

THREAD GROUP

1

2 Thread Specification

3 This document was approved as the Thread 1.1.1 Specification by the Thread Group, Inc.
4 Board of Directors at the conclusion of the Specification Review Period. Please refer to the
5 Thread Group INTELLECTUAL PROPERTY AND CONFIDENTIALITY POLICY in its entirety for
6 further details.

7 Revision History

Revision	Date	Comments
1.1.1	February 7, 2017	Addresses errata from the 1.1.0 specification.
1.1.0	July 19, 2016	Adds new features for updating the network operational datasets and addresses errata from the 1.0 and 1.0.1 specifications. Approved by the Thread Board of Directors.
1.0.1	January 27, 2016	Approved by the Board to address errata from the 1.0 specification.
1.0	October 29, 2015	Approved by the Thread Board of Directors.
1.0	July 13, 2015	Initial release

8

1 Notice is hereby given that this document is the Confidential Information of The Thread Group and its
2 use is subject to Thread Group Bylaws, Participation Agreement, and the Intellectual Property and
3 Confidentiality Policy. It may not be shared or distributed (in part or in whole) with any entity that is
4 not a Thread Group Participant without express written permission. Failure to abide by this notice
5 constitutes a violation of the Thread Group Bylaws, Participation Agreement, and the Intellectual
6 Property and Confidentiality Policy.

7 Elements of Thread Group specifications may be subject to third party intellectual property rights,
8 including without limitation, patent, copyright or trademark rights (such a third party may or may not
9 be a member of Thread Group). The Thread Group is not responsible and shall not be held responsible
10 in any manner for identifying or failing to identify any or all such third party intellectual property rights.

11 This document and the information contained herein is provided on an "AS IS" basis and THE THREAD
12 GROUP DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO (A) ANY
13 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF THIRD
14 PARTIES (INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING
15 PATENT, COPYRIGHT OR TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF
16 MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NONINFRINGEMENT.

17 IN NO EVENT WILL THE THREAD GROUP BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS,
18 LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL
19 OR EXEMPLARY, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND, IN CONTRACT
20 OR IN TORT, IN CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN,
21 EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

22 Copyright © 2015–2017 Thread Group, Inc. All Rights Reserved.

1 **Acknowledgements**

2 **Authors and Main Contributors**

3 Robert Alexander, Sorin Aliciuc, Skip Ashton, Cristian Cotiga, Thomas Cuyckens, Robert
4 Cragie, Esko Dijk, Grant Erickson, Cosmin Grumei, Jonathan Hui, Andrei Istodorescu,
5 Richard Kelsey, Alin Lazar, Katariina Liehu, Jay Logue, Roxana Motoi, Vlad Neamu, Matteo
6 Paris, Suvesh Pratapa, Marius Preda, Timothy Rosa, Claudia Rosu, Nate Smith, Thad Smith,
7 Andreea Tecuceanu, Mika Tervonen, Martin Turon, Mihai Urzica, Ovidiu Usturoi, James
8 Woodyatt

9 **Recognition**

10 We would like to recognize contributions from:

11 Silviu Barbulescu, Wojciech Bober, Chris Boross, George Capraru, Shu Chen, Sagar
12 Chinchani, Kenneth Coley, Alan Collins, Antonio Concio, Mick Conley, Catalina Costin, Marius
13 Doroftei, Michael Dow, Mihai Dragnea, Lukasz Duda, Constantin Enascuta, Andrei
14 Ergelegiu, Dave Fiore, Tero Heinonen, Juha Heiskanen, Edward Hill, Mihai Ignat, Ryan
15 Kelly, Arto Kinnunen, Navin Kochar, Vaas Krishnamurthy, Vinoth Kumar, Yuwen Lan,
16 Krzysztof Loska, John Loukota, Robert Lubos, Xiao Ma, Jeff Mabert, Bob MacDonald, Philip
17 MacKenzie, Don Markuson, Hubert Mis, Eduardo Montoya, Paul Moroney, Sujata Neidig,
18 Robert Newberry, Elvis Nica, Adrian Nicolau, Juan Carlos Pacheco, Dragos Parausanu, Kartik
19 Pasupathy, Emmanuel Pauchard, Jani Pellikka, Robert Power, Juhani Puurula, Robert
20 Quattlebaum, Christopher Rivera, Kevin Sakuma, Tom Sciorilli, Anthony Scocco, Anand
21 Shah, Arjuna Siva, Jacopo Soffritti, Daniel Soper, Tudor Stanescu, Rongli Sun, Robert
22 Szewczyk, Marcin Szczodrak, Alexandra Tatarascu, Keith Tilley, Kevin Traylor, Steven
23 Urbanski, Daniel Vasilescu, Jussi Vatjus-Anttila, Deepak Venugopal

1 **Contents**

2	Chapter 1 Overview	1-1
3	1.1 Purpose	1-2
4	1.2 Scope and Organization	1-2
5	1.3 Overview	1-3
6	1.3.1 No Single Point of Failure	1-3
7	1.3.2 Device Types	1-4
8	1.3.2.1 Border Routers	1-4
9	1.3.2.2 Routers	1-4
10	1.3.2.3 Router-Eligible End Devices	1-4
11	1.3.2.4 Sleepy End Devices	1-4
12	1.3.3 Security	1-4
13	1.3.3.1 Authentication and Key Agreement.....	1-4
14	1.3.3.2 Network-wide Key	1-5
15	Chapter 2 Supporting Information.....	2-1
16	2.1 Requirements Language.....	2-2
17	2.2 Normative References.....	2-2
18	2.3 Informative References.....	2-5
19	2.4 Terms and Definitions.....	2-5
20	2.5 Nomenclature for Tables Referencing Other Standards.....	2-9
21	2.6 Acronyms.....	2-9
22	Chapter 3 PHY/MAC/6LoWPAN	3-1
23	3.1 Physical Layer.....	3-2
24	3.2 MAC Layer	3-2
25	3.3 6LoWPAN Adaptation Layer	3-10
26	3.4 Multiple Interfaces	3-13
27	Chapter 4 Mesh Link Establishment.....	4-1
28	4.1 Introduction	4-3
29	4.2 Overview	4-3
30	4.2.1 Link Configuration	4-3
31	4.2.2 Parameter Dissemination	4-4
32	4.2.3 Neighbor Detection.....	4-4
33	4.3 Security Formats.....	4-4

1	4.4	Command Format	4-5
2	4.5	TLV Formats.....	4-6
3	4.5.1	Source Address TLV.....	4-7
4	4.5.2	Mode TLV	4-7
5	4.5.3	Timeout TLV	4-9
6	4.5.4	Challenge TLV.....	4-9
7	4.5.5	Response TLV	4-9
8	4.5.6	Link-layer Frame Counter TLV.....	4-9
9	4.5.7	Link Quality TLV.....	4-9
10	4.5.8	Network Parameter	4-10
11	4.5.9	MLE Frame Counter TLV.....	4-10
12	4.5.10	Route64 TLV	4-10
13	4.5.11	Address16 TLV.....	4-10
14	4.5.12	Leader Data TLV	4-10
15	4.5.13	Network Data TLV	4-11
16	4.5.14	TLV Request TLV	4-11
17	4.5.15	Scan Mask TLV.....	4-11
18	4.5.16	Connectivity TLV	4-11
19	4.5.17	Link Margin TLV	4-13
20	4.5.18	Status TLV	4-13
21	4.5.19	Version TLV	4-13
22	4.5.20	Address Registration TLV.....	4-13
23	4.5.21	Channel TLV	4-14
24	4.5.21.1	Supported Channel Pages.....	4-15
25	4.5.21.2	Supported Channels	4-15
26	4.5.22	PAN ID TLV	4-15
27	4.5.23	Active Timestamp TLV	4-16
28	4.5.24	Pending Timestamp TLV.....	4-16
29	4.5.25	Active Operational Dataset TLV	4-16
30	4.5.26	Pending Operational Dataset TLV	4-16
31	4.5.27	Thread Discovery TLV	4-17
32	4.6	Leader and Network Data.....	4-17
33	4.7	Network Attaching.....	4-17
34	4.7.1	Attaching to a Parent	4-17
35	4.7.2	Parent Selection.....	4-21
36	4.7.3	Child Update Request and Child Update Response Messages.....	4-21
37	4.7.4	Message Buffering for Children.....	4-22
38	4.7.5	Timing Out Children.....	4-22
39	4.7.6	Child Synchronization after Reset	4-23
40	4.7.7	Link Synchronization.....	4-24
41	4.7.7.1	Initial Router Synchronization	4-24

1	4.7.7.2	New Router Neighbor Synchronization	4-25
2	4.7.7.3	Router Synchronization after Reset.....	4-25
3	4.7.7.4	REED and FED Synchronization	4-26
4	4.8	Operational Dataset Announcements	4-26
5	4.8.1	Processing Announcements	4-27
6	4.9	Message Transmission	4-27
7	4.10	Processing of Incoming Messages.....	4-28
8	4.11	Parameters and Constants	4-29
9	4.12	IANA Notes	4-30
10	4.12.1	Security Suites	4-30
11	4.12.2	Command Types	4-31
12	4.12.3	TLV Types	4-32
13	4.12.4	Security Considerations.....	4-36

