transmitted via the generic segmentation and reassembly mechanism.
  2. The Lower Tester receives an Config AppKey Add message from the IUT using the generic segmentation and reassembly mechanism.
  3. The Lower Tester sends an Config AppKey Status message in response to the IUT.

- **Expected Outcome**
  **Pass verdict**

  The IUT sends an Config AppKey Add message containing a valid NetKeyIndex and AppKeyIndex and a valid 16-byte AppKey value.

  The Config AppKey Add message is encrypted using the device key of the Lower Tester.

**4.15.10.9 MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages]**

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Client Model can send Config AppKey Update messages to a Configuration Server.

- **Reference**
  [3] Section 3.10.4, 4.3.2.38, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10
• Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure
  1. The Upper Tester orders the IUT to send an Config AppKey Update message to the Lower Tester using the generic segmentation and reassembly mechanism. The message contains a valid NetKeyIndex and AppKeyIndex and AppKey value.
  2. The Lower Tester receives an Config AppKey Update message using the generic segmentation and reassembly mechanism.
  3. The Lower Tester sends an Config AppKey Status message in response to the IUT.

• Expected Outcome
  Pass verdict

  The IUT sends an Config AppKey Update messages with a valid NetKeyIndex and AppKeyIndex and a valid AppKey.

  The Config AppKey Update message is encrypted using the device key of the Lower Tester.

4.15.10.10  MESH/CFGCL/CFG/AKL/BV-03-C [Send Config AppKey Delete Messages]

• Test Purpose
  Verify that an IUT node supporting the Configuration Client Model can send Config AppKey Delete messages to a Configuration Server.

• Reference
  [3] Section 4.3.2.39, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

• Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
  - The Lower Tester has a NetKeyIndex 0x123 with a NetKey and an AppKeyIndex 0x456 with an AppKey bound to the NetKey.

• Test Procedure
  1. The Upper Tester orders the IUT to send an Config AppKey Delete message to the Lower Tester with a NetKeyIndex value equal to 0x123 and a AppKeyIndex value equal to 0x456.
  2. The Lower Tester receives an Config AppKey Delete message.
  3. The Lower Tester sends a Config AppKey Status message in response to the IUT.
**Expected Outcome**

**Pass verdict**

The IUT sends an Config AppKey Delete message with a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.

The Config AppKey Delete message is encrypted using the device key of the Lower Tester.

**4.15.10.11 MESH/CFGCL/CFG/AKL/BV-04-C [Send Config AppKey Get Messages]**

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Client Model can send Config AppKey Get messages to a Configuration Server.

- **Reference**
  [3] Section 4.3.2.41, 4.3.2.42, 4.4.1.2.10, 4.4.2.2.10

- **Initial Condition**
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Upper Tester orders the IUT to send an Config AppKey Get message to the Lower Tester with a NetKeyIndex value of 0x000.
  2. The Lower Tester receives a Config AppKey Get message with the NetKeyIndex field value set to 0x000.
  3. The Lower Tester sends an AppKey List message in response to the Config AppKey Get message.

- **Expected Outcome**
  **Pass verdict**

  The IUT sends an Config AppKey Get message with a NetKeyIndex value of 0x000.

  The Config AppKey Get message is encrypted using the device key of the Lower Tester.

**4.15.11 Model to AppKey List Procedures**

This section is for validating the Model to AppKey List procedures for both the Configuration Server Model and the Configuration Client Model.

**4.15.11.1 MESH/NODE/CFG/MAKL/BV-01-C [Respond to Config Model App Bind and SIG/Config Vendor Model App Get Messages]**

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model App Bind and SIG/Config Vendor Model App Get Messages.
• Reference

[3] Section 4.3.2.46, 4.3.2.48 – 4.3.2.52, 4.4.1.2, 4.4.1.2.11, 4.4.2.2.11

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.

- The Lower Tester supports Configuration Client Model.

- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- The IUT has an AppKey in the AppKey list with AppKeyIndex of 0x000.

• Test Procedure

1. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to zero (0).

2. The Lower Tester receive a Config Model App Status message in response from the IUT with the Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex set to 0 (zero), and the Model Identifier as referenced in step 1.

3. If the Model Identifier is a 2-octet SIG Model ID, then the Lower Tester sends a Config SIG Model App Get message to the IUT with the same ElementAddress and the same Model Identifier; If the Model Identifier is a 4-octet Vendor Model ID, then the Lower Tester sends a Config Vendor Model App Get message to the IUT with the same ElementAddress and the same Model Identifier.

4. The Lower Tester receives a Config SIG Model App List message (if the message sent in step 3 is Config SIG Model App Get) or a Config Vendor Model App List message (if the message sent in step 3 is Config Vendor Model App Get) in response from the IUT with the Status field value set to 0x00, the ElementAddress field value set to the Element Address used in step 3, Model Identifier field value set to the Model Identifier used in step 3, and the AppKeyIndexes field containing all AppKey indexes bound to the Model, and the AppKeyIndex added in step 1 is included in the list.
Figure 4.128: MESH/NODE/CFG/MAKL/BV-01-C

- Expected Outcome
  
  **Pass verdict**

In step 2, the IUT sends a Config Model App Status message with a Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex set to 0 (zero), and the Model Identifier as referenced in step 1.

In step 4, the IUT sends Config SIG Model App List message (if the message sent in step 3 is Config SIG Model App Get) or a Config Vendor Model App List message (if the message sent in step 3 is Config Vendor Model App Get) with the Status field value set to 0x00, the ElementAddress field value set to the Element Address used in step 3, Model Identifier field value set to the Model Identifier used in step 3, and the AppKeyIndexes field containing all AppKey indexes bound to the Model, and the AppKeyIndex added in step 1 is included in the list.

All configuration messages are encrypted using the device key of the IUT.
4.15.11.2  MESH/NODE/CFG/MAKL/BV-02-C [Respond to Config Model App Unbind and SIG/Config Vendor Model App Get Messages]

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model App Unbind and SIG/Config Vendor Model App Get Messages.

- **Reference**
  [3] Section 4.3.2.47 - 4.3.2.52, 4.4.1.2.11, 4.4.2.2.11

- **Initial Condition**
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - Test case MESH/NODE/CFG/MAKL/BV-01-C [Respond to Config Model App Bind and SIG/Config Vendor Model App Get Messages] has been run. That is, the AppKey identified by AppKeyIndex has been bound to the Model.

- **Test Procedure**
  1. The Lower Tester sends a Config Model App Unbind message to the IUT with a ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6], and the AppKeyIndex field set to 0 (zero).
  2. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex matching that in step 1, and the Model Identifier as referenced in step 1.
  3. If the Model Identifier is a 2-octet SIG Model ID, then the Lower Tester sends a Config SIG Model App Get message to the IUT with the same ElementAddress and the same Model Identifier; If the Model Identifier is a 4-octet Vendor Model ID, then the Lower Tester sends a Config Vendor Model App Get message to the IUT with the same ElementAddress and the same Model Identifier.
  4. The Lower Tester receives a Config SIG Model App List message (if the message sent in step 3 is Config SIG Model App Get) or a Config Vendor Model App List message (if the message sent in step 3 is Config Vendor Model App Get) in response from the IUT with the Status field value set to 0x00, the ElementAddress field value set to the Element Address used in step 3, Model Identifier field value set to the Model Identifier used in step 3, and the AppKeyIndexes field containing all AppKey indexes bound to the Model, and the AppKeyIndex of 0x000 is not included in the list.
**Expected Outcome**

**Pass verdict**

In step 2, the IUT sends a Config Model App Status message with a Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex matching that in step 1, and the Model Identifier as referenced in step 1.

In step 4, the IUT sends Config SIG Model App List message (if the message sent in step 3 is Config SIG Model App Get) or a Config Vendor Model App List message (if the message sent in step 3 is Config Vendor Model App Get) with the Status field value set to 0x00, the ElementAddress field value set to the Element Address used in step 3, Model Identifier field value set to the Model Identifier used in step 3, and the AppKeyIndexes field containing all AppKey indexes bound to the Model, and the AppKeyIndex 0x000 is not included in the list.

All configuration messages are encrypted using the device key of the IUT.
4.15.11.3 MESH/NODE/CFG/MAKL/BI-01-C [Respond to Config Model App Bind Message with Invalid Values]

- **Test Purpose**
  Verify that a Configuration Server IUT can respond to a Config Model App Bind message with values that cannot be accepted.

- **Reference**
  [3] Section 4.3.2.46, 4.3.2.48, 4.4.1.2.11, 4.4.2.2.11

- **Initial Condition**
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has at least two AppKeys in the AppKey list and bound to a Model.

- **Test Procedure**
  1. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to an invalid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a valid AppKeyIndex in IUT.
  2. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x02 (Invalid Model) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 1.
  3. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to an address which is not the address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a valid value in IUT.
  4. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x01 (Invalid Address) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 3.
  5. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the address of the Element, the Model Identifier field set to a value (e.g., 0xFFF) which is not stored in IUT.
  6. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x03 (Invalid AppKey) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 5.
  7. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the address of the Element, the Model Identifier field set to the Model Identifier of the Configuration Server Model and the AppKeyIndex field set to a valid value which is stored on the IUT.
  8. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x0D (Cannot Bind) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 7.
**Expected Outcome**

*Pass verdict*

In step 2, IUT sends a Config Model App Status message with the Status field set to 0x02 (Invalid Model) and other fields (ElementAddress, AppKeyId and Model Identifier) set to the values of the corresponding fields of the incoming message in step 1.

In step 4, IUT sends a Config Model App Status message with the Status field set to 0x01 (Invalid address) and other fields (ElementAddress, AppKeyId and Model Identifier) set to the values of the corresponding fields of the incoming message in step 3.
In step 6, IUT sends a Config Model App Status message with the Status field set to 0x03 (Invalid AppKey) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 5.

In step 8, IUT sends a Config Model App Status message with the Status field set to 0x0D (Cannot Bind) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 7.

The Config Model App Status message is encrypted using the device key of the IUT.

**4.15.11.4 MESH/NODE/CFG/MAKL/BI-02-C [Respond to Config Model App Unbind Message with Invalid Values]**

- **Test Purpose**
Verify that a Configuration Server IUT can respond to a Config Model App Unbind message with values that cannot be accepted.

- **Reference**
[3] Section 4.3.2.47, 4.3.2.48, 4.4.1.2.11, 4.4.2.2.11

- **Initial Condition**
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- **Test Procedure**
1. The Lower Tester sends a Config Model App Unbind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6], and an invalid AppKeyIndex field value.
2. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x03 (Invalid AppKey), and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 1.
3. The Lower Tester sends a Config Model App Unbind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to an invalid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a valid AppKeyIndex in IUT.
4. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x02 (Invalid Model) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 3.
5. The Lower Tester sends a Config Model App Unbind message to the IUT with an ElementAddress field set to an address which is not the address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a valid value in IUT.
6. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x01 (Invalid Address) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 5.
Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Model App Status message with a Status field set to 0x03 (Invalid AppKey), and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 1.

In step 4, IUT sends a Config Model App Status message with the Status field set to 0x02 (Invalid Model) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 3.

In step 6, IUT sends a Config Model App Status message with the Status field set to 0x01 (Invalid address) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in step 5.

The Config Model App Status messages are encrypted using the device key of the IUT.
### 4.15.11.5 MESH/NODE/CFG/MAKL/BI-03-C [Respond to Config SIG Model App Get Message with Invalid Values]

- **Test Purpose**
  Verify that a Configuration Server IUT can respond to a Config SIG Model App Get message with invalid values.

- **Reference**
  [3] Section 4.3.2.49, 4.3.2.50, 4.4.1.2.11, 4.4.2.2.11

- **Initial Condition**
  - The IUT supports the Configuration Server Model on the Primary Element and supports a SIG Model.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has at least one AppKey bound to a Model.

- **Test Procedure**
  1. The Lower Tester sends a Config SIG Model App Get message to the IUT with the ElementAddress field set to the unicast address of the Element and Model Identifier field set to an invalid Model Identifier value.
  2. The Lower Tester receives a Config SIG Model App List message in response from the IUT with the Status field set to 0x02 (Invalid Model), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in step 1, and the AppKeyIndexes field to a zero-length (empty) list.
  3. The Lower Tester sends a Config SIG Model App Get message to the IUT with an ElementAddress field set to an address which is not the address of any of the Elements of the IUT, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6].
  4. The Lower Tester receives a Config SIG Model App List message in response from the IUT with the Status field set to 0x01 (Invalid Address), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in step 3, and the AppKeyIndexes field to a zero-length (empty) list.
Figure 4.132: MESH/NODE/CFG/MAKL/BI-03-C

- Expected Outcome
  **Pass verdict**

In step 2, the IUT sends a Config SIG Model App List message in response to the Lower Tester with the Status field set to 0x02 (Invalid Model), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in step 1, and the AppKeyIndexes field to a zero-length (empty) list.

In step 4, the IUT sends a Config SIG Model App List message in response to the Lower Tester with the Status field set to 0x01 (Invalid Address), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in step 3, and the AppKeyIndexes field to a zero-length (empty) list.

The Config SIG Model App List messages are encrypted using the device key of the IUT.

4.15.11.6 **MESH/NODE/CFG/MAKL/BI-04-C [Respond to Config Vendor Model App Get Message with Invalid Values]**

- Test Purpose
  Verify that a Configuration Server IUT can respond to a Config Vendor Model App Get message with invalid values.

- Reference
  [3] Section 4.3.2.51, 4.3.2.52, 4.4.1.2.11, 4.4.2.2.11
• Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element and supports a Vendor Model.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has at least one AppKey bound to a Model.

• Test Procedure
  1. The Lower Tester sends a Config Vendor Model App Get message to the IUT with the ElementAddress field set to the unicast address of the Element and Model Identifier field set to a Model Identifier value not defined in the IXIT [6].
  2. The Lower Tester receives a Config Vendor Model App List message in response from the IUT with the Status field set to 0x02 (Invalid Model), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in step 1, and the AppKeyIndexes field to a zero-length (empty) list.
  3. The Lower Tester sends a Config Vendor Model App Get message to the IUT with an ElementAddress field set to an address which is not the address of any of the Elements of the IUT, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6].
  4. The Lower Tester receives a Config Vendor Model App List message in response from the IUT with the Status field set to 0x01 (Invalid Address), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in step 3, and the AppKeyIndexes field to a zero-length (empty) list.