14 **Chapter 5 Network Layer..... 5-1**

15	5.1	IPv6 Addressing	5-5
16	5.2	IPv6 Addressing Architecture	5-8
17	5.2.1	Address Scopes.....	5-8
18	5.2.1.1	Link-Local Scope	5-8
19	5.2.1.2	Realm-Local Scope	5-8
20	5.2.2	Unicast Addressing	5-8
21	5.2.2.1	Routing Locator (RLOC)	5-9
22	5.2.2.2	Anycast Locator (ALOC)	5-9
23	5.2.2.2.1	Leader ALOC.....	5-10
24	5.2.2.2.2	DHCPv6 Agent ALOC.....	5-10
25	5.2.2.2.3	Service ALOC.....	5-11
26	5.2.2.2.4	Commissioner ALOC	5-11
27	5.2.2.2.5	Neighbor Discovery ALOC.....	5-11
28	5.2.2.3	Endpoint Identifier (EID)	5-11
29	5.2.2.4	Link-Local Addresses	5-11
30	5.2.2.5	Mesh-Local Addresses.....	5-11
31	5.2.2.6	Global Addresses.....	5-12
32	5.2.3	Multicast Addressing	5-12
33	5.2.3.1	Link-Local Scope	5-12
34	5.2.3.2	Realm-Local Scope	5-13
35	5.2.3.3	Other Multicast Scopes	5-13
36	5.3	IPv6 Address Configuration	5-13
37	5.3.1	Link-Local Addresses	5-13
38	5.3.2	Mesh-Local Addresses.....	5-14
39	5.3.3	Global Addresses.....	5-14
40	5.4	EID-to-RLOC Mapping.....	5-14

1	5.4.1	Information Base	5-15
2	5.4.1.1	Local Address Set.....	5-15
3	5.4.1.2	MTD Child Address Set.....	5-15
4	5.4.1.3	Address Query Set	5-15
5	5.4.2	Address Query	5-15
6	5.4.2.1	Transmission of Address Query Messages.....	5-15
7	5.4.2.2	ADDR_QRY.qry - Address Query	5-16
8	5.4.2.3	Receipt of Address Query Messages.....	5-16
9	5.4.2.4	ADDR_NTF.ans - Address Notification	5-17
10	5.4.2.5	Receipt of Address Notification Messages	5-17
11	5.4.3	Proactive Address Notifications	5-17
12	5.4.3.1	ADDR_NTF.ntf - Proactive Address Notification.....	5-18
13	5.4.3.2	Receipt of Proactive Address Notifications.....	5-18
14	5.5	EID-to-RLOC Map Cache	5-18
15	5.5.1	Information Base	5-19
16	5.5.1.1	EID-to-RLOC Set	5-19
17	5.5.2	Managing EID-to-RLOC Map Cache Entries	5-19
18	5.5.2.1	Cache Errors.....	5-19
19	5.5.2.2	Optimizations	5-20
20	5.6	Duplicate IPv6 Address Detection.....	5-20
21	5.6.1	Address Queries.....	5-20
22	5.6.2	Proactive Address Notifications	5-20
23	5.6.3	Creation of Address Error Notifications.....	5-21
24	5.6.3.1	ADDR_ERR.ntf - Address Error Notification	5-21
25	5.6.4	Receipt of Address Error Notifications	5-21
26	5.7	DHCPv6 Services	5-22
27	5.8	ICMPv6	5-22
28	5.9	Routing Protocol	5-23
29	5.9.1	Routing Database.....	5-24
30	5.9.1.1	Router ID Set	5-24
31	5.9.1.2	Link Set	5-24
32	5.9.2	Route Set	5-24
33	5.9.3	Leader Database	5-24
34	5.9.3.1	ID Assignment Set	5-25
35	5.9.4	Link Margins and Link Metrics	5-25
36	5.9.5	Routing Cost and Next Hop.....	5-26
37	5.9.6	Loops and Loop Detection	5-26
38	5.9.7	Sending Advertisements.....	5-26
39	5.9.8	Processing Route64 TLVs.....	5-27
40	5.9.9	Router ID Management.....	5-27
41	5.9.10	Router ID Assignment.....	5-29
42	5.9.10.1	ADDR_SOL.req – Address Solicit Request	5-29

1	5.9.10.2	ADDR_SOL.rsp – Address Solicit Response	5-30
2	5.9.10.3	ADDR_REL.ntf – Address Release Notification	5-31
3	5.10	Unicast Packet Forwarding	5-31
4	5.10.1	Unicast Packet Forwarding inside the Thread Network.....	5-31
5	5.10.1.1	Full Thread Device Forwarding	5-31
6	5.10.1.2	Minimal Thread Device Forwarding	5-32
7	5.10.1.3	Forwarding of Packets with Mesh Address Headers	5-32
8	5.10.2	Unicast Packet Forwarding outside the Thread Network.....	5-33
9	5.11	Multicast Packets Forwarding.....	5-33
10	5.11.1	Link-Local Scope	5-33
11	5.11.2	Realm-Local and Larger Scopes.....	5-34
12	5.12	Selection of Link-Layer Destination Addresses	5-38
13	5.13	Thread Network Data	5-38
14	5.13.1	Version Number Set	5-39
15	5.13.2	Valid Prefix Set	5-39
16	5.13.3	External Route Set	5-40
17	5.13.4	6LoWPAN Context ID Set	5-40
18	5.13.5	Server Set	5-41
19	5.13.6	Commissioning Data.....	5-41
20	5.13.7	Provisioning Domain	5-41
21	5.14	Stable Thread Network Data.....	5-42
22	5.15	Network Data and Propagation	5-43
23	5.15.1	Modifying Network Data	5-43
24	5.15.2	Propagation to rx-on-when-idle Devices	5-43
25	5.15.3	Propagation of Operational Datasets	5-44
26	5.15.4	Propagation to rx-off-when-idle Devices	5-45
27	5.15.5	Leader and Thread Network Data	5-45
28	5.15.6	Server Behavior	5-46
29	5.15.6.1	SVR_DATA.ntf – Server Data Notification	5-46
30	5.15.7	Router Behavior	5-47
31	5.15.8	Host Behavior	5-48
32	5.15.8.1	ND_DATA.req (/nd) – Neighbor Discovery Data Request	5-48
33	5.15.8.2	ND_DATA.rsp (/nd) – Neighbor Discovery Data Response	5-49
34		
35		
36	5.15.9	6LoWPAN Contexts	5-49
37	5.16	Thread Network Partitions	5-49
38	5.16.1	Losing Connectivity	5-50
39	5.16.2	Starting a New Thread Network Partition.....	5-51
40	5.16.3	Merging Thread Network Partitions	5-51

1	5.16.4 Resetting Thread Network Partition Data	5-52
2	5.16.5 Loss of Leader	5-52
3	5.16.6 Children	5-53
4	5.17 Protocol Parameters and Constants	5-53
5	5.18 Network Data Encoding.....	5-57
6	5.18.1 Has Route TLV	5-58
7	5.18.2 Prefix TLV	5-59
8	5.18.3 Border Router TLV.....	5-60
9	5.18.4 6LoWPAN ID TLV.....	5-61
10	5.18.5 Commissioning Data TLV.....	5-61
11	5.18.6 Service TLV	5-61
12	5.18.7 Server TLV	5-62
13	5.19 Network Layer TLVs	5-63
14	5.19.1 Target EID TLV	5-64
15	5.19.2 MAC Extended Address TLV	5-64
16	5.19.3 RLOC16 TLV	5-64
17	5.19.4 ML-EID TLV	5-65
18	5.19.5 Status TLV	5-65
19	5.19.6 Time Since Last Transaction TLV	5-65
20	5.19.7 Router Mask TLV	5-66
21	5.19.8 ND Option TLV	5-66
22	5.19.9 ND Data TLV	5-66
23	5.19.10 Thread Network Data TLV	5-67
24	5.20 MLE Routing TLV.....	5-67
25	5.20.1 MLE Route64 TLV Format	5-67
26	Chapter 6 Transport Layer	6-1
27	6.1 UDP	6-2
28	6.2 TCP	6-3
29	Chapter 7 Security	7-1
30	7.1 Security Material Generation	7-3
31	7.1.1 Security Constants	7-3
32	7.1.2 Security MIB 7-3	
33	7.1.3 Sequence Counter Maintenance	7-4
34	7.1.4 Key Generation	7-4
35	7.1.4.1 Test Vector 1	7-5
36	7.1.4.1.1 Inputs 7-5	
37	7.1.4.1.2 Outputs	7-5
38	7.1.4.2 Test Vector 2	7-5

1	7.1.4.2.1	Inputs	7-5
2	7.1.4.2.2	Outputs	7-5
3	7.1.4.3	Test Vector 3.....	7-5
4	7.1.4.3.1	Inputs 7-5	
5	7.1.4.3.2	Outputs	7-6
6	7.1.4.4	Example Test Program.....	7-6
7	7.1.5	Key Index	7-6
8	7.1.6	Key Rotation	7-6
9	7.1.7	Key Switching.....	7-7
10	7.2	MAC Security.....	7-8
11	7.2.1	MAC Frame Security Processing	7-8
12	7.2.1.1	Key ID Mode 0	7-8
13	7.2.1.1.1	JOIN_ENT.req Processing	7-8
14	7.2.1.1.2	JOIN_ENT.rsp Processing	7-8
15	7.2.1.2	Key ID Mode 1	7-8
16	7.2.1.2.1	Key Index Match	7-9
17	7.2.1.2.2	Key Index Mismatch	7-9
18	7.2.1.3	Key ID Mode 2	7-9
19	7.2.2.2	MAC Security PIB Settings.....	7-9
20	7.2.2.1	MAC Default Key Source	7-10
21	7.2.2.2	MAC Key Table.....	7-10
22	7.2.2.2.1	Key Descriptor	7-10
23	7.2.2.2.2	Key ID Lookup for Key ID Mode 0.....	7-11
24	7.2.2.2.3	Key ID Lookup for Key ID Mode 1.....	7-11
25	7.2.2.2.4	Key ID Lookup for Key ID Mode 2.....	7-12
26	7.2.2.2.5	Key Device List	7-12
27	7.2.2.2.6	Key Usage List	7-13
28	7.2.2.3	MAC Device Table.....	7-13
29	7.2.2.3.1	Device Descriptor.....	7-13
30	7.2.2.4	MAC Security Level Table	7-14
31	7.2.3.3	MAC Auxiliary Security Header Format	7-15
32	7.2.3.1	Security Control Field	7-15
33	7.2.3.2	Frame Counter Field	7-15
34	7.2.3.3	Key Identifier Field	7-15
35	7.2.3.3.1	Key Index Subfield	7-15
36	7.2.3.3.2	Key Source Subfield	7-16
37	7.3	MLE Security	7-16
38	7.3.1	MLE Message Security Processing	7-16
39	7.3.1.1	Key ID Mode 1	7-16
40	7.3.1.1.1	Key Index Match	7-16
41	7.3.1.1.2	Key Index Mismatch	7-16
42	7.3.1.1.2.1	Key Index within 128.....	7-16
43	7.3.1.1.2.2	Key Index not within 128	7-17
44	7.3.1.2	Key ID Mode 2	7-18
45	7.3.1.2.1	Key Index Mismatch	7-18
46	7.3.1.2.1.1	Tentative MLE Messages	7-18
47	7.3.1.2.1.2	Authoritative MLE Messages	7-19
48	7.3.1.2.1.3	Peer MLE Messages	7-19