Figure 4.133: MESH/NODE/CFG/MAKL/BI-04-C
Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Vendor Model App List message in response to the Lower Tester with the Status field set to 0x02 (Invalid Model), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in step 1, and the AppKeyIndexes field to a zero-length (empty) list.

In step 4, the IUT sends a Config Vendor Model App List message in response to the Lower Tester with the Status field set to 0x01 (Invalid Address), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in step 3, and the AppKeyIndexes field to a zero-length (empty) list.

The Config Vendor Model App List messages are encrypted using the device key of the IUT.

4.15.11.7 MESH/CFGCL/CFG/MAKL/BV-01-C [Send Config Model App Bind Messages]

Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config Model App Bind messages to a Configuration Server.

Reference

[3] Section 4.3.2.46, 4.3.2.48, 4.4.1.2.11, 4.4.2.2.11

Initial Condition

- The IUT supports the Configuration Client Model.
- The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

Test Procedure

1. The Upper Tester orders the IUT to send a Config Model App Bind message to the Lower Tester with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to zero (0).
2. The Lower Tester receives the Config Model App Bind message with the expected values.
3. The Lower Tester sends a Config Model App Status message in response to the IUT.
• **Expected Outcome**

  **Pass verdict**

  The IUT sends a Config Model App Bind message from the IUT containing all valid fields – the ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to zero (0).

  The Config Model App Bind messages are encrypted using the device key of the Lower Tester.

4.15.11.8  **MESH/CFGCL/CFG/MAKL/BV-02-C [Send Config Model App Unbind Messages]**

• **Test Purpose**

  Verify that an IUT node supporting the Configuration Client Model can send Config Model App Unbind messages to a Configuration Server.

• **Reference**

  [3] Section 4.3.2.47, 4.3.2.48, 4.4.1.2.11, 4.4.2.2.11

• **Initial Condition**

  - The IUT supports Configuration Client Model.
  - The Lower Tester supports Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• **Test Procedure**

  1. The Upper Tester orders the IUT to send a Config Model App Unbind message to the Lower Tester with the ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6], and the AppKeyIndex field set to 0 (zero).
  2. The Lower Tester receives the Config Model App Unbind message.
  3. The Lower Tester sends a Config Model App Status message in response to the IUT.
• Expected Outcome

Pass verdict

The IUT sends a Config Model App Unbind message from the IUT containing all valid fields - the ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier, and the AppKeyIndex field set to 0 (zero).

The Config Model App Unbind message is encrypted using the device key of the Lower Tester.

4.15.11.9 MESH/CFGCL/CFG/MAKL/BV-03-C [Send Config SIG Model App Get Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config SIG Model App Get messages to a Configuration Server.

• Reference

[3] Section 4.3.2.49, 4.3.2.50, 4.4.1.2.11, 4.4.2.2.11

• Initial Condition

- The IUT supports Configuration Client Model.
- The Lower Tester supports Configuration Server Model supporting SIG Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure

1. The Upper Tester orders the IUT to send a Config SIG Model App Get message to the Lower Tester with the ElementAddress set to the address of the Element and Model Identifier set to the SIG Model ID that identifies the Model within the Element.
2. The Lower Tester receives the Config SIG Model App Get message containing the expected ElementAddress and Model Identifier values.
3. The Lower Tester sends a Config SIG Model App List message in response to the IUT.
Figure 4.136: MESH/CFGCL/CFG/MAKL/BV-03-C

- Expected Outcome

  Pass verdict

  The IUT sends a Config SIG Model App Get message to the Lower Tester with the ElementAddress set to the address of the Element and the SIG Model Identifier the SIG Model ID that identifies the Model within the Element.

  The Config SIG Model App Get message is encrypted using the device key of the Lower Tester.

4.15.11.10 MESH/CFGCL/CFG/MAKL/BV-04-C [Send Config Vendor Model App Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Vendor Model App Get messages to a Configuration Server.

- Reference

  [3] Section 4.3.2.51, 4.3.2.52, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition

  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config Vendor Model App Get message to the Lower Tester with the ElementAddress set to the address of the Element and Model Identifier the Vendor Model ID that identifies the Model within the Element.
  2. The Lower Tester receives the Config Vendor Model App Get message containing the expected ElementAddress and Model Identifier values.
  3. The Lower Tester sends a Config Vendor Model App List message in response to the IUT.
• Expected Outcome
  
  **Pass verdict**

  The IUT sends a Config Vendor Model App Get message to the Lower Tester with the ElementAddress set to the address of the Element and the Model Identifier the Vendor Model ID that identifies the Model within the Element.

  The Config Vendor Model App Get message is encrypted using the device key of the Lower Tester.

4.15.12  **Node Identity Procedures**

This section is for validating the Node Identity procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.12.1  **MESH/NODE/CFG/NID/BV-01-C [Receive Config Node Identity Get/Set Messages]**

• Test Purpose
  
  Verify that a Configuration Server IUT can respond to a Config Node Identity Get/Set message.

• Reference
  
  [3] Section 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

• Initial Condition
  
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The IUT supports the Mesh Proxy Service.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned with a NetKey identified by the NetKeyIndex of 0x000 and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

• Test Procedure
  
  1. The Lower Tester sends a Config Node Identity Set message with the NetKeyIndex field set to 0x000, and the Identity field set to 0x00 which means Mesh Proxy Service is supported, but Node Identity for a Subnet is stopped.
2. The Lower Tester expects the IUT to receive the Config Node Identity Set message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.

3. The Lower Tester sends a Config Node Identity Get message with the same NetKeyIndex used in step 1 to the IUT.

4. The Lower Tester expects the IUT to receive the Config Node Identity Get message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.

5. The Lower Tester sends a Config Node Identity Set message with the NetKeyIndex field set to 0x000, and the Identity field set to 0x01 which means Mesh Proxy Service is supported, and Node Identity for a Subnet is running.

6. The Lower Tester expects the IUT to receive the Config Node Identity Set message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.

7. The Lower Tester sends a Config Node Identity Get message with the same NetKeyIndex used in step 5 to the IUT.

8. The Lower Tester expects the IUT to receive the Config Node Identity Get message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.

Figure 4.138: MESH/NODE/CFG/NID/BV-01-C
• Expected Outcome

Pass verdict

In steps 2 and 4, IUT responds with a Config Node Identity Status message with Status field value set to 0x00, NetKeyIndex field value set to 0x000 and Identity field value set to 0x00.

In steps 6 and 8, IUT responds with a Config Node Identity Status message with Status field value set to 0x00, NetKeyIndex field value set to 0x000 and Identity field value set to 0x01.

The Config Node Identity Status messages are encrypted using the device key of the IUT.

4.15.12.2 MESH/NODE/CFG/NID/BV-02-C [Node Identity Advertising on Multiple Subnets]

• Test Purpose

Verify that a Configuration Server IUT can trigger Node Identity advertising based on the Node Identity state change and can interleave advertising for multiple Subnets.

• Reference

[3] Section 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT exposes the Mesh Proxy Service.
- The IUT supports multiple subnets.

• Test Procedure

1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT.
2. The Lower Tester sends a Config Node Identity Set message to the IUT with the NetKeyIndex field set randomly to 0x00 or 0x01 and the Identity field set to 0x01.
3. The Lower Tester expects the IUT to respond with a Config Node Identity Status message indicating the state change and to begin advertising with Node Identity only for the Subnet identified by the NetKeyIndex sent in step 2.
4. The Lower Tester sends a Config Node Identity Set message to the IUT with the NetKeyIndex field set to the same value as in step 2 and the Identity field set to 0x00.
5. The Lower Tester expects the IUT to respond with a Config Node Identity Status message indicating the state change and to stop advertising with Node Identity.
6. The Upper Tester orders the IUT to begin advertising with Node Identity, simulating user interaction.
7. The Lower Tester expects the IUT to begin sending interleaved Node Identity advertisings for both Subnets it supports.
8. The Upper Tester orders the IUT to stop advertising with Node Identity.
Expected Outcome

Pass verdict

In step 3, the IUT sends a Config Node Identity Status message with the Status field set to 0x00, the Identity field set to 0x01 and the NetKeyIndex field set to the value requested in step 2. The IUT then begins advertising with Node Identity only for the Subnet identified by the NetKeyIndex sent in step 2.

In step 5, the IUT sends a Config Node Identity Status message with the Status field set to 0x00, the Identity field set to 0x00 and the NetKeyIndex field set to the value requested in step 2. The IUT then stops advertising with Node Identity.

In step 7, the IUT begins advertising with Node Identity for both supported Subnets, interleaving the advertising packets for the two Subnets.

The Config Node Identity Status messages are encrypted using the device key of the IUT.
4.15.12.3 MESH/NODE/CFG/NID/BV-03-C [Node Identity Config NetKey Deleted]

• Test Purpose
  Verify that a Configuration Server IUT sets the Node Identity state to 0x00 when NetKey is deleted.

• Reference
  [3] Section 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

• Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT supports the Mesh Proxy Service.
  - The IUT supports multiple subnets.

• Test Procedure
  1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT with NetKeyIndex 1.
  2. The Lower Tester sends a Config Node Identity Set message to the IUT with the NetKeyIndex field set to 0x0001 and the Identity field set to 0x01.
  3. The Lower Tester expects the IUT to respond with a Config Node Identity Status message with the Status field set to 0x00, the NetKeyIndex field set to 0x0001 and the Identity field set to 0x01.
  4. The Lower Tester deletes a NetKey from IUT’s NetKey List, following the procedure described in MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message]. The NetKeyIndex field is set to 0x0001.
  5. Lower Tester sends a Config Node Identity Get message to the IUT with the NetKeyIndex field set to 0x0001.
  6. The Lower Tester expects the IUT to respond with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), the NetKeyIndex field set to 0x0001 and the Identity field set to 0x00.
Figure 4.140: MESH/NODE/CFG/NID/BV-03-C

- Expected Outcome
  Pass verdict

  In step 3, the IUT sends a Config Node Identity Status message with the Status field set to 0x00, the NetKeyIndex field set to 0x0001 and the Identity field set to 0x01.

  In step 6, the IUT sends a Config Node Identity Status message with the Status field set to 0x04, the NetKeyIndex field set to 0x0001 and the Identity field set to 0x00.

  All the Configuration messages are encrypted using the device key of the IUT.

4.15.12.4 MESH/NODE/CFG/NID/BI-01-C [Receive Config Node Identity Get/Set Messages – Invalid Parameter]

- Test Purpose
  Verify that a Configuration Server IUT can respond to a Config Node Identity Get/Set message with invalid parameters.

- Reference
  [3] Section 3.1.2, 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12
• Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT supports the Mesh Proxy Service.

• Test Procedure
  1. The Lower Tester sends a Config Node Identity Set message with a NetKeyIndex which doesn’t exist in the IUT’s NetKey list, and an Identity field set to 0x00.
  2. The Lower Tester expects the IUT to respond with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), and the NetKeyIndex and Identity fields as defined in the incoming message in step 1.
  3. The Lower Tester sends a Config Node Identity Get message with the same invalid NetKeyIndex used in step 1 to the IUT.
  4. The Lower Tester expects the IUT to respond with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), the Identity field set to 0x00, and the NetKeyIndex as defined in the incoming message in step 6.
  5. The Lower Tester sends a Config Node Identity Set message with a valid NetKeyIndex and the Identity field set to a random number between 0x03 and 0xFF (RFU).
  6. The Lower Tester expects the IUT to ignore the message and does not respond with any messages.

Figure 4.141: MESH/NODE/CFG/NID/BI-01-C
• Expected Outcome

Pass verdict

In step 2, the IUT responds with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), and the NetKeyIndex and Identity fields as defined in the incoming message in step 1.

In step 4, the IUT responds with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), the Identity field set to 0x00, and the NetKeyIndex as defined in the incoming message in step 3.

In step 6, the IUT does not respond with any messages.

The Config Node Identity Status messages are encrypted using the device key of the IUT.

4.15.12.5 MESH/NODE/CFG/NID/BI-03-C [Receive Config Node Identity Set Message – Change to Not Supported]

• Test Purpose

Verify that a Configuration Server IUT supporting the Mesh Proxy Service ignores the Config Node Identity Set message to change the Node Identity to be not supported.

• Reference

[3] Section 3.1.2, 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

• Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned with a NetKey identified by the NetKeyIndex of 0x000 and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT supports the Mesh Proxy Service and has the Identity field set to 0x00.

• Test Procedure

1. The Lower Tester sends a Config Node Identity Set message with a valid NetKeyIndex which is an index that identifies the global NetKey Index of the NetKey of the Node Identity state, and an Identity field set to 0x02 which means Mesh Proxy Service is not supported.
2. The Lower Tester expects the IUT to ignore the message and does not respond with any messages.
3. The Lower Tester sends a Config Node Identity Get message with the same NetKeyIndex used in step 1 to the IUT.
4. The Lower Tester expects the IUT to receive the Config Node Identity Get message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.
• **Expected Outcome**

Pass verdict

In step 2, the IUT does not respond with any messages.

In step 4, the IUT responds with a Config Node Identity Status message with the Status field set to 0x00, the NetKeyIndex field set to 0x000 and the Identity field set to 0x00.

The Config Node Identity Status messages are encrypted using the device key of the IUT.

### 4.15.12.6 MESH/CFGCL/CFG/NID/BV-01-C [Send Config Node Identity Get/Set Messages]

• **Test Purpose**

Verify that an IUT node supporting the Configuration Client Model can send Config Node Identity Get/Set messages to a Configuration Server.

• **Reference**

[3] Section 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.2.2.11

• **Initial Condition**

- The IUT supports the Configuration Client Model.
- The Lower Tester supports the Configuration Server Model.
- The Lower Tester supports the Mesh Proxy Service.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
• Test Procedure

1. The Upper Tester orders the IUT to send a Config Node Identity Set message to the Lower Tester with a valid NetKeyIndex which is an index that identifies the global NetKey Index of the NetKey of the Node Identity state, and an Identity field set to 0x00.
2. The Lower Tester expects to receive a Config Node Identity Set message with the fields set with the values defined in step 1.
3. The Lower Tester responds with a Config Node Identity Status message.
4. The Upper Tester orders the IUT to send a Config Node Identity Set message to the Lower Tester with the same NetKeyIndex value as in step 1, and an Identity field set to 0x01.
5. The Lower Tester expects to receive a Config Node Identity Set message with the fields set with the values defined in step 3.
6. The Lower Tester responds with a Config Node Identity Status message.
7. The Upper Tester orders the IUT to send the Config Node Identity Get message to the Lower Tester with the same NetKeyIndex value as in step 1.
8. The Lower Tester expects to receive a Config Node Identity Get message with the fields set with the values defined in step 5.

Figure 4.143: MESH/CFGCL/CFG/NID/BV-01-C
• **Expected Outcome**

  **Pass verdict**

In step 2, the IUT sent a properly formatted Config Node Identity Set message with a valid NetKeyIndex field and the Identity field value set to 0x00.

In step 5, the IUT sent a properly formatted Config Node Identity Set message with the Identity field value set to 0x01.

In step 8, the IUT sent a properly formatted Config Node Identity Get message with the same NetKeyIndex field value as in step 1.

The Config Node Identity Set messages are encrypted using the device key of the IUT.

### 4.15.13 Reset Procedures

This section is for validating the Reset procedures for both the Configuration Server Model and the Configuration Client Model.

#### 4.15.13.1 MESH/NODE/CFG/RST/BV-01-C [Receive Config Node Reset Message]

• **Test Purpose**

  Verify that a Configuration Server IUT can respond to a Config Node Reset message.

• **Reference**

  [3] Section 3.10.7, 4.3.2.53, 4.3.2.54, 4.4.1.2, 4.4.1.2.13

• **Initial Condition**

  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

• **Test Procedure**

  1. The Lower Tester sends a Config Beacon Set message to the IUT with the Beacon field set to 0x01.
  2. The Lower Tester receives a Config Beacon Status message from the IUT with the Beacon field value set to 0x01.
  3. The Lower Tester sends a Config Node Reset message to the IUT.
  4. The Lower Tester receives a Config Node Reset Status message in response from the IUT.
Figure 4.144: MESH/NODE/CFG/RST/BV-01-C

• Expected Outcome

Pass verdict

In step 4, the IUT sends a Config Node Reset Status message to the Lower Tester.

The Config Node Reset Status message is encrypted using the device key of the IUT.

4.15.13.2 MESH/CFGCL/CFG/RST/BV-01-C [Send Config Node Reset Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config Node Reset messages to a Configuration Server.

• Reference

[3] Section 3.10.7, 4.3.2.53, 4.3.2.54, 4.4.1.2.13, 4.4.2.2.13

• Initial Condition

- The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure

1. The Upper Tester orders the IUT to send a Config Node Reset message to the Lower Tester.
2. The Lower Tester sends a Config Node Reset Status message in response to the IUT.

Figure 4.145: MESH/CFGCL/CFG/RST/BV-01-C
4.15.14 Heartbeat Publication Procedures

This section is for validating the Heartbeat Publication procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.14.1 MESH/NODE/CFG/HBP/BV-01-C [Receive Config Heartbeat Publication Set/Get messages]

- **Test Purpose**
  
  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Heartbeat Publication Set message and update the identified Heartbeat Publication state.

- **Reference**
  
  [3] Section 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.17.1, 4.3.2.61, 4.3.2.62, 4.3.2.63, 4.4.1.2, 4.4.1.2.15, 4.4.2.2.15

- **Initial Condition**
  
  - The IUT is a node supporting the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and has a NetKey identified by NetKeyIndex of 0x000.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- **Test Procedure**
  
  1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT with the CountLog field set to 0x0C, the PeriodLog field set to 0x04, the TTL field set to 0x00, the Features field set to 0x0000, the Destination field set to the Lower Tester’s unicast address, and the NetKeyIndex field set to 0x0000.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
  3. The Lower Tester sends a Config Heartbeat Publication Get message to the IUT.
  4. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the same field values received in step 2.
Expected Outcome

Pass verdict

In steps 2 and 4, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message sent by the Lower Tester in step 1.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.14.2 MESH/NODE/CFG/HBP/BV-02-C [Periodic publishing of Heartbeat messages]

Test Purpose

Verify that the IUT can send Heartbeat messages as configured by a node that supports the Configuration Client Model.

Reference

[3] Section 3.1.2, 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.17.1, 4.2.17.6, 4.3.2.62, 4.3.2.63, 4.4.1.2.15, 4.4.2.2.15

Initial Condition

- The IUT is a node supporting the Configuration Server Model on the primary Element.
- The IUT’s primary Element has a dynamically-assigned group address programmed into.
- The Lower Tester is a node supporting the Configuration Client Model.
- The IUT has been provisioned.
- The IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
• Test Procedure

1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the Destination field set to the Lower Tester’s unicast address, the CountLog field set to 0x02, the PeriodLog field set to 0x02, the TTL field set to 0x7F, the Features field set to 0xFFF0, and the NetKeyIndex field set to the Index of the NetKey of the IUT.

2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00, the Features field set to 0x0000 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.

3. The Lower Tester expects the IUT to send two Heartbeat messages to the address set in the Destination field of the Config Heartbeat Publication Set message sent in step 1, with the InitTTL field set to 0x7F. The Lower Tester expects to receive the Heartbeat messages within an interval of 2 seconds (+/- 250 ms).

4. The Lower Tester repeats steps 1-3 using the Lower Tester’s dynamically-assigned group address as the Destination field of the Config Heartbeat Publication Set message sent in step 1.

5. The Lower Tester repeats steps 1-2 using the unassigned address (0x0000) instead of its unicast address in the Destination field of the Config Heartbeat Publication Set message sent in step 1.

6. Wait 2.5 seconds, and the Lower Tester expects the IUT not to send any other Heartbeat messages.
Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00, the Features field set to 0x0000 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message.
In step 3, the IUT sends two Heartbeat messages with the DST field set to the address from the Destination field of the Config Heartbeat Publication Set message. The time interval between the two Heartbeat messages is 2 seconds. The Opcode field of the Transport Control message is set to 0x0A.

In step 4, the IUT sends the Heartbeat messages as in step 2 and step 3.

In step 5, the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the values of the CountLog, PeriodLog, and TTL fields set to 0x00.

In step 6, the IUT does not send any other Heartbeat messages.

All configuration messages are encrypted and authenticated using the device key of the IUT.

The NetKey used by Transport Control message is the NetKey from step 1.

4.15.14.3 MESH/NODE/CFG/HBP/BV-03-C [Triggered publishing of Heartbeat messages – Relay State changes]

• Test Purpose
Verify that the IUT can send Heartbeat Messages triggered by the Relay State changes on the node, when configured to do so by a node supporting the Configuration Client Model.

• Reference
[3] Section 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.8, 4.3.2.13, 4.3.2.14, 4.3.2.62, 4.3.2.63, 4.4.1.2.6, 4.4.2.2.6

• Initial Condition
- The IUT supports the Relay feature and the feature is disabled.
- The Lower Tester is a node supporting the Configuration Client Model.
- The IUT has been provisioned.
- The IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

• Test Procedure
1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0001 and all the other fields set to valid values.
2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
3. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01.
4. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x01 and to send a Heartbeat message with the Relay bit of the Features field set to 1.
5. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x00.
6. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x00 and to send a Heartbeat message with the Relay bit of the Features field set to 0.
7. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x00.
8. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x00 and not to send any Heartbeat message.
9. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0000 and all the other fields set to valid values.

10. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.

11. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01.

12. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x01 and not to send any Heartbeat message.

Figure 4.148: MESH/NODE/CFG/HBP/BV-03-C
• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in step 1.

In step 4, the IUT sends a Config Relay Status message with the Relay field set to 0x01 and a Heartbeat message with the Relay bit of the Features field set to 1. The Heartbeat message has the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message received in step 1, and the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message. The Opcode field of the Transport Control message is set to 0x0A and the NetKey used is from step 1.

In step 6, the IUT sends a Config Relay Status message with the Relay field set to 0x00 and a Heartbeat message with the Relay bit of the Features field set to 0. The Heartbeat message has the DST field and the InitTTL field as in step 4.

In step 8, the IUT sends a Config Relay Status message with the Relay field set to 0x00 and does not send a Heartbeat message.

In step 10, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in step 9.

In step 12, the IUT sends a Config Relay Status message with the Relay field set to 0x01 and does not send a Heartbeat message.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.14.4 MESH/NODE/CFG/HBP/BV-04-C [Triggered publishing of Heartbeat messages - Friend state changes]

• Test Purpose

Verify that the IUT can send Heartbeat Messages triggered by the Friend State changes on the node, when configured to do so by a node supporting the Configuration Client Model.

• Reference

[3] Section 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.13, 4.3.2.56, 4.3.2.57, 4.3.2.62, 4.3.2.63, 4.4.1.2.5, 4.4.2.2.5

• Initial Condition

- The IUT has the Friend feature disabled.
- The Lower Tester is a node supporting the Configuration Client Model.
- The IUT has been provisioned.
- The IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
• Test Procedure

1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0004 and all the other fields set to valid values.
2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
3. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x01.
4. The Lower Tester expects the IUT to respond with a Config Friend Status message with the Friend field set to 0x01 and send a Heartbeat message with the Friend bit of the Features field set to 1.
5. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x00.
6. The Lower Tester expects the IUT to respond with a Config Friend Status message with the Friend field set to 0x00 and send a Heartbeat message with the Friend bit of the Features field set to 0.
7. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0000 and all the other fields set to valid values.
8. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
9. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x01.
10. The Lower Tester expects the IUT to respond with a Config Friend Status message with the Friend field set to 0x01, but not to send any Heartbeat message.
**Expected Outcome**

**Pass verdict**

In step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in step 1.
In step 4, the IUT responds with a Config Friend Status message with the Friend field set to 0x01 and sends a Heartbeat message with the Friend bit of the Features field set to 1. The Heartbeat message has the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message received in step 1, and the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message.

In step 6, the IUT responds with a Config Friend Status message with the Friend field set to 0x00 and sends a Heartbeat message with the Friend bit of the Features field set to 0. The Heartbeat message has the DST field and the InitTTL field as in step 4.

In step 8, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in step 7.

In step 10, the IUT responds with a Config Friend Status message with the Friend field set to 0x01 and does not send a Heartbeat message.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.14.5 MESH/NODE/CFG/HBP/BV-05-C [Triggered publishing of Heartbeat messages - Low Power State changes]

• Test Purpose
  Verify that the IUT supporting Low Power feature can send Heartbeat Messages triggered by the Low Power State changes on the node, when configured to do so by a node supporting the Configuration Client Model.

• Reference
  [3] Section 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.17.5, 4.3.2.62, 4.3.2.63

• Initial Condition
  - The IUT has the Low Power feature enabled.
  - The Lower Tester 2 is a node supporting the Friend feature and the feature is enabled.
  - The Lower Tester 1 is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT, the Lower Tester 1 and the Lower Tester 2 share previously established network security credentials.
  - The Lower Tester 1 has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The initial state of the Low Power bit of the Heartbeat Publication Features Values is set (i.e., 0b1).

• Test Procedure
  1. The Upper Tester orders the IUT to establish a friendship with the Lower Tester 2, following the procedure described in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node].
2. The Lower Tester 1 sends a Config Heartbeat Publication Set message to the IUT, with the Destination field set to the Lower Tester 1’s unicast address, the CountLog field set to 0x00, the Features field set to 0x0008 and all the other fields set to valid values.

3. The Upper Tester orders the IUT to send the Friend Poll message with the FSN field set to 0 to the Lower Tester 2.

4. The Lower Tester 2 forwards the Config Heartbeat Publication Set message to the IUT.

5. The Lower Tester 1 expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.

6. The Upper Tester orders the IUT to send four Friend Poll messages with the FSN field set to 1, 0, 1 and 0, respectively.

7. The Lower Tester 2 does not respond to the four Friend Poll messages within the PollTimeout (in Friend Request message) from the IUT, which will cause the friendship to be terminated.

8. The Lower Tester 1 expects the IUT to send a Heartbeat message with the Low Power bit of the Features field set to 0.

9. Repeat step 1 to re-establish the friendship.

10. The Lower Tester 1 expects the IUT to send a Heartbeat message with the Low Power bit of the Features field set to 1.

11. The Lower Tester 1 sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0000 and all the other fields set to valid values.

12. Repeat steps 3 to 4.

13. The Lower Tester 1 expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.


15. The Lower Tester 1 expects the IUT not to send any Heartbeat message.
Figure 4.150: MESH/NODE/CFG/HBP/BV-05-C
• Expected Outcome

Pass verdict

In step 5, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message in step 2.

In step 8, the IUT sends a Heartbeat message with the Low Power bit of the Features field set to 0. The Heartbeat message has the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message in step 2, the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message and the NetKey used is the NetKey identified by the NetKeyIndex used in step 2.

In step 10, the IUT sends a Heartbeat message with the Low Power bit of the Features field set to 1. The Heartbeat message has the DST field and the InitTTL field as in step 2.

In step 13, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message in step 11.

In step 15, the IUT does not send a Heartbeat message.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.14.6 MESH/NODE/CFG/HBP/BV-06-C [Disabling publishing of Heartbeat messages – Config NetKey Deleted]

• Test Purpose

Verify that the IUT disables Heartbeat Messages publishing when NetKey used in Heartbeat Publication is deleted because of the processing of the Config NetKey Delete message.

• Reference

[3] Section 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.17.2, 4.2.17.3, 4.3.2.3.3, 4.3.2.62, 4.4.1.2.9

• Initial Condition

- The Lower Tester is a node supporting the Configuration Client Model.
- The IUT has been provisioned.
- The IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT has a list of two NetKeys with NetKeyIndex of 0 and 1.

• Test Procedure

1. The Lower Tester sets its Heartbeat Subscription Source to IUT’s unicast address and Heartbeat Subscription PeriodLog to 0x11, and then sends a Config Heartbeat Publication Set message with the Destination field set to the Lower Tester’s unicast address, the CountLog field set to 0xFF, the PeriodLog field set to 0x01, the TTL field set to 0x7F, the Features field set to 0x0000, and the NetKeyIndex field set to NetKey Index of the NetKey to the IUT.
2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.

3. The Lower Tester expects the IUT to start sending periodical Heartbeat messages to the Lower Tester's unicast address every second.

4. The Lower Tester deletes the NetKey from IUT's NetKey List, following the procedure described in MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message]. The NetKeyIndex field is an index that identifies the global NetKey Index of the NetKey specified in step 1.