1	7.3.2 MLE Security MIB Settings.....	7-19
2	7.3.2.1 MLE Shared Frame Counter	7-20
3	7.3.2.2 MLE Frame Counter	7-20
4	7.3.2.3 MLE Key Table	7-20
5	7.3.2.3.1 MLE Key Descriptor	7-20
6	7.3.2.3.2 Key ID Lookup for Key ID Mode 1.....	7-21
7	7.3.2.3.3 Key ID Lookup for Key ID Mode 2.....	7-21
8	7.3.2.3.4 MLE Key Device List.....	7-22
9	7.3.2.4 MLE Device Table	7-22
10	7.3.2.4.1 MLE Device Descriptor	7-22
11	7.3.3 MLE Auxiliary Security Header Format.....	7-23
12	7.3.3.1 Security Control Field	7-23
13	7.3.3.2 Frame Counter Field	7-23
14	7.3.3.3 Key Identifier Field	7-24
15	7.3.3.3.1 Key Index Subfield	7-24
16	7.3.3.3.2 Key Source Subfield	7-24
17	7.3.4 MLE Frame Counter Processing	7-24
18	7.3.4.1 Frame Counter Values	7-24
19	7.3.4.1.1 auxFrameCounter.....	7-24
20	7.3.4.1.2 storedFrameCounter.....	7-24
21	7.3.4.2 MLE Message Processing	7-24
22	7.3.4.2.1 MLE Device Descriptor Present.....	7-24
23	7.3.4.2.1.1 auxFrameCounter Is Invalid	7-25
24	7.3.4.2.1.2 auxFrameCounter Is Valid	7-25
25	7.3.4.2.2 MLE Device Descriptor Absent.....	7-25
26	7.3.4.3 MLE Message Processing Example	7-25
27	7.4 DTLS	7-26
28	7.4.1 Conventions	7-26
29	7.4.1.1 7.4.1.1 Elliptic Curve Points.....	7-26
30	7.4.1.2 Integers.....	7-27
31	7.4.1.3 Octet Strings	7-27
32	7.4.2 Integer and Octet String Conversions	7-27
33	7.4.2.1 Integer to Octet String Conversion	7-27
34	7.4.2.2 Octet String to Integer Conversion	7-27
35	7.4.3 Handshake	7-28
36	7.4.4 Failure Processing	7-28
37	7.4.5 Retransmission Processing	7-28
38	7.4.5.1 Retransmission example	7-29
39	7.4.6 Random Number Generation.....	7-29
40	7.4.7 Elliptic Curve J-PAKE Extensions.....	7-29
41	7.4.7.1 Existing Definitions.....	7-29
42	7.4.7.2 Additional Definitions	7-30
43	7.4.7.2.1 Public Key and ZKP Pair	7-30
44	7.4.7.2.2 Schnorr Zero Knowledge Proof.....	7-30
45	7.4.7.2.2.1 Schnorr ZKP Hash Calculation.....	7-30
46	7.4.7.3 ClientHello and ServerHello Extensions	7-31
47	7.4.7.3.1 Public Key Generation.....	7-31
48	7.4.7.3.2 Schnorr ZKP Generation.....	7-32

1	7.4.7.3.3	Schnorr ZKP Verification	7-32
2	7.4.7.4	Shared Secret Conversion	7-32
3	7.4.7.4.1	Example.....	7-33
4	7.4.7.5	ServerKeyExchange and ClientKeyExchange Enumeration.....	
5		7-33
6	7.4.7.6	ServerKeyExchange.....	7-33
7	7.4.7.6.1	Public Key Generation.....	7-33
8	7.4.7.6.2	Schnorr ZKP Generation.....	7-34
9	7.4.7.6.3	Schnorr ZKP Verification	7-34
10	7.4.7.7	ClientKeyExchange	7-35
11	7.4.7.7.1	Public Key Generation.....	7-35
12	7.4.7.7.2	Schnorr ZKP Generation.....	7-35
13	7.4.7.7.3	Schnorr ZKP Verification	7-36
14	7.4.7.8	Server Secret Key Generation.....	7-36
15	7.4.7.9	Client Secret Key Generation	7-37
16	7.4.7.10	AEAD Record Protection	7-37
17	7.4.7.10.1	Nonce Formation	7-37
18	7.4.8	Test Program for Hash Generation.....	7-38
19			

20 **Chapter 8 Mesh Commissioning Protocol 8-1**

21	8.1	Introduction	8-5
22	8.2	Terminology	8-5
23	8.3	Overview	8-7
24	8.4	Functional Specification	8-8
25	8.4.1	Commissioner Protocol.....	8-8
26	8.4.1.1	Commissioner Discovery of Thread Network	8-8
27	8.4.1.1.1	Commissioner Discovery - Native 802.15.4	8-8
28	8.4.1.1.2	Commissioner Discovery - Ethernet / Wi-Fi.....	8-8
29	8.4.1.1.3	Commissioner Discovery - BLE / other	8-13
30	8.4.1.2	Commissioner Authentication.....	8-14
31	8.4.1.2.1	Derivation of PSKc.....	8-14
32	8.4.1.2.2	Test Vector for Derivation of PSKc	8-14
33	8.4.1.3	Commissioner Registration	8-15
34	8.4.2	Thread Management Protocol.....	8-15
35	8.4.2.1	Commissioner Petitioning	8-15
36	8.4.2.2	Commissioner Management.....	8-16
37	8.4.2.2.1	Commissioner Dataset	8-17
38	8.4.2.2.2	Active Operational Dataset	8-17
39	8.4.2.2.3	Pending Operational Dataset.....	8-18
40	8.4.3	Dissemination of Datasets	8-19
41	8.4.3.1	Management of Thread Network Data Versions	8-19
42	8.4.3.2	Transmitting Thread Network Data	8-19
43	8.4.3.3	Receiving New Datasets	8-19
44	8.4.3.4	Delay Timer Management.....	8-20
45	8.4.3.5	Migrating Thread Network Partitions	8-20
46	8.4.3.5.1	Synchronizing Active Operational Datasets	8-20
47	8.4.3.5.2	Synchronizing Pending Operational Datasets.....	8-21

1	8.4.4	Joiner Protocol	8-21
2	8.4.4.1	Discovery of Thread Network	8-21
3	8.4.4.1.1	Native Discovery Messages.....	8-21
4	8.4.4.1.1.1	Discovery Request.....	8-21
5	8.4.4.1.1.2	Discovery Response.....	8-22
6	8.4.4.1.1.3	Discovery Messages Security Considerations.....	8-23
7	8.4.4.1.2	Joiner Discovery.....	8-23
8	8.4.4.2	IEEE 802.15.4 Beacon Frame Format.....	8-24
9	8.4.4.3	Bloom Filters	8-26
10	8.4.4.4	Joiner Provisional Join.....	8-27
11	8.4.4.5	Joiner Authentication	8-27
12	8.4.5	Joiner Finalization	8-29
13	8.4.5.1	Joiner Entrust	8-30
14	8.4.5.2	Joiner Provisioning	8-31
15	8.4.5.3	Joiner Session Close	8-31
16	8.4.6	Out-of-band Commissioning	8-31
17	8.4.6.1	First Border Agent Commissioning	8-31
18	8.4.6.2	Secure Out-of-band Commissioning.....	8-32
19	8.4.6.3	Out-of-band Network Commands	8-32
20	8.4.6.3.1	Network Form Request.....	8-32
21	8.4.6.3.2	Network Scan Request	8-33
22	8.4.6.3.3	Network Join Request	8-33
23	8.4.6.3.4	Network Leave Request.....	8-33
24	8.4.7	Diagrams	8-34
25	8.4.7.1	Topology	8-34
26	8.4.7.1.1	Expanded Example	8-34
27	8.4.7.1.2	Collapsed Case: Border Agent is Joiner Router	8-35
28	8.4.7.1.3	Collapsed Case: Commissioner is Border Agent	8-36
29	8.4.7.1.4	Collapsed Case: Commissioner is Joiner Router.....	8-37
30	8.4.7.2	Flow Charts	8-38
31	8.4.7.2.1	Joiner State Machine Flow Chart.....	8-38
32	8.4.7.2.2	Commissioner State Machine Flow Chart	8-39
33	8.4.7.2.3	Joiner Router State Machine Flow Chart	8-40
34	8.4.7.3	Data Flows	8-41
35	8.4.7.3.1	Collapsed Thread-only Data Flow	8-41
36	8.4.7.4	Port Usage Diagrams	8-41
37	8.4.7.4.1	External Commissioner Port Usage	8-42
38	8.4.7.4.2	On-Mesh Commissioner Port Usage	8-43
39	8.4.7.4.3	Commissioner Management Port Usage.....	8-43
40	8.5	Message Format	8-44
41	8.6	Commissioner Scope TMF Messages	8-44
42	8.6.1	Petitioning Commands	8-44
43	8.6.1.1	COMM_PET.req – Commissioner Petition Request	8-44
44	8.6.1.2	COMM_PET.rsp – Commissioner Petition Response	8-45
45	8.6.1.3	COMM_KA.req – Commissioner Keep Alive Request	8-45
46	8.6.1.4	COMM_KA.rsp – Commissioner Keep Alive Response	8-46
47	8.6.2	Proxy Commands	8-46
48	8.6.2.1	UDP_RX.ntf – Proxy Receive UDP Notification	8-46

1	8.6.2.2	UDP_TX.ntf – Proxy Transmit UDP Notification	8-47
2	8.7	Commissioner and Thread Network Scope Commands	8-47
3	8.7.1	Relay Commands	8-48
4	8.7.1.1	RLY_RX.ntf – DTLS Relay Receive Notification.....	8-48
5	8.7.1.2	RLY_TX.ntf – DTLS Relay Transmit Notification	8-49
6	8.7.2	Network Management Commands	8-49
7	8.7.2.1	MGMT_GET.req – Get Management Data Request.....	8-50
8	8.7.2.2	MGMT_GET.rsp – Get Management Data Response.....	8-50
9	8.7.2.3	MGMT_SET.req – Set Management Data Request	8-50
10	8.7.2.4	MGMT_SET.rsp – Set Management Data Response	8-51
11	8.7.3	Updating the Commissioner Dataset	8-51
12	8.7.3.1	Updates from a Commissioner	8-51
13	8.7.3.2	MGMT_COMMISsIONER_GET.req – Get Commissioner	
14		Dataset Request	8-51
15	8.7.3.3	MGMT_COMMISsIONER_GET.rsp – Get Commissioner	
16		Dataset Response	8-52
17	8.7.3.4	MGMT_COMMISsIONER_SET.req – Set Commissioner	
18		Dataset Request	8-52
19	8.7.3.5	MGMT_COMMISsIONER_SET.rsp – Set Commissioner	
20		Dataset Response	8-53
21	8.7.3.6	TLV Encoding.....	8-53
22	8.7.3.7	Leader Behavior.....	8-54
23	8.7.4	Updating the Active Operational Dataset	8-54
24	8.7.4.1	Updates from a Commissioner	8-55
25	8.7.4.2	Updates from a Thread Device	8-55
26	8.7.4.3	MGMT_ACTIVE_GET.req – Get Active Operational Dataset	
27		Request	8-56
28	8.7.4.4	MGMT_ACTIVE_GET.rsp – Get Active Operational Dataset	
29		Response	8-57
30	8.7.4.5	MGMT_ACTIVE_SET.req – Set Active Operational Dataset	
31		Request	8-57
32	8.7.4.6	MGMT_ACTIVE_SET.rsp – Set Active Operational Dataset	
33		Response	8-58
34	8.7.4.7	TLV Encoding.....	8-58
35	8.7.4.8	Leader Behavior.....	8-59
36	8.7.5	Updating the Pending Operational Dataset.....	8-60
37	8.7.5.1	Updates from a Commissioner	8-60
38	8.7.5.2	Updates from a Thread Device	8-61
39	8.7.5.3	MGMT_PENDING_GET.req – Get Pending Operational	
40		Dataset Request	8-61
41	8.7.5.4	MGMT_PENDING_GET.rsp – Get Pending Operational	
42		Dataset Response	8-62
43	8.7.5.5	MGMT_PENDING_SET.req – Set Pending Operational	
44		Dataset Request	8-62
45	8.7.5.6	MGMT_PENDING_SET.rsp – Set Pending Operational	
46		Dataset Response	8-63
47	8.7.5.7	TLV Encoding.....	8-63
48	8.7.5.8	Leader Behavior.....	8-64
49	8.7.6	Concurrent Updates.....	8-65

1	8.7.6.1	MGMT_DATASET_CHANGED.ntf –Dataset Changed Notification.....	8-65
2	8.7.7	Orphaned End Devices	8-65
3	8.7.8	Merging Channel and PAN ID Partitions	8-66
4	8.7.8.1	MGMT_ANNOUNCE_BEGIN.ntf –Announce Begin Notification.....	8-66
5	8.7.9	Avoiding PAN ID Conflicts.....	8-67
6	8.7.9.1	MGMT_PANID_QUERY.qry –PAN ID Query Notification ..	8-67
7	8.7.9.2	MGMT_PANID_CONFLICT.ans – PAN ID Conflict Notification.....	8-67
8	8.7.10	Collecting Energy Scan Information	8-69
9	8.7.10.1	MGMT_ED_SCAN.qry – Energy Detect Scan Notification	8-69
10	8.7.10.2	MGMT_ED_REPORT.ans –Energy Detect Report Notification.....	8-70
11	8.8	Thread Network Scope Commands	8-71
12	8.8.1	Leader Commands	8-71
13	8.8.1.1	LEAD_PET.req – Leader Petition Request.....	8-71
14	8.8.1.2	LEAD_PET.rsp – Leader Petition Response	8-71
15	8.8.1.3	LEAD_KA.req – Leader Petition Keep Alive Request	8-72
16	8.8.1.4	LEAD_KA.rsp – Leader Petition Keep Alive Response.....	8-72
17	8.8.2	Thread Network Data Update.....	8-73
18	8.8.2.1	Commissioning Data TLV.....	8-73
19	8.8.2.2	Thread Network Data Change Callback.....	8-74
20	8.8.2.3	Thread Network Data Change Command	8-74
21	8.8.2.4	Thread Network Data Update after Attach	8-75
22	8.9	Joiner Scope Commands	8-75
23	8.9.1	Joiner Entrust Commands.....	8-75
24	8.9.1.1	JOIN_ENT.ntf – Joiner Entrust Notification	8-75
25	8.9.2	Joiner Session Commands	8-76
26	8.9.2.1	JOIN_FIN.req – Joiner Finalization Request.....	8-76
27	8.9.2.2	JOIN_FIN.rsp – Joiner Finalization Response	8-77
28	8.9.3	Joiner Provisioning Commands	8-77
29	8.9.3.1	JOIN_APP.req – Joiner Application Request.....	8-77
30	8.9.3.2	JOIN_APP.rsp – Joiner Application Response	8-78
31	8.10	MeshCoP TLV Formats	8-79
32	8.10.1	Network Management TLVs	8-79
33	8.10.1.1	Channel TLV (0)	8-80
34	8.10.1.1.1	Supported Channel Pages.....	8-80
35	8.10.1.1.2	Supported Channels	8-80
36	8.10.1.2	PAN ID TLV (1)	8-80
37	8.10.1.3	Extended PAN ID TLV (2)	8-81
38	8.10.1.4	Network Name TLV (3)	8-81
39	8.10.1.5	PSKc TLV (4)	8-81
40	8.10.1.6	Network Master Key TLV (5)	8-83
41	8.10.1.7	Network Key Sequence Counter TLV (6).....	8-83

1	8.10.1.8	Network Mesh-Local Prefix TLV (7)	8-83
2	8.10.1.9	Steering Data TLV (8).....	8-83
3	8.10.1.10	Border Agent Locator TLV (9)	8-84
4	8.10.1.11	Commissioner ID TLV (10)	8-85
5	8.10.1.12	Commissioner Session ID TLV (11).....	8-85
6	8.10.1.13	Active Timestamp TLV (14)	8-85
7	8.10.1.14	Commissioner UDP Port TLV (15)	8-86
8	8.10.1.15	Security Policy TLV (12)	8-86
9	8.10.1.16	Pending Timestamp TLV (51).....	8-87
10	8.10.1.17	Delay Timer TLV (52)	8-88
11	8.10.1.18	Channel Mask TLV (53)	8-88
12	8.10.1.18.1	Channel Mask Entry.....	8-88
13	8.10.2	MeshCoP Protocol Command TLVs	8-89
14	8.10.2.1	Get TLV (13)	8-89
15	8.10.2.2	State TLV (16).....	8-89
16	8.10.2.3	Joiner DTLS Encapsulation TLV (17).....	8-89
17	8.10.2.4	Joiner UDP Port TLV (18).....	8-90
18	8.10.2.5	Joiner IID TLV (19).....	8-90
19	8.10.2.6	Joiner Router Locator TLV (20)	8-90
20	8.10.2.7	Joiner Router KEK TLV (21)	8-91
21	8.10.2.8	Count TLV (54)	8-91
22	8.10.2.9	Period TLV (55).....	8-92
23	8.10.2.10	Scan Duration TLV (56).....	8-92
24	8.10.2.11	Energy List TLV (57).....	8-92
25	8.10.3	TMF Provisioning and Discovery TLVs.....	8-93
26	8.10.3.1	Provisioning URL TLV (32)	8-93
27	8.10.3.2	Vendor Name TLV (33)	8-93
28	8.10.3.3	Vendor Model TLV (34)	8-94
29	8.10.3.4	Vendor SW Version TLV (35)	8-94
30	8.10.3.5	Vendor Data TLV (36)	8-94
31	8.10.3.6	Vendor Stack Version TLV (37)	8-95
32	8.10.3.7	UDP Encapsulation TLV (48)	8-95
33	8.10.3.8	IPv6 Address TLV (49)	8-96
34	8.10.3.9	Discovery Request TLV (128).....	8-96
35	8.10.3.10	Discovery Response TLV (129).....	8-97
36	8.11	Parameters and Constants	8-97
37	8.12	IANA Considerations.....	8-99