5. The Lower Tester expects the IUT to stop sending periodical Heartbeat messages. Wait 4 seconds, and the Lower Tester does not receive any Heartbeat messages.

6. The Lower Tester sends a Config Heartbeat Publication Set message with the Destination field set to the Lower Tester's unicast address, the CountLog field set to 0xFF, the PeriodLog field set to 0x01, the TTL field set to 0x7F, the Features field set to 0x0000, and the NetKeyIndex field set to NetKey Index of the NetKey just deleted to the IUT.

7. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x04 (Invalid NetKey) and the Destination, CountLog, PeriodLog, and TTL fields set to the values of corresponding fields of the incoming message.

Figure 4.151: MESH/NODE/CFG/HBP/BV-06-C
• Expected Outcome
  
  Pass verdict

In step 3, the IUT starts sending periodical Heartbeat message every second and the NetKey used is the NetKey identified by the NetKeyIndex used in step 1.

In step 5, the IUT stops sending periodical Heartbeat messages.

In step 7, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x04 (Invalid NetKey) and the Destination, CountLog, PeriodLog, and TTL fields set to the values of corresponding fields of the incoming message.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.14.7 MESH/NODE/CFG/HBP/BV-07-C [Triggered publishing of Heartbeat messages – Proxy state changes]

• Test Purpose
  
  Verify that the IUT can send Heartbeat Messages triggered by the Proxy state changes on the node, when configured to do so by a node supporting the Configuration Client Model.

• Reference
  
  [3] Section 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.11, 4.3.2.10, 4.3.2.11, 4.3.2.62, 4.3.2.63, 4.4.1.2.4, 4.4.2.2.4

• Initial Condition
  
  - The IUT supports the Proxy feature and the feature is disabled.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

• Test Procedure
  
  1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0002 and all the other fields set to valid values.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
  3. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01.
  4. The Lower Tester expects the IUT to respond with a Config GATT Proxy Status message with the GATTProxy field set to 0x01 and to send a Heartbeat message with the Proxy bit of the Features field set to 1.
  5. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x00.
  6. The Lower Tester expects the IUT to respond with a Config GATT Proxy Status message with the GATTProxy field set to 0x00 and to send a Heartbeat message with the Proxy bit of the Features field set to 0.
7. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x00.

8. The Lower Tester expects the IUT to respond with a Config GATTProxy Status message with the GATT Proxy field set to 0x00 and not to send any Heartbeat message.

9. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0000 and all the other fields set to valid values.

10. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.

11. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01.

12. The Lower Tester expects the IUT to respond with a Config GATT Proxy Status message with the GATTProxy field set to 0x01 and not to send any Heartbeat message.
Figure 4.152: MESH/NODE/CFG/HBP/BV-07-C
• Expected Outcome
  
  Pass verdict

In step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in step 1.

In step 4, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x01 and a Heartbeat message with the Proxy bit of the Features field set to 1. The Heartbeat message has the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message received in step 1, and the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message. The Opcode field of the Transport Control message is set to 0x0A and the NetKey used is from step 1.

In step 6, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x00 and a Heartbeat message with the Proxy bit of the Features field set to 0. The Heartbeat message has the DST field and the InitTTL field as in step 4.

In step 8, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x00 and does not send a Heartbeat message.

In step 10, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in step 9.

In step 12, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x01 and does not send a Heartbeat message.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.14.8 MESH/NODE/CFG/HBP/BI-01-C [Respond to Config Heartbeat Publication Set – Invalid Publication Features RFU Values]

• Test Purpose
  
  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Heartbeat Publication Set message with invalid Publication Features RFU Values.

• Reference
  
  [3] Section 3.1.2, 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.8, 4.2.17.5, 4.3.1, 4.3.2.13, 4.3.2.14, 4.3.2.62, 4.3.2.63, 4.4.1.2.6, 4.4.2.2.6

• Initial Condition
  
  - The IUT has the Relay mode disabled.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
• Test Procedure

1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT with CountLog set to 0x00, the Features field set to 0xFFF9 (All RFU bits and Relay bit are set, Friend and Low Power bits are not set) and all the other fields set to valid values.

2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00, the Features field set to 0x0001 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.

3. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01.

4. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x01 and to send a Heartbeat message with the Relay bit of the Features field set to 1.

5. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x00.

6. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x00 and to send a Heartbeat message with the Relay bit of the Features field set to 0.

7. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT with CountLog set to 0x00, the Features field set to 0xFFF8 (All RFU bits are set and all Relay, Friend and Low Power bits are not set) and all the other fields set to valid values.

8. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01.

9. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x01, but not to send any Heartbeat message.
**Expected Outcome**

**Pass verdict**

In step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00, the Features field set to 0x0001 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in step 1.

In step 4, the IUT sends a Config Relay Status message with the Relay field set to 0x01 and a Heartbeat message with the Relay bit of the Features field set to 1. The Heartbeat message is published with the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message received in step 1, and the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message.
In step 6, the IUT sends a Config Relay Status message with the Relay field set to 0x00 and a Heartbeat message with the Relay bit of the Features field set to 0. The Heartbeat message has the DST field and the InitTTL field values set as set in step 4.

In step 9, the IUT sends a Config Relay Status message with the Relay field set to 0x01 and does not send a Heartbeat message.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.14.9 MESH/NODE/CFG/HBP/BI-02-C [Respond to Config Heartbeat Publication Get – Wrongly Encrypted by AppKey]

- Test Purpose
  Verify that an IUT can ignore an improperly encrypted Config Heartbeat Publication Get message.

- Reference
  [3] Section 3.7.4.3, 3.7.4.4, 4.3.2.61

- Initial Condition
  - The IUT is a node supporting the Configuration Server Model.
  - The Lower Tester is an node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The AppKey is known to the IUT within the context of the NetKey used.

- Test Procedure
  1. The Lower Tester sends a Config Heartbeat Publication Get message to the IUT. This Configuration message is supposed to be encrypted by IUT’s DevKey, but in this step, the Lower Tester intentionally encrypts the message using the AppKey.
  2. The Lower Tester expects the IUT not to send any response message.

*Figure 4.154: MESH/NODE/CFG/HBP/BI-02-C*

- Expected Outcome
  Pass verdict

In step 2, the IUT ignores the incoming message encrypted by a wrong key and does not send any response message back.
4.15.14.10  MESH/CFGCL/CFG/HBP/BV-01-C [Send Config Heartbeat Publication Set/Get Messages]

• Test Purpose
  Verify that an IUT node supporting the Configuration Client Model can transmit a Config Heartbeat Publication Get/Set message.

• Reference
  [3] Section 3.6.7.1, 3.6.7.2, 4.2.17.1 - 4.2.17.6, 4.3.2.61, 4.3.2.62, 4.4.2.2.15

• Initial Condition
  - The Lower Tester is a node supporting the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials.

• Test Procedure
  1. The Upper Tester orders the IUT to send a Config Heartbeat Publication Set message with the Destination field set to IUT’s unicast address, the CountLog field set to 0x11, the PeriodLog field set to 0x02, the TTL field set to 0x7F, the Features field set to 0x0000, and the NetKeyIndex field set to valid NetKey Index to the Lower Tester.
  2. The Lower Tester expects to receive a Config Heartbeat Publication Set message from the IUT with all the fields set to the values in step 1.
  3. The Lower Tester responds with a Config Heartbeat Publication Status message.
  4. The Upper Tester orders the IUT to send a Config Heartbeat Publication Get message to the Lower Tester.
  5. The Lower Tester expects to receive a Config Heartbeat Publication Get message from the IUT without parameters.
  6. The Lower Tester responds with a Config Heartbeat Publication Status message.
  7. Repeat step 1 and 2 two more times with the Destination field set to the unassigned address, and a dynamically assigned group address.
Figure 4.155: MESH/CFGCL/CFG/HBP/BV-01-C

- **Expected Outcome**

  **Pass verdict**

  In step 2, the IUT sends a Config Heartbeat Publication Set message with all the fields set to the values in step 1.

  In step 4, the IUT sends a Config Heartbeat Publication Get message without parameters.

  In step 5, the IUT sends Config Heartbeat Publication Set message.

  All configuration messages are encrypted and authenticated using the device key of the Lower Tester.
4.15.15 Heartbeat Subscription Procedures

This section is for validating the Heartbeat Subscription procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.15.1 MESH/NODE/CFG/HBS/BV-01-C [Receive Config Heartbeat Subscription Set/Get message]

- **Test Purpose**
  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Heartbeat Subscription Set/Get message, and update the identified Heartbeat Subscription state.

- **Reference**
  [3] Section 3.7.5.2, 4.2.18.3, 4.2.18.4, 4.3.2.64, 4.3.2.65, 4.3.2.66, 4.4.1.2.16

- **Initial Condition**
  - The IUT is a node supporting the Configuration Server Model on the Primary Element.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- **Test Procedure**
  1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to the IUT's unicast address and the PeriodLog field set to 0x11.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the values of the corresponding fields from the received Config Heartbeat Subscription Set message in step 1, and each of the other 4 fields set to 1-octet value (which will be verified in the next test cases.)
  3. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
  4. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same fields as the message received in step 2.
  5. Repeat step 1, but with Destination field set to the unassigned address 0x0000, and the PeriodLog field set to 0x10.
  6. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, the MinHops field set to 0x7F, and the CountLog, PeriodLog and MaxHops fields set to 0x00.
  7. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
  8. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, and the CountLog, PeriodLog, MinHops and MaxHops fields set to 0x00.
  9. Repeat step 1, but with the Source field set to the unassigned address 0x0000, and the PeriodLog field set to 0x10.
  10. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned
address 0x0000, the MinHops field set to 0x7F, and the CountLog, PeriodLog and MaxHops fields are set to 0x00.

11. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
12. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, and the CountLog, PeriodLog, MinHops and MaxHops fields set to 0x00.

![Diagram of MESH/NODE/CFG/HBS/BV-01-C protocol messages](image)
• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the values of the corresponding fields from the received Config Heartbeat Subscription Set message in step 1, and each of the other 4 fields set to 1-octet value.

In step 4, the IUT sends a Config Heartbeat Subscription Status message with the same fields as the message received in step 2.

In steps 6 and 10 the IUT sends a Config Heartbeat Subscription Status message with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, the MinHops field set to 0x7F, and the CountLog, PeriodLog, MinHops and MaxHops fields set to 0x00.

In steps 8 and 12 the IUT sends a Config Heartbeat Subscription Status message with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, and the CountLog, PeriodLog, MinHops and MaxHops fields set to 0x00.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.15.2 MESH/NODE/CFG/HBS/BV-02-C [Receive Heartbeat messages – Controlled by PeriodLog State]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model can receive the periodic Heartbeat messages controlled by Heartbeat Subscription PeriodLog state.

• Reference

[3] Section 3.6.7.2, 4.2.18.1 - 4.2.18.4, 4.3.2.64 - 4.3.2.66, 4.4.1.2.16, 4.4.2.2.16

• Initial Condition

- The Lower Tester is a node supporting the Configuration Server Model and the Configuration Client Model.
- 4.4. The IUT has been provisioned with a NetKey identified by a NetKeyIndex.
- The IUT and the Lower Testers share previously established network security credentials.
- The Lower Testers have access to the Device Key for the IUT defined in the IXIT [6] parameters.

• Test Procedure

1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester’s unicast address, the Destination field set to the IUT’s unicast address, and the PeriodLog field set to 0x00.
2. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the CountLog field is set to 0x00.
3. The Lower Tester sends a Heartbeat message to the IUT with the InitTTL field set to 0x00 and the Features field set to 0x0000.
4. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
5. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the CountLog field is set to 0x00.
6. Repeat step 1, but with the PeriodLog field set to 0x02.
7. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.
8. Within 2 seconds, repeat step 3.
9. After 2 seconds, the Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
10. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x01 and the PeriodLog field set to 0x00.
11. Repeat steps 3 and 4.
12. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x01 and the PeriodLog field set to 0x00.
13. Repeat steps 1-12, using the same address as used in step 1 (for the Destination field) and in step 3 (for the DST field), but in this iteration expect the CountLog field in step 2 to be equal to 0x01.
14. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to the IUT's unicast address, and the PeriodLog field set to 0x10.
15. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.
16. The Lower Tester sends a Heartbeat message to the IUT with the InitTTL field set to 0x00 and the Features field set to 0x000.
17. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
18. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x01.
19. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to the IUT's unicast address, and the PeriodLog field set to 0x00.
20. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x01.
Figure 4.157: MESH/NODE/CFG/HBS/BV-02-C
• Expected Outcome

Pass verdict

In the first iteration through steps 1–12, the following are true:

In step 2, 5, and 7, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.

In steps 10 and 12, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01 and PeriodLog field set to 0x00.

In the second iteration through steps 1–12, the following are true:

In steps 2 and 5, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01.

In step 7, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.

In steps 10 and 12, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01 and PeriodLog field set to 0x00.

In step 15, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.

In steps 18 and 20, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.15.3 MESH/NODE/CFG/HBS/BV-03-C [Not receiving Heartbeat messages – Unassigned or Mismatched SRC field]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model cannot receive the periodic Heartbeat messages if Source/Destination field is set to an unassigned address or Source mismatches with SRC field.

• Reference

[3] Section 3.6.7.2, 4.2.18.1, 4.2.18.2, 4.3.2.64, 4.3.2.65, 4.3.2.66, 4.4.1.2.16, 4.4.2.2.16

• Initial Condition

- The Lower Tester is a node supporting the Configuration Server Model and the Configuration Client Model.
- The IUT has been provisioned.
- The IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
• Test Procedure

1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to an address which is not the Lower Tester’s unicast address, the Destination field set to the IUT’s unicast address, and PeriodLog field is set to 0x02.
2. Within 2 seconds, the Lower Tester sends a Heartbeat message to the IUT with the InitTTL field set to 0x00 and the Features field set to 0x0000.
3. After 2 seconds, the Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
4. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the CountLog field is set to 0x00.
5. Repeat step 1, but with the Source field set to the Lower Tester’s unicast address and the Destination field set to an address which is not the IUT’s unicast address.
6. Repeat steps 2 and 3.
7. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the CountLog field is set to 0x00.
8. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the unassigned address 0x0000, Destination field set to the IUT’s unicast address, and PeriodLog field is set to 0x02.
9. Repeat steps 2 and 3.
10. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the Source and Destination fields are set to the unassigned address, and the values of the CountLog, PeriodLog, MinHops, and MaxHops fields are set to 0x00.
11. Repeat step 8, but with the Source field set to the Lower Tester’s unicast address, and Destination field set to the unassigned address 0x0000.
12. Repeat steps 2 and 3.
13. The Lower Tester expects the IUT to respond with the same Config Heartbeat Subscription Status message as that in step 10.
Figure 4.158: MESH/NODE/CFG/HBS/BV-03-C
• Expected Outcome
  
  Pass verdict

In steps 4 and 7, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.