38	Chapter 9	Border Router.....	9-1
39	9.1	Introduction	9-2
40	9.2	Overview	9-2
41	9.2.1	Common Characteristics.....	9-2
42	9.2.2	Border Router Availability.....	9-3
43	9.2.3	Global Addresses.....	9-3
44	9.2.4	Supplemental Unique Local Addresses	9-3
45	9.2.5	Network Data	9-4
46	9.2.6	Coping with IPv4 Infrastructure	9-4

1	9.2.7	Packet Filtering	9-5
2	9.2.8	Role in Commissioning	9-5
3	9.3	Functional Specification	9-5
4	9.3.1	Participation in the Interior Network	9-5
5	9.3.1.1	Configuration of Network Data	9-6
6	9.3.1.2	Propagation of Network Data	9-7
7	9.3.1.3	DHCPv6 Server for Global EID Assignment	9-7
8	9.3.2	Packet Forwarding Between Interior and Exterior Interfaces	9-7
9	9.3.3	Participation in Exterior Networks	9-8
10	9.3.3.1	General Functionality	9-8
11	9.3.3.2	Border Routers as Customer Edge Devices	9-9
12	9.3.4	Participation in Commissioning	9-9
13	9.3.5	Participation in Port Control	9-9
14	9.3.6	Security Considerations	9-10
15	9.4	Example of Operation and Role Fulfillment	9-10
16	9.4.1	Functional Roles of a Border Router	9-10
17	9.4.2	Border Router Role Transitions	9-11
18	9.4.2.1	Network Formation by the Border Router as Leader	9-12
19	9.4.2.2	Joining with Co-located Commissioner	9-13
20	9.4.2.3	Address Assignment, External Routing	9-13
21	9.4.2.4	Petitioning by External Commissioner	9-14
22	9.4.2.5	Joining with External Commissioner	9-15
23	9.4.2.6	Transitioning from the Leader Role	9-16
24	9.4.2.7	Multiple Border Routers	9-18

25	Chapter 10	Management.....	10-1
26	10.1	Introduction	10-3
27	10.2	Framework Overview	10-3
28	10.3	URIs and Notation	10-3
29	10.3.1	URIs	10-3
30	10.3.2	CoAP Request Notation	10-4
31	10.3.3	CoAP Response Notation	10-5
32	10.4	CoAP Message Format	10-5
33	10.4.1	Header Format	10-5
34	10.4.2	CoAP Options	10-5
35	10.4.2.1	Mandatory Options	10-5
36	10.4.2.2	Options Not Recommended	10-5
37	10.4.3	Payload Format	10-6
38	10.4.4	CoAP Response Codes	10-7
39	10.5	CoAP Parameters	10-7
40	10.6	CoAP Message Delivery	10-7

1	10.6.1 Reliability	10-7
2	10.6.2 TMF IPv6/UDP Addressing Rules.....	10-8
3	10.6.3 TMF Received Message Processing.....	10-8
4	10.7 TMF Message Types.....	10-8
5	10.7.1 TMF Request	10-9
6	10.7.2 TMF Response.....	10-9
7	10.7.3 TMF Query	10-9
8	10.7.4 TMF Answer	10-9
9	10.7.5 TMF Notification	10-9
10	10.7.6 CoAP Mapping.....	10-9
11	10.8 TMF Message Transaction Patterns	10-10
12	10.8.1 Unicast TMF Messages	10-10
13	10.8.2 Multicast TMF Messages	10-10
14	10.8.3 Token Usage	10-11
15	10.8.4 CoAP Request/Response Pairing	10-11
16	10.8.5 TMF Request, Piggybacked TMF Response	10-11
17	10.8.6 TMF Request, Separate TMF Response	10-12
18	10.8.6.1 Example Scenario.....	10-12
19	10.8.7 Confirmable TMF Notification, Query or Answer, Piggybacked Dummy	
20	Response	10-13
21	10.8.8 Non-confirmable TMF Notification, Query or Answer.....	10-13
22	10.9 Commissioner Transactions	10-14
23	10.9.1 Native or External Commissioner with Thread Device.....	10-14
24	10.9.1.1 Destined for Thread Device	10-15
25	10.9.1.1.1 Commissioner Behavior.....	10-16
26	10.9.1.1.2 Border Agent Behavior.....	10-16
27	10.9.1.2 Originating from Thread Device.....	10-16
28	10.9.1.2.1 Thread Device Behavior	10-16
29	10.9.1.2.2 Border Agent Behavior.....	10-16
30	10.9.2 Native or External Commissioner with Border Agent	10-17
31	10.9.3 On-mesh Commissioner	10-17
32	10.10 Message Mapping	10-18
33	10.10.1 Address Resolution API (/a/....)	10-18
34	10.10.2 MeshCop API (/c/....).....	10-19
35	10.10.3 Diagnostic API (/d/....).....	10-21
36	10.11 Network Diagnostics	10-21
37	10.11.1 Overview.....	10-21
38	10.11.1.1 Device Diagnostics	10-21
39	10.11.1.2 Network Diagnostics	10-21
40	10.11.1.3 Link Diagnostics	10-22
41	10.11.2 Diagnostic Commands	10-22

1	10.11.2.1	DIAG_GET.req – Get Diagnostic Request	10-22
2	10.11.2.2	DIAG_GET.rsp – Get Diagnostic Response	10-22
3	10.11.2.3	DIAG_GET.qry – Get Diagnostic Query	10-23
4	10.11.2.4	DIAG_GET.ans – Get Diagnostic Answer	10-23
5	10.11.2.5	DIAG_RST.ntf – Reset Diagnostic Notification	10-23
6	10.11.3	Diagnostic TLVs Overview	10-24
7	10.11.4	Diagnostic TLVs	10-25
8	10.11.4.1	MAC Counters TLV (9)	10-25
9	10.11.4.2	Battery Level TLV (14)	10-26
10	10.11.4.3	Supply Voltage TLV (15)	10-26
11	10.11.4.4	Child Table TLV (16)	10-26
12	10.11.4.5	Channel Pages TLV (17)	10-27
13	10.11.4.5.1	Supported Channel Pages	10-27
14	10.11.4.6	Type List TLV (18)	10-27
15	10.11.4.7	Max Child Timeout TLV (19)	10-28
16	10.12	Persistent Data	10-28
17	10.13	IANA Considerations	10-29

18	Chapter 11 Functional Description.....	11-1
19	11.1 Forming a Network	11-2
20	11.1.1 Types of Devices and Network Structure	11-2
21	11.1.2 Thread Network Data	11-3
22	11.2 Joining a Thread Network	11-3
23	11.3 Links and Link Establishment	11-4
24	11.4 Message Forwarding and Routing	11-5
25	11.5 Router Selection	11-5
26	11.6 FTD Parents and MTD Children	11-6
27	11.7 Partitioning	11-6
28	11.8 Commissioning	11-7
29		

1 List of Figures

2	Figure 4-1. MLE Message Format.....	4-4
3	Figure 4-2. MLE Message Command Format	4-5
4	Figure 4-3. Base TLV Format	4-6
5	Figure 4-4. Extended TLV Format	4-7
6	Figure 4-5. Mode TLV Format.....	4-7
7	Figure 4-6. Leader Data TLV Format	4-10
8	Figure 4-7. TLV Request TLV Format.....	4-11
9	Figure 4-8. Scan Mask TLV Format	4-11
10	Figure 4-9. Connectivity TLV Format.....	4-12
11	Figure 4-10. Link Margin TLV Format	4-13
12	Figure 4-11. Status TLV Format	4-13
13	Figure 4-12. Version TLV Format	4-13
14	Figure 4-13. Address Registration TLV Formats	4-14
15	Figure 4-14. Channel TLV Format	4-15
16	Figure 4-15. PAN ID TLV Format	4-15
17	Figure 4-16. Active Timestamp TLV Format	4-16
18	Figure 5-1. Unicast Routing Locator Bits.....	5-9
19	Figure 5-2. Thread Network Data Message Format	5-57
20	Figure 5-3. Has Route TLV Format.....	5-59
21	Figure 5-4. Prefix TLV Format	5-59
22	Figure 5-5. Border Router TLV Format	5-60
23	Figure 5-6. 6LoWPAN ID TLV Format	5-61
24	Figure 5-7. Service TLV Format.....	5-62
25	Figure 5-8. Server TLV Format	5-63
26	Figure 5-9. Base TLV Format	5-63
27	Figure 5-10. Target EID TLV Format	5-64
28	Figure 5-11. RLOC16 TLV Format	5-64
29	Figure 5-12. ML-EID TLV Format	5-65
30	Figure 5-13. Status TLV Format	5-65
31	Figure 5-14. Time Since Last Transaction TLV Format	5-66
32	Figure 5-15. Router Mask TLV Format	5-66
33	Figure 5-16. ND Option TLV Format	5-66
34	Figure 5-17. ND Data TLV Format	5-67
35	Figure 5-18. Thread Network Data TLV Format	5-67
36	Figure 5-19. MLE Route64 TLV Format	5-67
37	Figure 5-20. Link Quality and Route Data Encoding	5-68
38	Figure 7-1. Sequence Counter Maintenance	7-4
39	Figure 7-2. Key Generation.....	7-5
40	Figure 7-3. Key Rotation	7-7
41	Figure 7-4. MLE Message Processing Example.....	7-26
42	Figure 7-5. Basic DTLS Message Exchange	7-28
43	Figure 7-6. Retransmission Example	7-29
44	Figure 8-1. Commissioner Petitioning Data Flow	8-16
45	Figure 8-2. Commissioner Management Flow	8-17
46	Figure 8-3. Joiner Discovery Data Flow	8-24
47	Figure 8-4. Beacon Payload	8-25
48	Figure 8-5. Joiner Authentication Data Flow	8-29
49	Figure 8-6. Joiner Finalization Data Flow	8-30
50	Figure 8-7. Joiner Entrust Data Flow	8-31
51	Figure 8-8. Joiner Session Closure Data Flow	8-31
52	Figure 8-9. Expanded Commissioning Topology.....	8-34

1	Figure 8-10. Collapsed Case Topology–Border Agent is Joiner Router.....	8-35
2	Figure 8-11. Collapsed Case Topology–Commissioner is Border Agent	8-36
3	Figure 8-12. Collapsed Case Topology–Commissioner is Joiner Router	8-37
4	Figure 8-13. Joiner State Machine Flow Chart	8-38
5	Figure 8-14. Commissioner State Machine Flow Chart	8-39
6	Figure 8-15. Joiner Router State Machine Flow Chart	8-40
7	Figure 8-16. Petitioning and Authentication on Thread-only Collapsed Case	8-41
8	Figure 8-17. External Commissioner Port Usage	8-42
9	Figure 8-18. On-Mesh Commissioner Port Usage	8-43
10	Figure 8-19. Commissioner Management Port Usage	8-43
11	Figure 8-20. General Thread Management TLV Format.....	8-79
12	Figure 8-21. Channel TLV (0).....	8-80
13	Figure 8-22. PAN ID TLV (1).....	8-80
14	Figure 8-23. Extended PAN ID TLV (2).....	8-81
15	Figure 8-24. Network Name TLV (3)	8-81
16	Figure 8-25. PSKc TLV (4)	8-82
17	Figure 8-26. Network Master Key TLV (5).....	8-83
18	Figure 8-27. Network Key Sequence Counter TLV (6).....	8-83
19	Figure 8-28. Network Mesh-Local Prefix TLV (7)	8-83
20	Figure 8-29. Steering Data TLV (8).....	8-84
21	Figure 8-30. Border Agent Locator TLV (9)	8-84
22	Figure 8-31. Commissioner ID TLV (10)	8-85
23	Figure 8-32. Commissioner Session ID TLV (11).....	8-85
24	Figure 8-33. Active Timestamp TLV (14)	8-85
25	Figure 8-34. Commissioner UDP Port TLV (15)	8-86
26	Figure 8-35. Security Policy TLV (12).....	8-86
27	Figure 8-36. Pending Timestamp TLV (51).....	8-87
28	Figure 8-37. Delay Timer TLV (52)	8-88
29	Figure 8-38. Channel Mask TLV (53).....	8-88
30	Figure 8-39. Channel Mask Entry.....	8-88
31	Figure 8-40. Get TLV (13)	8-89
32	Figure 8-41. State TLV (16).....	8-89
33	Figure 8-42. Joiner DTLS Encapsulation TLV (17)	8-89
34	Figure 8-43. Joiner UDP Port TLV (18)	8-90
35	Figure 8-44. Joiner IID TLV (19)	8-90
36	Figure 8-45. Joiner Router Locator TLV (20)	8-91
37	Figure 8-46. Joiner Router KEK TLV (21)	8-91
38	Figure 8-47. Count TLV (54)	8-91
39	Figure 8-48. Period TLV (55)	8-92
40	Figure 8-49. Scan Duration TLV (55)	8-92
41	Figure 8-50. Energy List TLV (57).....	8-92
42	Figure 8-51. Provisioning URL TLV (32).....	8-93
43	Figure 8-52. Vendor Name TLV (33)	8-93
44	Figure 8-53. Vendor Model TLV (34)	8-94
45	Figure 8-54. Vendor SW Version TLV (35)	8-94
46	Figure 8-55. Vendor Data TLV (36).....	8-94
47	Figure 8-56. Vendor Stack Version TLV (37)	8-95
48	Figure 8-57. UDP Encapsulation TLV (48)	8-95
49	Figure 8-58. IPv6 Address TLV (49).....	8-96
50	Figure 8-59. Discovery Request TLV (128).....	8-96
51	Figure 8-60. Discovery Response TLV (129)	8-97
52	Figure 9-1. High Level Overview of Border Router Role	9-2
53	Figure 9-2. Propagation and Notification of Network Data	9-4

1	Figure 9-3. BR1 starts a Thread Network as Leader, activates external interface.....	9-12
2	Figure 9-4. BR1 acts as an internal Commissioner for joining R1.....	9-13
3	Figure 9-5. External Commissioner registers with BR1.....	9-14
4	Figure 9-6. BR1 relays commissioning for C to allow R2 to join.....	9-15
5	Figure 9-7. BR1 becomes unavailable. R1 takes over Leader role.....	9-16
6	Figure 9-8. BR1 returns to the network. R1 remains Leader.....	9-17
7	Figure 9-9. BR2 joins the network. Both BR1 and BR2 provide external routing.....	9-18
8	Figure 10-1. Base TLV Format.....	10-6
9	Figure 10-2. Extended TLV Format	10-6
10	Figure 10-3. Req+Rsp_Piggybacked Example Transaction	10-12
11	Figure 10-4. Req+Rsp_Separate Example Transaction	10-12
12	Figure 10-5. Ntf/Qry/Ans_CON Example Transaction	10-13
13	Figure 10-6. Ntf/Qry/Ans_NON Example Transaction	10-14
14	Figure 10-7. Native or External Commissioner Transactions with Thread Device	10-15
15	Figure 10-8. Native or External Commissioner Transactions with Border Agent	10-17
16	Figure 10-9. Thread Device Transactions with Other Thread Device	10-18
17	Figure 10-10. MAC Counters TLV (9)	10-25
18	Figure 10-11. Battery Level TLV (14).....	10-26
19	Figure 10-12. Supply Voltage TLV (15)	10-26
20	Figure 10-13. Child Table TLV (16)	10-26
21	Figure 10-14. Child Entry Data Format.....	10-26
22	Figure 10-15. Channel Pages TLV (17)	10-27
23	Figure 10-16. Type List TLV (18)	10-27
24	Figure 10-17. Max Child Timeout TLV (18)	10-28
25		

1 List of Tables

2	Table 2-1. Terms and Definitions.....	2-5
3	Table 2-2. Nomenclature for Tables Referencing Other Standards.....	2-9
4	Table 2-3. Acronyms and Definitions.....	2-9
5	Table 3-1. Modifications and Statements to Chapter 6 of [IEEE802154]	3-2
6	Table 3-2. Modifications and Statements to Chapter 7 and Annexes of [IEEE802154]	3-3
7	Table 3-3. Modifications and Statements to [RFC 4944]	3-10
8	Table 3-4. Modifications and Statements to [RFC 6282]	3-12
9	Table 4-1. MLE Message Command Types	4-5
10	Table 4-2. MLE Parameters.....	4-29
11	Table 4-3. MLE Security Suites.....	4-31
12	Table 4-4. MLE Command Types	4-31
13	Table 4-5. MLE TLV Types	4-32
14	Table 4-6. MLE Message Command Types and Included TLVs.....	4-33
15	Table 5-1. Modifications and Statements to [RFC 2460]	5-5
16	Table 5-2. Modifications and Statements to [RFC 4291]	5-6
17	Table 5-3. Allocation of the ALOC Space.....	5-10
18	Table 5-5. Modifications and Statements to [RFC 4443]	5-22
19	Table 5-6. Average Link Margin Conversion to One-Way Link Quality	5-25
20	Table 5-7. Modifications and Statements to [RFC 7731]	5-34
21	Table 5-8. Configuration of MPL Parameters	5-37
22	Table 5-9. Protocol Parameters	5-53
23	Table 5-10. TLV Usage.....	5-57
24	Table 6-1. Modifications and Statements to [RFC 7681]	6-2
25	Table 6-2. Modifications and Statements to Chapter 4.1 of [RFC 1122]	6-2
26	Table 7-1. Security Constants	7-3
27	Table 7-2. Security MIB	7-3
28	Table 7-3. MAC Default Key Source	7-10
29	Table 7-4. MAC Key Table	7-10
30	Table 7-5. Key Descriptor.....	7-11
31	Table 7-6. Key ID Lookup for Key ID Mode 0	7-11
32	Table 7-7. Key ID Lookup for Key ID Mode 1	7-12
33	Table 7-8. Key Device List.....	7-12
34	Table 7-9. Key Usage List for MAC Data Frame	7-13
35	Table 7-10. Key Usage List for MAC Data Request Command Frame	7-13
36	Table 7-11. MAC Device Table.....	7-13
37	Table 7-12. Device Descriptor Entry Set.....	7-14
38	Table 7-13. MAC Security Level Table	7-15
39	Table 7-14. Security Control Field	7-15
40	Table 7-15. Example of Key Index Mismatch.....	7-17
41	Table 7-16. MLE Shared Frame Counter	7-20
42	Table 7-17. MLE Frame Counter	7-20
43	Table 7-18. MLE Key Table	7-20
44	Table 7-19. MLE Key Descriptor	7-21
45	Table 7-20. Key ID Lookup for Key ID Mode 1	7-21
46	Table 7-21. Key ID Lookup for Key ID Mode 2	7-22
47	Table 7-22. Key Device List	7-22
48	Table 7-23. MLE Device Table	7-22
49	Table 7-24. MLE Device Descriptor	7-23
50	Table 7-25. Security Control Field	7-23
51	Table 7-26. Schnorr ZKP Hash Calculation	7-30
52	Table 8-1. Participant Roles in the MeshCoP	8-5