In steps 10 and 13, the IUT responds with a Config Heartbeat Subscription Status message, and in the message the Source and Destination fields are set to the unassigned address, and the CountLog, PeriodLog, MinHops, and MaxHops fields are set to 0x00.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.15.4 MESH/NODE/CFG/HBS/BV-04-C [Receive Heartbeat messages – CountLog and Hops]

• Test Purpose
  
  Verify that an IUT node supporting the Configuration Server Model can receive the periodic Heartbeat messages and update the state accordingly.

• Reference
  
  [3] Section 3.6.7.2, 4.2.18.3, 4.2.18.4, 4.2.18.5, 4.2.18.6, 4.3.2.64 – 4.3.2.66, 4.4.2.2.15

• Initial Condition
  
  - The Lower Tester is a node supporting the Configuration Server Model and the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

• Test Procedure
  
  1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester’s unicast address, the Destination field set to IUT’s unicast address, and the PeriodLog field is set to 0x00.
  2. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to IUT’s unicast address, and the PeriodLog field is set to 0x11.
  3. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
  4. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message with the CountLog field set to 0x00, the MinHops field set to 0x7F and the MaxHops field set to 0x00.
  5. The Lower Tester sends a Heartbeat message to the IUT with the InitTTL field set to 0x02 and the Features field set to 0x0000.
  6. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
  7. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x01, the MinHops field is set to 0x01 and the MaxHops field is set to 0x01.
  8. Repeat steps 5 and 6.
9. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x02.
10. Repeat steps 5 and 6.
11. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x02.
12. Repeat steps 5 and 6.
13. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x03.
14. Repeat steps 5 and 6.
15. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x03.
Figure 4.159: MESH/NODE/CFG/HBS/BV-04-C
• Expected Outcome

Pass verdict

In step 4, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00, the MinHops field set to 0x7F and the MaxHops field set to 0x00.

In step 7, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01, the MinHops field set to 0x01 and the MaxHops field set to 0x01.

In step 9, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x02.

In step 11, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x02.

In step 13, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x03.

In step 15, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x03.

All configuration messages are encrypted and authenticated using the device key of the IUT.

4.15.15.5 MESH/NODE/CFG/HBS/BV-05-C [Receive Heartbeat messages – Friendship]

• Test Purpose

Verify that an IUT acting as Friend node can cache periodic Heartbeat messages and forward them to its Low Power node if the DST of the messages is Low Power node’s unicast address or a dynamically-assigned group address that has been added to the Friend Subscription List.

• Reference

[3] Section 3.6.5.10, 3.6.7.2, 4.2.13, 4.2.18.1, 4.2.18.2, 4.3.2.65, 4.4.1.2.16, 4.4.2.2.16

• Initial Condition

- The IUT has the Friend feature enabled.
- The Lower Tester 1 is a node supporting the Low Power feature and the feature is enabled.
- The Lower Tester 2 is a node supporting the Configuration Client Model.
- The IUT has been provisioned.
- The IUT and the Lower Testers share previously established network security credentials.
- The Lower Tester 2 has access to the Device Key for the IUT defined in the IXIT [6] parameters.

• Test Procedure

1. The Lower Tester 1 establishes a friendship with the IUT, following the procedure described in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].
2. The Lower Tester 1 sends a Friend Subscription List Add message to the IUT with the AddressList field value containing the Lower Tester 1’s group address and a random TransactionNumber field value.
3. The Lower Tester 1 expects to receive a Friend Subscription List Confirm message with a TransactionNumber field equal to that of the value it sent in step 2 from the IUT.

4. The Lower Tester 2 sends a Config Heartbeat Subscription Set message with the Source field set to the Lower Tester 2’s unicast address, the Destination field set to LPN’s (the Lower Tester 1’s) unicast address, and the PeriodLog field set to 0x11. The DST field of the Network PDU containing this message is set to the unicast address of the Lower Tester 1, therefore the IUT is expected to cache the PDU in its Friend Queue.

5. The Lower Tester 1 sends a Friend Poll message to the IUT with the FSN field toggled.

6. The Lower Tester 1 expects the IUT to forward the message received and cached in step 4.

7. The Lower Tester 2 sends a Heartbeat message, with the DST field set to the unicast address of the LPN, and with the InitTTL field set to 0x7F and the Features field set to 0x0002.

8. The Lower Tester 1 sends a Friend Poll message to the IUT with the FSN field toggled.

9. The Lower Tester 1 expects the IUT to forward the Heartbeat message received and cached in step 7.

10. Repeat steps 4–6, but with the Destination field set to LPN’s (the Lower Tester 1’s) group address.

11. The Lower Tester 2 sends a Heartbeat message to the IUT with the DST field set to the LPN’s group address.

12. The Lower Tester 1 sends a Friend Poll message to the IUT with the FSN field toggled.

13. The Lower Tester 1 expects the IUT to forward the Heartbeat message received and cached in step 11.
Figure 4.160: MESH/NODE/CFG/HBS/BV-05-C

- **Expected Outcome**
  
  **Pass verdict**

  In steps 9 and 13, the IUT forwards the Heartbeat message to the Lower Tester 1.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

### 4.15.15.6 MESH/NODE/CFG/HBS/BI-01-C [Ignore Config Heartbeat Subscription Set Message – Invalid Source Address or Destination Address]

- **Test Purpose**
  
  Verify that an IUT node supporting the Configuration Server Model ignores the Config Heartbeat Subscription Set message if with invalid Source field.

- **Reference**
  
  [3] Section 3.6.7.2, 4.2.18.1, 4.2.18.2, 4.3.2.64, 4.3.2.65, 4.3.2.66, 4.4.1.2.16, 4.4.2.2.16
• **Initial Condition**
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned with a NetKey identified by a NetKeyIndex of 0x000.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

• **Test Procedure**
  1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester’s unicast address, the Destination field set to the IUT’s unicast address, and PeriodLog field set to 0x01.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the Source field and the Destination field are set as set in step 1.
  3. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester’s group address, the Destination field set to the IUT’s unicast address, and PeriodLog field is set to 0x05.
  4. The Lower Tester expects the IUT not to respond with any messages.
  5. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
  6. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in step 2.
  7. Repeat step 3, but with the Source field set to the all nodes address.
  8. The Lower Tester expects the IUT not to respond with any messages.
  9. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
 10. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in step 2.
 11. Repeat step 3, but with the Source field set to a virtual address.
 12. The Lower Tester expects the IUT not to respond with any messages.
 13. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
 14. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in step 2.
 15. Repeat step 1, but with the Destination field set to a virtual address.
 16. The Lower Tester expects the IUT not to respond with any messages.
 17. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
 18. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in step 2.
Figure 4.161: MESH/NODE/CFG/HBS/BI-01-C
• Expected Outcome

  Pass verdict

  In steps 4, 8, 12, and 16 the IUT does not respond with any messages.

  In steps 6, 10, 14 and 18 the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in step 2.

  **4.15.15.7 MESH/CFGCL/CFG/HBS/BV-01-C [Send Config Heartbeat Subscription Set/Get Messages]**

• Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can transmit a Config Heartbeat Subscription Set/Get message.

• Reference

  [3] Section 3.6.7.1, 4.2.18.1, 4.2.18.2, 4.3.2.64, 4.3.2.65, 4.3.2.66, 4.4.2.2.16

• Initial Condition

  - The Lower Tester is a node supporting the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials.

• Test Procedure

  1. The Upper Tester orders the IUT to send a Config Heartbeat Subscription Set message to the Lower Tester with Source field set to the Lower Tester’s unicast address, the Destination field set to IUT’s unicast address, and the PeriodLog field set to 0x11.
  2. The Lower Tester receives the Config Heartbeat Subscription Set message and replies with a Config Heartbeat Subscription Status message.
  3. The Upper Tester orders the IUT to send a Config Heartbeat Subscription Get message to the Lower Tester.
  4. The Lower Tester receives the Config Heartbeat Subscription Get message and replies with a Config Heartbeat Subscription Status message.
  5. Repeat steps 1–2 two times first with the Destination field set to unassigned address 0x0000, then a dynamically-assigned group address.
Figure 4.162: MESH/CFGCL/CFG/HBS/BV-01-C

- **Expected Outcome**

  **Pass verdict**

  In step 2, the IUT sends a Config Heartbeat Subscription Set message with Source field set to the Lower Tester's unicast address, the Destination field set to IUT's unicast address, and the PeriodLog field set to 0x11.

  In step 4, the IUT sends a Config Heartbeat Subscription Get message without parameters.

  In step 5, the IUT sends a Config Heartbeat Subscription Set message with the values specified in step 5.

  All configuration messages are encrypted and authenticated using the device key of the Lower Tester.
4.15.16 Network Transmit Procedures

This section is for validating the Network Transmit procedures for both the Configuration Server Model and the Configuration Client Model.

4.15.16.1 MESH/NODE/CFG/NTX/BV-01-C [Respond to Config Network Transmit Get/Set Messages]

- **Test Purpose**
  
  Verify that an IUT node supporting the Configuration Server Model can respond to Config Network Transmit Get and Config Network Transmit Set messages.

- **Reference**

  [3] Section 3.7.4.3, 4.3.2.69, 4.3.2.70, 4.3.2.71, 4.4.1.2

- **Initial Condition**
  
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester acts as Configuration Client.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.

- **Test Procedure**

  1. The Lower Tester sends a Config Network Transmit Get message to the IUT.
  2. The Lower Tester expects to receive a Config Network Transmit Status message in response from the IUT.
  3. The Lower Tester sends a Config Network Transmit Set message to the IUT with the NetworkTransmitCount and the NetworkTransmitIntervalSteps fields set to values different than the ones received in step 2.
  4. The Lower Tester expects to receive a Config Network Transmit Status message in response from the IUT with the field values equal to the ones sent in step 3.
  5. The Lower Tester sends a Config Network Transmit Get message to the IUT.
  6. The Lower Tester expects to receive a Config Network Transmit Status message in response from the IUT with the field values equal to the ones sent in step 3.

- **Expected Outcome**
  
  **Pass verdict**

  In steps 2, 4, and 6, the IUT sends a Config Network Transmit Status message to the Lower Tester. The field values in steps 4 and 6 are equal to the values requested in step 3.

  All configuration messages are encrypted using the device key of the IUT.

4.15.16.2 MESH/CFGCL/CFG/NTX/BV-01-C [Send Config Network Transmit Get/Set Messages]

- **Test Purpose**

  Verify that an IUT node supporting the Configuration Client Model can send Config Network Transmit Get and Config Network Transmit Set messages to a Configuration Server.
• Reference

[3] Section 3.7.4.3, 4.3.2.69, 4.3.2.70, 4.3.2.71, 4.4.2.2

• Initial Condition

- The IUT supports the Configuration Client Model on the Primary Element.
- The Lower Tester acts as Configuration Server.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure

1. The Upper Tester orders the IUT to send a Config Network Transmit Get message to the Lower Tester.
2. The Lower Tester expects to receive a Config Network Transmit Get message from the IUT.
3. The Lower Tester sends a Config Network Transmit Status message in response to the IUT.
4. The Upper Tester orders the IUT to send a Config Network Transmit Set message to the Lower Tester with random values for the RelayRetransmitCount and RelayRetransmitIntervalSteps fields.
5. The Lower Tester expects to receive a Config Network Transmit Set message from the IUT with the values requested in step 4.
6. The Lower Tester sends a Config Network Transmit Status message in response to the IUT with the values received in step 5.

• Expected Outcome

Pass verdict

In step 2, the IUT sends a Config Network Transmit Get message to the Lower Tester.

In step 5, the IUT sends a Config Network Transmit Set message to the Lower Tester with the requested values.

All configuration messages are encrypted using the device key of the Lower Tester.

4.15.17 Low Power Node PollTimeout Procedures

This section is for validating the Low Power Node PollTimeout procedures for both the Configuration Server Model and the Configuration Client Model. Additional valid behavior tests for the Configuration Server Model are covered by MESH/NODE/FRND/FN/BV-21-C [Friend Node Respond to Config Low Power Node PollTimeout Get].

4.15.17.1 MESH/NODE/CFG/LPNPT/BI-01-C [Ignore Invalid Config Low Power Node PollTimeout Get Messages]

• Test Purpose

Verify that an IUT node supporting the Configuration Server Model ignores invalid Config Low Power Node PollTimeout Get messages.

• Reference

[3] Section 4.3.2.67, 4.3.2.68, 4.4.1.2.17, 4.4.1.2.18, 4.4.2.2.17
Mesh Profile (MESH) / Test Suite

• Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element and the Friend feature.
  - The Lower Tester acts as Configuration Client.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.

• Test Procedure
  1. The Lower Tester sends a Config Low Power Node PollTimeout Get message to the IUT with the LPNAddress field set to the unassigned address.
  2. The Lower Tester expects no response from the IUT.
  3. The Lower Tester repeats steps 1–2 two more times, setting the LPNAddress field to a group address and then to a virtual address.

• Expected Outcome
  Pass verdict

In all iterations through step 2, the IUT does not respond to the invalid messages.

All configuration messages are encrypted using the device key of the IUT.

4.15.17.2 MESH/CFGCL/CFG/LPNPT/BV-01-C [Send Config Low Power Node PollTimeout Get Messages]

• Test Purpose
  Verify that an IUT node supporting the Configuration Client Model can send Config Low Power Node PollTimeout Get messages to a Configuration Server.