1	Table 8-2. Credentials in the MeshCoP Protocol	8-6
2	Table 8-3. Other Useful MeshCoP Terms.....	8-7
3	Table 8-4. DNS-SD TXT Answer Record Key-Value Pair Format	8-10
4	Table 8-5. Border Agent State Bitmap.....	8-13
5	Table 8-6. Active Operational Dataset Parameter TLV Encoding.....	8-58
6	Table 8-7. Pending Operational Dataset Parameter TLV Encoding	8-63
7	Table 8-8. Supported Channel Pages	8-80
8	Table 8-9. Glossary of Constants.....	8-97
9	Table 8-10. MeshCoP Ports	8-100
10	Table 9-1. Border Router Network Data Attributes.....	9-6
11	Table 9-2. Set of External Prefixes Entry	9-6
12	Table 9-3. Border Router Roles and Functions.....	9-10
13	Table 10-1. TMF to CoAP Message Mapping	10-9
14	Table 10-2. TMF Transaction Patterns	10-10
15	Table 10-3. Unicast TMF Message Transaction Patterns	10-10
16	Table 10-4. Multicast TMF Message Transaction Patterns	10-11
17	Table 10-5. Address Resolution API	10-19
18	Table 10-6. MeshCoP API	10-19
19	Table 10-7. Diagnostic API	10-21
20	Table 10-8. Diagnostic TLVs	10-24
21	Table 10-9. Supported Channel Pages.....	10-27
22	Table 10-10. Thread Management Framework Port.....	10-29

1 **CHAPTER 1 OVERVIEW**2 **Contents**

3	1.1 Purpose.....	1-2
4	1.2 Scope and Organization	1-2
5	1.3 Overview.....	1-3
6	1.3.1 No Single Point of Failure	1-3
7	1.3.2 Device Types	1-4
8	1.3.2.1 Border Routers	1-4
9	1.3.2.2 Routers.....	1-4
10	1.3.2.3 Router-Eligible End Devices	1-4
11	1.3.2.4 Sleepy End Devices	1-4
12	1.3.3 Security	1-4
13	1.3.3.1 Authentication and Key Agreement.....	1-4
14	1.3.3.2 Network-wide Key	1-5

1.1 Purpose

2 This chapter provides an overview of the Thread networking stack and the organization of
3 this specification.

4 1.2 Scope and Organization

5 This specification details everything necessary to implement a Thread networking stack. It is
6 intended to be used by implementers as a complete specification but where necessary other
7 references are noted with details on how these references apply to the Thread Network.

8 This specification uses the requirements language from the IETF (Internet Engineering Task
9 Force). See Section 2.2, "Requirements Language" in Chapter 2, Supporting Information,
10 for an explanation of this language.

11 In addition, this specification also refers to IETF standards and other references. For more
12 information, see Section 2.3, "Normative References" and Section 2.4, "Informative
13 References" in Chapter 2, Supporting Information.

14 This specification is organized into the following chapters:

15 **Chapter 1, Overview:** Provides an overview of the Thread networking stack and its goals.

16 **Chapter 2, Supporting Information:** Describes the requirements language, references,
17 key terms and definitions, acronyms used in this specification.

18 **Chapter 3, PHY/MAC/6LoWPAN:** Details the underlying configuration and usage of the
19 IEEE 802.15.4 radio PHY (Physical layer), MAC (Media Access Control sub-layer), and
20 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) usage.

21 **Chapter 4, Mesh Link Establishment:** Provides the full protocol specification for MLE
22 (Mesh Link Establishment) and link layer exchanges required to establish and maintain the
23 Thread Network.

24 **Chapter 5, Network Layer:** Details the network routing and management as well as the
25 network data management.

26 **Chapter 6, Transport Layer:** Provides the references and usage of a transport layer on
27 the Thread Network.

28 **Chapter 7, Security:** Details the security requirements for MLE and the Thread Network.

29 **Chapter 8, Mesh Commissioning Protocol:** Details the protocols and mechanism for a
30 device to be commissioned and attached to a Thread Network. Also includes requirements
31 for the commissioning device.

32 **Chapter 9, Border Router:** Details the requirements for implementing a Border Router, a
33 device that provides connectivity of nodes in the Thread Network to other devices in other
34 IP-based external networks.

35 **Chapter 10, Management:** Describes the management functionality within a Thread
36 Network that is available to the application layer.

37 **Chapter 11, Functional Description:** Provides the functional requirements for devices
38 and the Thread Network during startup and operation.

1.3 Overview

The Thread stack is an open standard for reliable, cost-effective, low power, wireless device-to-device communication. It is designed specifically for Connected Home applications where IP-based networking is desired. A variety of application layers can be used on the stack. The general characteristics of the Thread stack and network are as follows:

- Simple network installation, start up and operation – The simple protocols for forming, joining and maintaining Thread Networks allow systems to self-configure and fix routing problems as they occur.
- Secure – Devices do not join the Thread Network unless authorized and all communications are encrypted and secure.
- Small and large networks – Home networks vary from several devices to hundreds of devices communicating seamlessly. The Thread networking layer is designed to optimize the network operation based on the expected use.
- Range – Typical devices, in conjunction with mesh networking, provide sufficient range to cover a normal home. Spread spectrum technology is used at the physical layer to provide good immunity to interference.
- No single point of failure – The stack is designed to provide secure and reliable operations even with the failure or loss of individual devices.
- Low power – Sleepy End Devices can typically operate for several years on AA type batteries using suitable duty cycles.

The Thread standard is based on the IEEE 802.15.4 MAC and physical layer operating at 250 kbps in the 2.4 GHz band. The IEEE 802.15.4 2006 version of the specification [IEEE802154] is used for the Thread stack.

1.3.1 No Single Point of Failure

In a system comprised of devices running the Thread stack, none of these devices represents a single point of failure. While there are a number of devices in the system that perform special functions, the Thread design is such that they can be replaced without impacting the ongoing communication within the Thread Network. For example, a Sleepy End Device Child requires a Parent router for communications so this Parent represents a single point of failure for its communications. However, the Sleepy End Device can and will select another Parent Router if its Parent Router is unavailable. As a result, this transition should not be visible to the user.

While the Thread Network is designed for no single point of failure, under certain topologies there will be individual devices that do not have backup capabilities. For example, in a Thread Network Partition with a single Border Router, if the Border Router loses power, there is no means to switch to an alternative Border Router.

A router or Border Router can assume a Leader role for certain functions in the Thread Network. This Leader is required to make decisions within the Thread Network Partition. For example, the Leader assigns router addresses and allows new router requests. The Leader is elected and if the Leader fails, another router or Border Router becomes the Leader. It is this autonomous operation which ensures there is no single point of failure.

1 1.3.2 Device Types

2 The following device types are used within a Thread Network. These devices and their
3 behavior are detailed in appropriate chapters in this specification.

4 1.3.2.1 Border Routers

5 A Border Router is a specific type of router that provides connectivity from the 802.15.4
6 network to adjacent networks on other physical layers (Wi-Fi, Ethernet, etc.). Border
7 Routers provide services for devices within the 802.15.4 network, including routing services
8 for off network operations. A Thread Network typically contains one or more Border Routers.

9 1.3.2.2 Routers

10 Routers provide routing services to network devices. Routers also provide joining and
11 security services for devices trying to join the Thread Network. Routers are not designed to
12 sleep. Routers can downgrade their functionality and become REEDs (Router-Eligible End
13 Devices).

14 1.3.2.3 Router-Eligible End Devices

15 REEDs can become routers but due to the network topology or conditions these devices are
16 not acting as routers. As such, a REED is not a specific device type but a state of a routing-
17 capable device when in the Thread Network. These devices do not forward messages or
18 provide joining or security services for other devices in the network. If necessary, the
19 network manages the transition of a device from REED to router without user interaction.

20 1.3.2.4 Sleepy End Devices

21 Sleepy End Devices (SEDs) are host devices. They communicate only through their parent
22 router and cannot forward messages for other devices.

23 1.3.3 Security

24 The Thread Network is designed to provide a high level of security during the process of
25 adding devices to the network and during normal network operation. During the joining
26 process, devices MUST BE specifically authenticated and authorized, and required to
27 complete a key agreement mechanism. Once on the network, all communications are
28 secured with a network key.

29 1.3.3.1 Authentication and Key Agreement

30 The fundamental security used during the joining for authentication and key agreement is
31 an elliptic curve variant of J-PAKE (EC-JPAKE), using the NIST P-256 elliptic curve. J-PAKE is
32 a PAKE (Password Authenticated Key Exchange) with "juggling" (hence the "J"). It
33 essentially uses elliptic curve Diffie-Hellmann for key agreement and Schnorr signatures as
34 a NIZK (Non-Interactive Zero-Knowledge) proof mechanism to authenticate two peers and
35 to establish a shared secret between them based on the passphrase. Authorization is
36 implied through the entry of the specific passphrase into the commissioning device.

37 A TLS (Transport Layer Security) handshake is used for EC-JPAKE, which can be used in
38 both TLS and DTLS (Datagram Transport Layer Security). DTLS is a variant of TLS with
39 additional fields in the records to make it suitable for use over an unreliable datagram-

1 based transport (for example, UDP (User Datagram Protocol)), whereas TLS assumes a
2 reliable transport such as TCP (Transport Control Protocol).