• Reference
  [3] Section 4.3.2.67, 4.3.2.68, 4.4.1.2.17, 4.4.1.2.18, 4.4.2.2.17

• Initial Condition
  - The IUT supports the Configuration Client Model on the Primary Element.
  - The Lower Tester acts as Configuration Server.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

• Test Procedure
  1. The Upper Tester orders the IUT to send a Config Low Power Node PollTimeout Get message to the Lower Tester, for a given LPN unicast address.
  2. The Lower Tester expects to receive a Config Low Power Node PollTimeout Get message from the IUT, with the LPNAddress field set to the LPN unicast address requested in step 1, and responds with a Config Low Power Node PollTimeout Status message with valid parameters.
• Expected Outcome
  Pass verdict

In step 2, the IUT sends a Config Low Power Node PollTimeout Get message to the Lower Tester with the requested LPNAddress field value.

All configuration messages are encrypted using the device key of the Lower Tester.

4.16 Health Model
The test group objective is to verify functionality related to both the Health Server Model and the Health Client Model.

4.16.1 Current Fault State
This section is for validating the behavior of the Current Fault State and its messages for both the Health Server Model and the Health Client Model.

4.16.1.1 MESH/SR/HM/CFS/BV-01-C [Send Health Current Status Messages - No Faults]
• Test Purpose
  Verify that an IUT node supporting the Health Server Model can publish unsolicited Health Current Status messages periodically when it has no active faults.

• Reference
  [3] Section 3.7.4.3, 4.3.3.1, 4.3.3.9, 4.4.3.1, 4.4.3.2.1, 4.4.3.2.3

• Initial Condition
  - The IUT’s FaultArray contains no records.
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the Lower Tester (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

• Test Procedure
  1. Execute step 1 through step 6 in test MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages]. In step 5, the Lower Tester used a Publish Period value of 0x48 (8 seconds).
  2. The Lower Tester sends a Health Period Set Unacknowledged message to the IUT with the Fast Period Log field set to 0x02.
  3. The Lower Tester expects the IUT to send an unsolicited Health Current Status message every 8 seconds and no re-transmission.
  4. The Lower Tester sends a Config Model Publication Set message to the IUT with the ElementAddress, Model Identifier, and AppKeyIndex fields set to values stored from step 1, a PublishAddress equal to the address of the Lower Tester, CredentialFlag field set to 0b0, RFU field set to 0b111, Publish TTL field set to a random number between 0x00 and 0x7F, and
Publish Period field set to 0x48, the Publish Retransmit Count field set to 0b001, and the Publish Retransmit Interval Steps field set to 0b00001.

5. The Lower Tester expects the IUT to send an unsolicited Health Current Status message every 8 seconds with one retransmit 100 ms later.

---

**Figure 4.163: MESH/SR/HM/CFS/BV-01-C**

- **Expected Outcome**
  
  **Pass verdict**

  In step 3, the IUT sends an unsolicited Health Current Status message every 8 seconds with the Company ID field set to the CID field of the Composition Data state, and an empty FaultArray field and without re-transmit.

  In step 5, the IUT sends an unsolicited Health Current Status message every 8 seconds with one retransmit 100 ms later

  All Health messages are encrypted with the shared application key bound to the Health Models.

---

**4.16.1.2 MESH/SR/HM/CFS/BV-02-C [Send Health Current Status Messages - With Faults]**

- **Test Purpose**

  Verify that an IUT node supporting the Health Server Model can publish unsolicited Health Current Status messages periodically when it has active faults.

- **Reference**

  [3] Section 3.7.4.3, 4.3.3.1, 4.3.3.9, 4.4.3.1, 4.4.3.2.1, 4.4.3.2.3

- **Initial Condition**

  - The Lower Tester supports the Health Client Model.
  - The IUT’s Publish Period state contains 0x48 (i.e., 8 seconds).
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the Lower Tester (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages]).

- **Test Procedure**
  1. The Lower Tester sends a Health Period Set Unacknowledged message to the IUT with the Fast Period Log field set to 0x02.
  2. The Upper Tester generates some faults on IUT such that the FaultArray field contains records.
  3. The Lower Tester expects the IUT to send an unsolicited Health Current Status message every 2 seconds.

![Diagram](image)

**Figure 4.164: MESH/SR/HM/CFS/BV-02-C**

- **Expected Outcome**
  
  **Pass verdict**

  In step 3, the IUT sends an unsolicited Health Current Status message every 2 seconds with the Company ID field set to the CID field of the Composition Data state, and the FaultArray field set to a sequence of faults representing a sequence of faults in the Current Fault FaultArray.

  All Health messages are encrypted with the shared application key bound to the Health Models.

**4.16.2 Registered Fault State**

This section is for validating the behavior of the Registered Fault State and its messages for both the Health Server Model and the Health Client Model.
4.16.2.1 MESH/SR/HM/RFS/BV-01-C [Respond to Health Fault Get]

- **Test Purpose**
  
  Verify that an IUT node supporting the Health Server Model can respond to Health Fault Get messages encrypted using the AppKey used in the message received.

- **Reference**
  
  [3] Section 3.7.5.2, 4.3.3.2, 4.3.3.7, 4.4.3.1, 4.4.3.2.2

- **Initial Condition**
  
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the Company ID of the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- **Test Procedure**
  
  1. The Lower Tester sends a Health Fault Get message to the IUT, with the Company ID field set to the Company ID declared by the IUT.
  2. The Lower Tester expects the IUT to respond with a Health Fault Status message.

- **Expected Outcome**
  
  **Pass verdict**

  In step 2, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to the ID of the most recently performed test and the FaultArray field containing a sequence of faults representing a sequence of faults in the Registered Fault FaultArray. The destination of the Health Fault Status message is the Lower Tester's unicast address.

  All Health messages are encrypted with the shared application key bound to the Health Models.
4.16.2.2 MESH/SR/HM/RFS/BV-02-C [Respond to Health Fault Clear and Health Fault Clear Unacknowledged Messages]

• Test Purpose
Verify that an IUT node supporting the Health Server Model can respond to Health Fault Clear messages and can clear the Registered Fault state upon receiving Health Fault Clear and Health Fault Clear Unacknowledged messages.

• Reference
[3] Section 3.7.4.3, 4.3.3.3, 4.3.3.4, 4.3.3.7, 4.4.3.1, 4.4.3.2.2

• Initial Condition
- The Lower Tester supports the Health Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
- The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

• Test Procedure
1. The Upper Tester generates some faults on the IUT, such that the IUT’s Registered Fault state is not empty.
2. The Lower Tester sends a Health Fault Get message to the IUT using the Company ID of the IUT.
3. The Lower Tester sends a Health Fault Clear message to the IUT using the Company ID of the IUT.
4. The Lower Tester expects the IUT to respond with a Health Fault Status message.
5. Repeat step 1.
6. The Lower Tester sends a Health Fault Clear Unacknowledged message to the IUT using the Company ID of the IUT.
7. The Lower Tester expects the IUT not to respond with any messages.
8. The Lower Tester sends a Health Fault Get message to the IUT using the Company ID of the IUT.
9. The Lower Tester expects the IUT to respond with a Health Fault Status message.
**Figure 4.166: MESH/SR/HM/RFS/BV-02-C**

- **Expected Outcome**

  **Pass verdict**

  In step 3, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state and an empty FaultArray field.

  In step 7, the IUT does not respond with any messages.

  In step 9, the IUT sends a Health Fault Status message with an empty FaultArray field.

  All Health messages are encrypted with the shared application key bound to the Health Models.

---

**4.16.2.3 MESH/SR/HM/RFS/BV-03-C [Respond to Health Fault Test and Health Fault Test Unacknowledged Messages]**

- **Test Purpose**

  Verify that an IUT node supporting the Health Server Model can respond to Health Fault Test messages and can execute a self-test upon receiving Health Fault Test and Health Fault Test Unacknowledged messages.

- **Reference**

  [3] Section 4.3.3.5, 4.3.3.6, 4.3.3.7, 4.4.3.1, 4.4.3.2.2
- **Initial Condition**
  - The IUT is a Node and has a primary Element.
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- **Test Procedure**
  1. The Lower Tester sends a Health Fault Test message with the Company ID field set to the value of the CID field of the Composition Data state and Test ID field set to 0x00 (Standard test).
  2. The Lower Tester expects the IUT to respond with a Health Fault Status message.
  3. The Lower Tester sends a Health Fault Test Unacknowledged message with the Company ID field set to the value of the CID field of the Composition Data state and Test ID field set to 0x00 (Standard test).

![Diagram showing the test procedure](image)

*Figure 4.167: MESH/SR/HM/RFS/BV-03-C*

- **Expected Outcome**

  **Pass verdict**

  In step 2, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to the ID of the performed test and a FaultArray field containing a sequence of faults representing a sequence of faults in the Registered Fault FaultArray.

  In step 3, the IUT does not respond with any response messages.

  All Health messages are encrypted with the shared application key bound to the Health Models.
4.16.2.4 MESH/SR/HM/RFS/BI-01-C [Ignore Invalid Health Fault Test message]

- **Test Purpose**
  
  Verify that an IUT node supporting the Health Server Model ignores Health Fault Test messages with invalid parameters.

- **Reference**
  
  [3] Section 4.3.3.5, 4.3.3.7, 4.4.1.3, 4.4.3.1, 4.4.3.2.2

- **Initial Condition**
  
  - The IUT is a Node and has a primary Element.
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).
  - The valid Test IDs the IUT supports are defined in IXIT [6].

- **Test Procedure**
  
  1. The Lower Tester sends a Health Fault Test message with the Company ID field set to the value of the CID field of the Composition Data state, but the Test ID doesn’t exist.
  2. The Lower Tester expects the IUT not to respond with any message.
  3. The Lower Tester sends a Health Fault Test message with a valid Test ID, but wrong Company ID.
  4. The Lower Tester expects the IUT not to respond with any message.
  5. The Lower Tester sends a Health Fault Test message with Test ID 0x00 (Standard test) and a valid Company ID.
  6. The Lower Tester expects the IUT to respond with a Health Fault Status message.
• Expected Outcome

**Pass verdict**

In step 2 and step 4, the IUT does not send any response messages.

In step 6, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to 0x00 and a FaultArray field containing an empty list or a sequence of faults representing a sequence of faults in the Registered Fault FaultArray.

All Health messages are encrypted with the shared application key bound to the Health Models.

**4.16.2.5 MESH/SR/HM/RFS/BI-02-C [Ignore Invalid Health Fault Test Unacknowledged message]**

• Test Purpose

Verify that an IUT node supporting the Health Server Model ignores Health Fault Test Unacknowledged messages with invalid parameters.

• Reference

[3] Section 4.3.3.6, 4.4.3.1, 4.4.3.2.2

• Initial Condition

- The IUT is a Node and has a primary Element.
- The Lower Tester supports the Health Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
- The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).
- The valid Test IDs the IUT supports are defined in IXIT [6].

• Test Procedure
  1. The Lower Tester sends a Health Fault Test Unacknowledged message with the Company ID field set to the value of the CID field of the Composition Data state, but the Test ID doesn’t exist.
  2. The Lower Tester expects the IUT not to respond with any messages.
  3. The Lower Tester sends a Health Fault Test Unacknowledged message with a valid Test ID, but wrong Company ID.
  4. The Lower Tester expects the IUT not to respond with any messages.
  5. The Lower Tester sends a Health Fault Test message with a valid Test ID 0x00 (Standard test) and a valid Company ID.
  6. The Lower Tester expects the IUT to respond with a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to 0x00 and a FaultArray field containing an empty list or a sequence of faults representing a sequence of faults in the Registered Fault FaultArray.

![Diagram of test procedure](image)

*Figure 4.169: MESH/SR/HM/RFS/BI-02-C*

• Expected Outcome
  *Pass verdict*

  In step 2 and step 4, the IUT does not send any response messages.
In step 6, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to 0x00 and a FaultArray field containing an empty list or a sequence of faults representing a sequence of faults in the Registered FaultArray.

All Health messages are encrypted with the shared application key bound to the Health Models.

4.16.2.6 MESH/CL/HM/RFS/BV-01-C [Send Health Messages]

• Test Purpose

Verify that an IUT node supporting the Health Client Model can send Health Fault Get, Health Fault Clear, Health Fault Clear Unacknowledged, Health Fault Test, and Health Fault Test Unacknowledged messages to a Health Server.

• Reference

[3] Section 4.3.3.2 – 4.3.3.6, 4.4.4.1, 4.4.4.2.2

• Initial Condition

- The Lower Tester supports the Health Server Model.
- The IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
- The Upper Tester indicates a valid Company ID to the IUT when ordering it to send a message.

• Test Procedure

1. The Upper Tester orders the IUT to send a Health Fault Get message to the Lower Tester.
2. The Lower Tester sends a Health Fault Status message in response to the IUT.
3. The Upper Tester orders the IUT to send a Health Fault Clear message to the Lower Tester.
4. The Lower Tester sends a Health Fault Status message in response to the IUT.
5. The Upper Tester orders the IUT to send a Health Fault Clear Unacknowledged message to the Lower Tester.
6. The Lower Tester does not respond to the Health Fault Clear Unacknowledged message.
7. The Upper Tester orders the IUT to send a Health Fault Test message to the Lower Tester with the Test ID field set to 0x00, and the Company ID field set to the Bluetooth-assigned Company Identifier.
8. The Lower Tester sends a Health Fault Status message in response to the IUT.
9. The Upper Tester orders the IUT to send a Health Fault Test Unacknowledged message to the Lower Tester with the Test ID field set to the 0x00, and the Company ID field set to the Bluetooth-assigned Company Identifier.
10. The Lower Tester does not respond to the Health Fault Test Unacknowledged message.
Figure 4.170: MESH/CL/HM/RFS/BV-01-C

- Expected Outcome

Pass verdict

In step 1, the IUT sends a properly formatted Health Fault Get message with the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

In step 3, the IUT sends a properly formatted Health Fault Clear message with the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

In step 5, the IUT sends a properly formatted Health Fault Clear Unacknowledged message with the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

In step 7, the IUT sends a properly formatted Health Fault Test message with the Test ID field set to 0x00 and the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

In step 9, the IUT sends a properly formatted Health Fault Test Unacknowledged message with the Test ID field set to 0x00 and the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.
All Health messages are encrypted with the shared application key bound to the Health Models.

### 4.16.3 Health Period State

This section is for validating the behavior of the Health Period State and its messages for both the Health Server Model and the Health Client Model.

#### 4.16.3.1 MESH/SR/HM/HPS/BV-01-C [Respond to Health Period Messages]

- **Test Purpose**
  
  Verify that an IUT node supporting the Health Server Model can respond to Health Period Get and Health Period Set messages and can process but not respond to Health Period Set Unacknowledged messages.

- **Reference**
  
  [3] Section 4.3.3.8 – 4.3.3.11, 4.4.3.1, 4.4.3.2.3

- **Initial Condition**
  
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- **Test Procedure**
  
  1. The Lower Tester sends a Health Period Set message with the Fast Period Log field set to 0x02.
  2. The Lower Tester expects the IUT to respond with a Health Period Status message with the Fast Period Log field set to 0x02.
  3. The Lower Tester sends a Health Period Set Unacknowledged message with the Fast Period Log field set to 0x03.
  4. The Lower Tester expects no response from the IUT.
  5. The Lower Tester sends a Health Period Get message to the IUT.
  6. The Lower Tester expects the IUT to respond with a Health Period Status message with the Fast Period Log field set to 0x03.
• **Expected Outcome**

**Pass verdict**

In step 2, the IUT sends a response Health Period Status message with the Fast Period Log field set to 0x02.

In step 4, the IUT does not send any response messages.

In step 6, the IUT sends a response Health Period Status message with the Fast Period Log field set to 0x03.

All Health messages are encrypted with the shared application key bound to the Health Models.

**4.16.3.2 MESH/CL/HM/HPS/BV-01-C [Send Health Period Messages]**

• **Test Purpose**

Verify that an IUT node supporting the Health Client Model can send Health Period Get, Health Period Set and Health Period Set Unacknowledged messages to a Health Server.

• **Reference**

[3] Section 4.3.3.8, 4.3.3.9, 4.3.3.10, 4.3.3.11, 4.4.4.1, 4.4.4.2.3

• **Initial Condition**

- The Lower Tester supports the Health Server Model.
- The IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
- The IUT uses the Device Composition Data of the Lower Tester to store the ElementAddress and Company ID on the Lower Tester (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

• Test Procedure

1. The Upper Tester orders the IUT to send a Health Period Get message to the Lower Tester.
2. The Lower Tester responds with a Health Period Status message.
3. The Upper Tester orders the IUT to send a Health Period Set message to the Lower Tester.
4. The Lower Tester responds with a Health Period Status message.
5. The Upper Tester orders the IUT to send a Health Period Set Unacknowledged message to the Lower Tester.

![Diagram](image)

*Figure 4.172: MESH/CL/HM/HPS/BV-01-C*

• Expected Outcome

**Pass verdict**

In step 1, the IUT sends a properly formatted Health Period Get message.

In step 3, the IUT sends a properly formatted Health Period Set message with the Fast Period Log field set to a 1-octet value.

In step 5, the IUT sends a properly formatted Health Period Set Unacknowledged message with the Fast Period Log field set to a 1-octet value.

All Health messages are encrypted with the shared application key bound to the Health Models.
4.16.4 Attention Timer State

This section is for validating the behavior of the Attention Timer State and its messages for both the Health Server Model and the Health Client Model.

4.16.4.1 MESH/SR/HM/ATS/BV-01-C [Respond to Attention Messages]

- **Test Purpose**
  Verify that an IUT node supporting the Health Server Model can respond to Health Attention Get and Health Attention Set messages and can process but not respond to Health Attention Set Unacknowledged messages.

- **Reference**
  [3] Section 4.2.9, 4.3.3.12 – 4.3.3.15, 4.4.3.1, 4.4.3.2.4

- **Initial Condition**
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Testers share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- **Test Procedure**
  1. The Lower Tester sends a Health Attention Set message with the Attention field set to 0x05.
  2. The Lower Tester expects the IUT to respond with a Health Attention Status message with the Attention value less than or equal to the value received in step 1.
  3. The Lower Tester sends a Health Attention Set Unacknowledged message with the Attention field set to 0x07.
  4. The Lower Tester expects the IUT not to respond with a Health Attention Status message.
  5. The Lower Tester sends a Health Attention Get message to the IUT.
  6. The Lower Tester expects the IUT to respond with a Health Attention Status message with the Attention field equal to or less than the value sent in step 3.
**Expected Outcome**

**Pass verdict**

In step 2, the IUT responds with a Health Attention Status message with the Attention field set to 0x05.

In step 4, the IUT does not respond with any message.

In step 6, the IUT responds with a Health Attention Status message from the IUT with the Attention field set to 0x07.

All Health messages are encrypted with the shared application key bound to the Health Models.

**4.16.4.2 MESH/CL/HM/ATS/BV-01-C [Send Attention Messages]**

**Test Purpose**

Verify that an IUT node supporting the Health Client Model can send Health Attention Get, Health Attention Set, and Health Attention Set Unacknowledged messages to a Health Server.

**Reference**

[3] Section 4.2.9, 4.3.3.12, 4.3.3.13, 4.3.3.14, 4.3.3.15, 4.4.4.1, 4.4.4.2.3

**Initial Condition**

- The Lower Tester supports the Health Server Model.
- The IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.

- The IUT uses the Device Composition Data of the Lower Tester to store the ElementAddress and Company ID on the Lower Tester (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

• Test Procedure

1. The Upper Tester orders the IUT to send a Health Attention Get message to the Lower Tester.
2. The Lower Tester responds with a Health Attention Status message.
3. The Upper Tester orders the IUT to send a Health Attention Set message to the Lower Tester.
4. The Lower Tester responds with a Health Attention Status message with an Attention value less than or equal to the Attention value received in step 3.
5. The Upper Tester orders the IUT to send a Health Attention Set Unacknowledged message to the Lower Tester.

Figure 4.174: MESH/CL/HM/ATS/BV-01-C

• Expected Outcome

Pass verdict

In step 1, the IUT sends a properly formatted Health Attention Get message with no parameters.

In step 3, the IUT sends a properly formatted Health Attention Set message.

In step 5, the IUT sends a properly formatted Health Attention Set Unacknowledged message.

All Health messages are encrypted with the shared application key bound to the Health Models.
## 5 Test Case Mapping

The Test Case Mapping Table (TCMT) maps test cases to specific capabilities in the ICS. Profiles, protocols, and services may define multiple roles, and it is possible that a product may implement more than one role. The product shall be tested in all roles for which support is declared in the ICS document.

The columns for the TCMT are defined as follows:

**Item:** contains a y/x reference, where y corresponds to the table number and x corresponds to the feature number as defined in the ICS Proforma for the Mesh Profile specification [4]. If the item is defined with Protocol, Profile or Service abbreviation before y/x, the table and feature number referenced are defined in the abbreviated ICS proforma document.

**Feature:** recommended to be the primary feature defined in the ICS being tested or may be the test case name.

**Test Case(s):** the applicable test case identifiers required for Bluetooth Qualification if the corresponding y/x references defined in the Item column are supported.

For purpose and structure of the ICS/IXIT proforma and instructions for completing the ICS/IXIT proforma refer to the Bluetooth ICS and IXIT proforma document.

<table>
<thead>
<tr>
<th>Item</th>
<th>Feature</th>
<th>Test Case(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESH 4/1</td>
<td>PB-ADV – Node</td>
<td>MESH/NODE/PBADV/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/PBADV/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/PBADV/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/PBADV/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/PBADV/BI-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/PBADV/BI-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/PBADV/BI-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/PBADV/BI-04-C</td>
</tr>
<tr>
<td>MESH 18/2</td>
<td>PB-ADV – Provisioner</td>
<td>MESH/PVNR/PBADV/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/PVNR/PBADV/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/PVNR/PBADV/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/PVNR/PBADV/BI-01-C</td>
</tr>
<tr>
<td>MESH 4/2</td>
<td>PB-GATT – Node</td>
<td>MESH/NODE/MPS/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/MPS/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/MPS/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/MPS/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/MPS/BV-05-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/MPS/BV-06-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/MPS/BV-07-C</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>MESH 18/5</td>
<td>PB-GATT – Provisioner</td>
<td>MESH/NODE/PROV/BV-01-C</td>
</tr>
<tr>
<td>MESH 4/8 AND MESH 4/11</td>
<td>Provisioning: Public Key No OOB – Node, Provisioning: Authentication Static OOB</td>
<td>MESH/NODE/PROV/BV-03-C</td>
</tr>
<tr>
<td>MESH 4/8 AND MESH 4/12</td>
<td>Provisioning: Public Key No OOB – Node,</td>
<td>MESH/NODE/PROV/BV-07-C</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>Provisioning: Authentication No OOB – Node</td>
<td></td>
</tr>
<tr>
<td>MESH 4/7 AND MESH 4/12</td>
<td>Provisioning: Public Key OOB – Node, Provisioning: Authentication No OOB – Node</td>
<td>MESH/NODE/PROV/BV-08-C</td>
</tr>
<tr>
<td>MESH 4/6 AND MESH 0a/1</td>
<td>Erratum 10395 for Provisioning Server</td>
<td>MESH/NODE/PROV/BV-12-C, MESH/NODE/PROV/BI-13-C</td>
</tr>
<tr>
<td>MESH 18/4 AND MESH 0a/1</td>
<td>Erratum 10395 for Provisioning Client</td>
<td>MESH/PVNR/PROV/BI-14-C</td>
</tr>
<tr>
<td>MESH 10/1</td>
<td>Secure Network Beacons</td>
<td>MESH/NODE/BCN/SNB/BV-01-C</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MESH 9/1</td>
<td>Message Replay Protection</td>
<td>MESH/NODE/TNPT/BV-13-C</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>MESH 10/4</td>
<td>IV Index Recovery Procedure</td>
<td>MESH/NODE/IVU/BV-03-C</td>
</tr>
<tr>
<td>MESH 10/2</td>
<td>Key Refresh Procedure</td>
<td>MESH/NODE/KR/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/KR/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/KR/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/KR/BI-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/KR/BI-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/KR/BI-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/KR/BI-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/KR/BI-05-C</td>
</tr>
<tr>
<td>MESH 13/2</td>
<td>Mesh Proxy Service</td>
<td>MESH/SR/MPXS/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/MPXS/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/MPXS/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/MPXS/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/MPXS/BV-05-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/MPXS/BV-06-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/MPXS/BV-07-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/MPXS/BV-09-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/HBP/BV-07-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/GPXY/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/NET/BV-11-C</td>
</tr>
<tr>
<td>MESH 11/1</td>
<td>Configuration Server Model</td>
<td>MESH/NODE/CFG/SNBP/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SNBP/BI-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/COMP/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/DTTL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/DTTL/BI-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MP/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MP/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MP/BI-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MP/BI-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MP/BI-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BV-02-C</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-05-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-06-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-07-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-08-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-09-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/SL/BL-10-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/NKL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/NKL/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/NKL/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/NKL/BL-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/NKL/BL-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/NKL/BL-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/AKL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/AKL/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/AKL/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/AKL/BL-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/AKL/BL-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/AKL/BL-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/AKL/BL-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MAKL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MAKL/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MAKL/BL-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MAKL/BL-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MAKL/BL-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/MAKL/BL-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/RST/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/GPXY/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/GPXY/BL-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/NTX/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/CFGF/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/CFGF/BL-01-C</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/CFGR/BV-01-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/CFGR/BI-01-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBP/BV-01-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBP/BV-02-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBP/BI-02-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBP/BV-06-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBS/BV-01-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBS/BV-02-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBS/BV-03-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBS/BV-04-C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/CFG/HBS/BI-01-C</td>
<td></td>
</tr>
<tr>
<td>MESH 11/1 AND MESH 5/4</td>
<td>Configuration Server Model</td>
<td>MESH/NODE/CFG/NKL/BV-03-C</td>
</tr>
<tr>
<td></td>
<td>Multiple Subnet Support</td>
<td>MESH/NODE/CFG/AKL/BI-04-C</td>
</tr>
<tr>
<td>MESH 11/3</td>
<td>Configuration Client Model</td>
<td>MESH/CFGCL/KR/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/KR/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/KR/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/KR/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/SNBP/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/COMP/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/DTTL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/SL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/SL/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/NKL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/NKL/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/NKL/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/AKL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/AKL/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/AKL/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/AKL/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/MAKL/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/MAKL/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/MAKL/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/MAKL/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/NID/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/RST/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CFGCL/CFG/NTX/BV-01-C</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>MESH/NODE/FRND/FN/BV-</td>
<td>MESH/NODE/FRND/FN/BV-12-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-13-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-14-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-15-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-16-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-17-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-18-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-19-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-20-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BV-21-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BI-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BI-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/FN/BI-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/HBP/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/HBS/BV-05-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/LPNPT/BI-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/NET/BV-12-C</td>
</tr>
<tr>
<td>MESH 7/4</td>
<td>Low Power Feature</td>
<td>MESH/NODE/FRND/LPN/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BV-05-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BV-06-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BV-07-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BV-08-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BI-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BI-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/FRND/LPN/BI-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/HBP/BV-05-C</td>
</tr>
<tr>
<td>MESH 12/1</td>
<td>Proxy Server</td>
<td>MESH/SR/PROX/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/PROX/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/PROX/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/PROX/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/PROX/BV-05-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/PROX/BV-06-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/SR/PROX/BV-07-C</td>
</tr>
<tr>
<td>Item</td>
<td>Feature</td>
<td>Test Case(s)</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MESH 12/5</td>
<td>Proxy Client</td>
<td>MESH/CL/MPXS/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/MPXS/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/MPXS/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/MPXS/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/MPXS/BV-05-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/MPXS/BV-06-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-01-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-05-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-06-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-07-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-08-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-09-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BV-10-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/CL/PROX/BI-01-C</td>
</tr>
</tbody>
</table>

*Table 5.1: Test Case Mapping*
# 6 Revision History and Contributors

## Revision History

<table>
<thead>
<tr>
<th>Revision History</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>D05r00</td>
<td>2015-Nov-19</td>
<td>Initial draft for review.</td>
</tr>
<tr>
<td>D05R01</td>
<td>2015-Nov-26</td>
<td>Added Future Bit test cases.</td>
</tr>
<tr>
<td>D05R02</td>
<td>2015-Dec-02</td>
<td>Fixed formatting issues, table of contents. Referenced specification r34.</td>
</tr>
<tr>
<td>D05R03</td>
<td>2016-Jan-19</td>
<td>Updated test cases for consistency with specification r37. Added new network layer test case. Added new configuration model test cases (client and server). Added new ping model test cases.</td>
</tr>
<tr>
<td>D05R04</td>
<td>2016-Feb-25</td>
<td>Added PB-MESH test cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added transport layer test cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added Smart Mesh GATT Services (Provisioning and Proxy Service) test purposes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added component address testing in MGAP tests</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added virtual addresses to test cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added negative test cases to MGAP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated test cases for D07r00 specification</td>
</tr>
<tr>
<td>D07r01</td>
<td>2016-Mar-11</td>
<td>Synchronized the document with the spec revision 09r00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated the test spec revision to 07r01</td>
</tr>
<tr>
<td>D07r02</td>
<td>2016-Mar-30</td>
<td>Converted to new Test Case ID conventions as defined in TSTO v4.1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated the test spec revision to 07r02</td>
</tr>
<tr>
<td>D09r00</td>
<td>2016-Apr-15</td>
<td>Updated to match Mesh spec D09r00-draft3. Moved several tests from Configuration Model and Network Management Model sections into new sections for the Friendship and Provisioning models. NMM section removed.</td>
</tr>
<tr>
<td>D09r01a</td>
<td>2016-May-08</td>
<td>Added over 100 test cases new test cases for missing functionalities</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated all existing test cases to Synchronized with Mesh spec D09r02.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Editorial changes, correcting numbering, font, spacing, formatting and spelling related issues.</td>
</tr>
<tr>
<td>D09r02</td>
<td>2016-May-10</td>
<td>Correct the contents and resolved formatting issues of MESH/NODE/NET/RLY/BV-10-C through MESH/NODE/NET/RLY/BV-14-C. Update several test cases for the REL bit removal and related message name changes</td>
</tr>
<tr>
<td>Revision History</td>
<td>Date</td>
<td>Comments</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update Ping test cases for message name and field changes</td>
</tr>
<tr>
<td>D09r03</td>
<td>2016-May-31</td>
<td>Add server and client test cases for Node Discoverable Get and Node Discoverable Set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corrected MESH/NODE/PROV/PVN/BV-03-C test case heading by removing the “or No OOB Authentication” which is covered in test case MESH/NODE/PROV/PVN/BV-08-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Deleted PB-Mesh Provisioning related test cases, since it is moved to Mesh 1.1 or future releases.</td>
</tr>
<tr>
<td>D09r04</td>
<td>2016-Jun-08</td>
<td>Add test purposes and expand existing tests to cover shall statements in Mesh spec d09r02-LBL04.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Filled out TCMT.</td>
</tr>
<tr>
<td>D09r05</td>
<td>2016-Jul-15</td>
<td>Updated for Bucharest IOP priorities against Mesh spec d09r09.</td>
</tr>
<tr>
<td>D09r06</td>
<td>2016-Aug-12</td>
<td>Updated for Suzhou IOP priorities against Mesh spec d09r09.</td>
</tr>
<tr>
<td>D09r07</td>
<td>2016-Sep-12</td>
<td>Synchronized the Madrid IOP priorities features test cases (Friendship, Key Refresh, IV Update, SAR, PB-ADV Provisioning, Proxy Protocol, Heartbeat, Health Model, Mesh GATT Proxy Service) with Mesh spec d09r11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Created test cases for new features Proxy Protocol, Health Model, Heartbeat.</td>
</tr>
<tr>
<td>D09r08</td>
<td>2016-Sep-20</td>
<td>Reorganize Test Groups per Madrid IOP to sync up with Mesh specification organization as of Mesh spec d09r11.</td>
</tr>
<tr>
<td>D09r09</td>
<td>2016-10-08</td>
<td>Synchronized the following features test cases with Mesh Spec d09r12 draft 1:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.2 - ADV Provisioning Bearer, 4.3 - GATT Provisioning Bearer, 4.5 - Provisioning Procedure, 4.6 - Network Layer, 4.8 – Transport Layer, 4.9 – IV Index, 4.10 – Key Refresh, 4.13 – Proxy Protocol, 4.15.14 - Heartbeat Publication Procedures, and 4.15.15 – Heartbeat Subscription Procedures</td>
</tr>
<tr>
<td>D09r11</td>
<td>2016-11-06</td>
<td>Extensive updates throughout to address feedback from BTI, Mesh WG, and PTS reviews.</td>
</tr>
<tr>
<td>D09r12</td>
<td>2016-12-08</td>
<td>Synchronized the following sections test case with r13 plus accepted errata</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.3.10 - 4.3.12 - BI and Future Proofing - ADV Provisioning Bearer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.4 - Mesh Provisioning Service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.5.9 and 4.5.10 - BI and Future Proofing - Provisioning Procedure</td>
</tr>
<tr>
<td>Revision History</td>
<td>Date</td>
<td>Comments</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.8 - Message Relaying</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.15 - Mesh Proxy Service</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.16.1 - 4.16.13 - Configuration Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.17.1 - 4.17.4 - Health Model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolved WG member and Boston IOP comments</td>
</tr>
<tr>
<td>0.9.0r13</td>
<td>2017-01-24</td>
<td>SIG staff review. Minor editorial updates. Template conversion.</td>
</tr>
<tr>
<td>0.9.0r14</td>
<td>2017-01-25</td>
<td>Resolved BTI, WG and several round of PTS comments.</td>
</tr>
<tr>
<td>0.9.0r15</td>
<td>2017-02-10</td>
<td>Synchronized with Mesh spec d09r20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolved BTI, WG and PTS comments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added MSCs for some of the Configuration Model test cases</td>
</tr>
<tr>
<td>0.9.0r16</td>
<td>2017-03-16</td>
<td>Synchronized with Mesh spec d09r23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolved IOP, WG and PTS comments</td>
</tr>
<tr>
<td>0.9.0r17</td>
<td>2017-04-12</td>
<td>Synchronized with Mesh spec d09r24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolved IOP, WG and PTS comments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Changed “Mesh” to “Mesh Profile” where appropriate</td>
</tr>
<tr>
<td>0.9.0r18</td>
<td>2017-05-10</td>
<td>Synchronized with Mesh spec d09r26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resolved open TS errata</td>
</tr>
<tr>
<td>0.9.0r19</td>
<td>2017-05-26</td>
<td>Updates during the final Formal IOP (Espoo). Further editorial updates: message renaming, decoupling Provisioner from Configuration Client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added new test case MESH/NODE/PROV/UPD/BV-11-C. Removed test case MESH/PVNR/IVU/BV-01-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Renumbered TCIDs after removing the paragraphs of previously deleted test cases:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/IVU/BV-04-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/NKL/BI-02-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/NODE/CFG/AKL/BI-03-C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Updated TCIDs by replacing:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/PVNR/KR -&gt; MESH/CFGCL/KR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/PVNR/CFG -&gt; MESH/CFGCL/CFG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESH/PVNR/HM -&gt; MESH/CL/HM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FTMO -&gt; LPNPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In sync with Mesh Profile spec d09r28.</td>
</tr>
<tr>
<td>1.0.0r01</td>
<td>2017-06-22</td>
<td>Fixes based on BTI feedback and feedback from the latest test events and the PTS team.</td>
</tr>
<tr>
<td>Revision History</td>
<td>Date</td>
<td>Comments</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1.0.0</td>
<td>2017-07-12</td>
<td>Approved by BTI. Prepared for TCRL 2017-1 publication.</td>
</tr>
<tr>
<td>1.0.1r00</td>
<td>2017-08-25</td>
<td>TSE 9589: Deleted test case MESH/NODE/CFG/NID/BV-02-C from the TS and TCMT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 9626: Revised MESH/NODE/PROV/BV-01-C test purpose.</td>
</tr>
<tr>
<td>1.0.1r01</td>
<td>2017-08-28</td>
<td>TSE 9716: Removed MESH/NODE/IVU/BV-03-C from text and the TCMT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 9741: Clarified test case MESH/NODE/IVU/BV-05-C and updated pass verdict step numbers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 9742: Corrected the test procedure calculation in MESH/NODE/CFG/HBP/BV-02-C.</td>
</tr>
<tr>
<td>1.0.1r02</td>
<td>2017-09-05</td>
<td>TSE 9746: Updated pass verdict step 4 for MESH/NODE/KR/BV-01-C.</td>
</tr>
<tr>
<td>1.0.1r03</td>
<td>2017-09-14</td>
<td>TSE 9795: Revised text and figure for test case MESH/NODE/CFG/HBS/BV-05-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 9750: Revised text and figure for test case MESH/NODE/CFG/HBS/BV-01-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 9714: Removed test case MESH/NODE/BCN/SNB/BV-02-C.</td>
</tr>
<tr>
<td>1.0.1r04</td>
<td>2017-09-15</td>
<td>TSE 9774: Revised text and figure for test case MESH/NODE/FRND/LPN/BV-02-C.</td>
</tr>
<tr>
<td>1.0.1r05</td>
<td>2017-09-29</td>
<td>TSE 9760: Revised text and figure for test case MESH/NODE/FRND/FN/BV-20-C.</td>
</tr>
<tr>
<td>1.0.1r06</td>
<td>2017-10-02</td>
<td>TSE 9824: Added missing friend enables to test cases: MESH/NODE/FRND/FN/BV-01-C – 21-C, and MESH/NODE/FRND/FN/BV-01-C – 03-C.</td>
</tr>
<tr>
<td>1.0.1</td>
<td>2017-11-22</td>
<td>Approved by BTI. Prepared for TCRL 2017-2 publication.</td>
</tr>
<tr>
<td>Revision History</td>
<td>Date</td>
<td>Comments</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 10009 (rating 3): Added fail verdict to all Friend node BV test cases (MESH/NODE/FRND/FN/BV-01-C to 21-C) verifying that a Friend node uses a single Network PDU to send a message to its LPN.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 9905 (rating 3): Revised initial condition and test procedure for MESH/SR/PROX/BV-12-C and 14-C, and replaced MESH/SR/PROX/BV-14-C MSC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 9864 (rating 4): Removed test case MESH/PVNR/MPS/BV-07-C from text and TCMT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 10279 (rating 1): Corrected test procedure step 1 for MESH/NODE/FRND/FN/BV-13-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 10461 (rating 3): Added clarification to MESH/NODE/TNPT/BV-05-C initial condition.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 10330 (rating 3): Revised test procedure step 13, MSC, and expected outcome pass verdict for MESH/NODE/CFG/HBS/BV-02-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 10407 (rating 3): Corrected the expectation in the MESH/NODE/CFG/MP/BV-01-C test procedure steps 14 and 16. Replaced the test procedure MSC and updated the Pass Verdict.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 10286 (rating 3): Removed OOB from the MESH/PVNR/PROV/PVN/BV-10-C test procedure steps 3 and 4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E10395: Added 3 new test cases to TS and TCMT: MESH/NODE/PROV/UPD/BV-12-C, MESH/NODE/PROV/UPD/BV-13-C, and MESH/PVNR/PROV/UPD/BI-14-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corrected implementation of TSE 10330 by changing “a group address” to “the same address” in step 13.</td>
</tr>
<tr>
<td></td>
<td>1.0.2</td>
<td>2018-06-27</td>
</tr>
<tr>
<td></td>
<td>1.0.3r00-r03</td>
<td>2018-10-02 - 2018-11-05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 11166 (rating 3): Updated initial condition for test cases MESH/SR/MPXS/BV-09-C and MESH/NODE/CFG/NID/BV-02-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 11167 (rating 3): Updated test purpose, test procedure steps, pass verdict for test case MESH/NODE/PBADV/BV-02-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Editorial correction: Stray &quot;TBD&quot; removed from pass verdict MESH/NODE/BCN/SNB/BV-01-C.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TSE 11230 (rating 1): Updated TCIDs for the majority of the test cases in section 4.5 Provisioning Procedure to match the correct format, by removing particles /UPD/ and /PVN/. Moved the two invalid key BI tests at the end of 4.5.</td>
</tr>
<tr>
<td>Revision History</td>
<td>Date</td>
<td>Comments</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1.0.1.0</td>
<td>2018-11-09</td>
<td>Updated version number from 1.0.3 to 1.0.1.0 to align with adoption of specification version 1.0.1.0.</td>
</tr>
<tr>
<td>1.0.1.0</td>
<td>2018-11-21</td>
<td>Approved by BTI. Prepared for TCRL 2018-2 publication.</td>
</tr>
</tbody>
</table>
| 1.0.1.1r00–r05   | 2019-04-04–2019-06-19 | TSE 11180 (rating 2): Updated step 13 in test case MESH/NODE/CFG/HBS/BV-02-C so that the exception covers only step 2.  
TSE 11250 (rating 3): Updated to new test case figure for MESH/NODE/CFG/HBS/BV-02-C.  
TSE 11123 (rating 3): Updated test cases MESH/NODE/TNPT/BV-07-C and -08-C to enhance transport tests.  
TSE 11445 (rating 4): Deleted test case MESH/SR/MPXS/BV-08-C and updated TCMT accordingly.  
TSE 11760 (rating 2): Updated initial condition and/or test steps and/or pass verdict in test cases MESH/NODE/KR/BV-01-C -03-C; MESH/NODE/KR/BV-05-C; MESH/NODE/CFG/NKL/BV-01-C, -02-C, and -04-C; MESH/NODE/CFG/NKL/BI-01-C -03-C; MESH/NODE/CFG/AKL/BV-01-C -03-C; and MESH/NODE/CFG/AKL/BV-02-C and -04-C. Updated TCMT accordingly. |
| 1.0.1.1          | 2019-07-25     | Approved by BTI. Prepared for TCRL 2019-1 publication.                                        |

**Contributors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>Josh Toole</td>
<td>Cloud2GND</td>
</tr>
<tr>
<td>Jawid Mirani</td>
<td>Cloud2GND</td>
</tr>
<tr>
<td>Jeff Drake</td>
<td>Cloud2GND</td>
</tr>
<tr>
<td>Nathaniel Roby</td>
<td>Cloud2GND</td>
</tr>
<tr>
<td>Christopher Badder</td>
<td>Cloud2GND</td>
</tr>
<tr>
<td>Jim Harper</td>
<td>Cloud2GND</td>
</tr>
<tr>
<td>Bogdan Alexandru</td>
<td>Cloud2GND</td>
</tr>
<tr>
<td>Alex Andreescu</td>
<td>Cloud2GND</td>
</tr>
</tbody>
</table>