3 **1.3.3.2 Network-wide Key**

4 The Thread Network is protected with a network-wide key, from which derived keys are
5 used at the MAC and MLE sub-layers to protect the 802.15.4 MAC data frames and MLE
6 messages. This is an elementary form of security used to prevent casual eavesdropping and
7 targeted disruption of the Thread Network from outsiders without knowledge of the
8 network-wide key. Because it is a network-wide key, compromise of any Thread device
9 could potentially reveal the key. As a result, the network-wide key is not typically used as
10 the only form of protection within the Thread Network. From the point of view of joining the
11 network, it is used to discriminate between an authenticated and authorized Thread device
12 and the joining device (in its initial state).

13 Because the joining device is untrusted at the point of joining, it is common practice to
14 enforce some sort of policing mechanism to ensure the joining device can be verified and at
15 the same time limit the effect of rogue devices attempting to join the Thread Network. In a
16 Thread Network, this requires the joining device to identify a trusted device and to
17 communicate solely in a point-to-point fashion with this trusted device. The trusted device
18 polices any traffic from the joining device and forwards it to the commissioning device in a
19 controlled manner to allow the authentication protocol (DTLS handshake) to execute.

20 In the usual case where the commissioning device is not in direct communication with the
21 joining device, the trusted device MUST relay the DTLS handshake with the commissioning
22 device through one or two relay agents. A relay protocol provides encapsulation of the DTLS
23 handshake and relaying of the DTLS handshake from the joining device all the way to the
24 commissioning device in a simple manner.

CHAPTER 2 SUPPORTING INFORMATION

Contents

2.1 Requirements Language	2-2
2.2 Normative References.....	2-2
2.3 Informative References	2-5
2.4 Terms and Definitions	2-5
2.5 Nomenclature for Tables Referencing Other Standards....	2-9
2.6 Acronyms	2-9

2.1 Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as described in [\[RFC 2119\]](#).

2.2 Normative References

[\[AES\]](#) National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)," FIPS 197, November 2001.

[\[CCM\]](#) National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality," SP 800-38C, May 2004.

[\[IEEE802154\]](#) Institute of Electrical and Electronics Engineers, "Wireless Personal Area Networks," IEEE Standard 802.15.4-2006, 2006.

[\[RFC 768\]](#) Postel, J., "User Datagram Protocol," August 1980.

[\[RFC 793\]](#) Information Sciences Institute, "Transmission Control Protocol," September 1981.

[\[RFC 1122\]](#) Braden, R., "Requirements for Internet Hosts -- Communication Layers," October 1989.

[\[RFC 1902\]](#) Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)," January 1996.

[\[RFC 1918\]](#) Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets," February 1996.

[\[RFC 2080\]](#) Malkin, G. and R. Minnear, "RIPng for IPv6," January 1997.

[\[RFC 2119\]](#) Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997.

[\[RFC 2131\]](#) Droms, R., Ed., "Dynamic Host Configuration Protocol," March 1997.

[\[RFC 2460\]](#) Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," December 1998.

[\[RFC 2464\]](#) Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks," December 1998.

[\[RFC 2473\]](#) Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification," December, 1998.

[\[RFC 2863\]](#) McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB," June 2000.

[\[RFC 2892\]](#) Tsiang, D. and G. Suwala, "The Cisco SRP MAC Layer Protocol," August 2000.

[\[RFC 3306\]](#) Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses," August 2002.

[\[RFC 3315\]](#) Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," July 2003.

- 1 [\[RFC 3633\]](#) Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration
2 Protocol (DHCP) version 6," December 2003.
- 3 [\[RFC 3646\]](#) Droms, R. Ed., "DNS Configuration options for Dynamic Host Configuration
4 Protocol for IPv6 (DHCPv6)," December 2003.
- 5 [\[RFC 4086\]](#) Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for
6 Security," June 2005.
- 7 [\[RFC 4191\]](#) Draves, R. and D. Thaler, "Default Router Preferences and More-Specific
8 Routes," November 2005.
- 9 [\[RFC 4193\]](#) Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses," October
10 2005.
- 11 [\[RFC 4291\]](#) Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture," February
12 2006.
- 13 [\[RFC 4443\]](#) Conta, A., Deering, S. and M. Gupta, "Internet Control Message Protocol
14 (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," March 2006.
- 15 [\[RFC 4492\]](#) Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic
16 Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)," May 2006.
- 17 [\[RFC 4615\]](#) Song, J., Poovendran, R., Lee, J., and T. Iwata, "The Advanced Encryption
18 Standard-Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-
19 CMAC-PRF-128) Algorithm for the Internet Key Exchange Protocol (IKE)," August 2006.
- 20 [\[RFC 4861\]](#) Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery
21 for IP version 6 (IPv6)," September 2007.
- 22 [\[RFC 4862\]](#) Thomson, S., Narten, T. and T. Jinmei, "IPv6 Stateless Address
23 Autoconfiguration," September 2007.
- 24 [\[RFC 4864\]](#) Van de Velde, G., Hain, T., Droms, R., and E. Klein, "Local Network Protection
25 for IPv6," May 2007.
- 26 [\[RFC 4868\]](#) Kelly, S., and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-
27 SHA-512 with IPsec," May 2007.
- 28 [\[RFC 4944\]](#) Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6
29 Packets over IEEE 802.15.4 Networks," September 2007.
- 30 [\[RFC 4949\]](#) Shirey, R., "Internet Security Glossary, Version 2," August 2007.
- 31 [\[RFC 5007\]](#) Brzozowski, J., Kinnear, K., Volz, B. and S. Zeng, "DHCPv6 Leasequery,"
32 September 2007.
- 33 [\[RFC 5116\]](#) McGrew, D., "An Interface and Algorithms for Authenticated Encryption,"
34 January 2008.
- 35 [\[RFC 5246\]](#) Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version
36 1.2," August 2008.
- 37 [\[RFC 6092\]](#) J. Woodyatt, "Recommended Simple Security Capabilities in Customer Premises
38 Equipment (CPE) for Providing Residential IPv6 Internet Service," January 2011.
- 39 [\[RFC 6106\]](#) Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement
40 Options for DNS Configuration," November 2010.
- 41 [\[RFC 6144\]](#) Baker, F., Li, X., Bao, C. and K. Yin, "Framework for IPv4/IPv6 Translation,"
42 April 2011.

- 1 [\[RFC 6145\]](#) Li, X., Bao, C. and F. Baker, "IP/ICMP Translation Algorithm," April 2011.
- 2 [\[RFC 6146\]](#) Bagnulo, M., Matthews, P. and I. van Beijnum, "Stateful NAT64: Network
3 Address and Protocol Translation from IPv6 Clients to IPv4 Servers," April 2011.
- 4 [\[RFC 6206\]](#) Levis, P., Clausen, T., Hui, J., Gnawali, O. and J. Ko, "The Trickle Algorithm,"
5 March 2011.
- 6 [\[RFC 6282\]](#) Hui, J., and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE
7 802.15.4-Based Networks," September 2011.
- 8 [\[RFC 6347\]](#) Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version
9 1.2," January 2012.
- 10 [\[RFC 6434\]](#) Jankiewicz, E., Loughney, J. and T. Narten, "IPv6 Node Requirements,"
11 December 2011.
- 12 [\[RFC 6655\]](#) McGrew, D., and D. Bailey "AES-CCM Cipher Suites for Transport Layer Security
13 (TLS)," July 2012.
- 14 [\[RFC 6690\]](#) Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format," August
15 2012.
- 16 [\[RFC 6724\]](#) Thaler, D. Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address
17 Selection for Internet Protocol Version 6 (IPv6)," September 2012.
- 18 [\[RFC 6761\]](#) Cheshire, S. and M. Krochmal, "Special-Use Domain Names," February 2013.
- 19 [\[RFC 6762\]](#) Cheshire, S. and M. Krochmal, "Multicast DNS," February 2013.
- 20 [\[RFC 6763\]](#) Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery," February 2013.
- 21 [\[RFC 6887\]](#) Wing, D. Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port
22 Control Protocol (PCP)," April 2013.
- 23 [\[RFC 6970\]](#) Boucadair, M., Penno, R., and D. Wing, "Universal Plug and Play (UPnP) IGD-
24 PCP Interworking Function," July 2013.
- 25 [\[RFC 7084\]](#) Singh, H., Beebee, W., Donley, C. and B. Stark, "Basic Requirements for IPv6
26 Customer Edge Routers," November 2013.
- 27 [\[RFC 7217\]](#) Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers
28 with IPv6 Stateless Address Autoconfiguration (SLAAC)," April 2014.
- 29 [\[RFC 7225\]](#) Boucadair, M., "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol
30 (PCP)," May 2014.
- 31 [\[RFC 7252\]](#) Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol
32 (CoAP)," June 2014.
- 33 [\[RFC 7346\]](#) Droms, R., "IPv6 Multicast Address Scopes," August 2014.
- 34 [\[RFC 7368\]](#) Chown, T. Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home
35 Networking Architecture Principles," October 2014.
- 36 [\[RFC 7488\]](#) Boucadair, M., Penno, R., Wing, D., Patil, P., and T. Reddy, "Port Control
37 Protocol (PCP) Server Selection," March 2015.
- 38 [\[RFC 7731\]](#) Hui, J. and R. Kelsey, "Multicast Protocol for Low power and Lossy Networks
39 (MPL)," February 2016.
- 40 [\[RFC 7921\]](#) Boucadair, M., Penno, R., and D. Wing, "DHCP Options for the Port Control
41 Protocol (PCP)," July 2014.

2.3 Informative References

- 1 [draft-hao-jpake-03] Hao, F., "J-PAKE: Password Authenticated Key Exchange by Juggling,"
2 September 2016.
- 3 [draft-hao-schnorr-01] Hao, F., "Schnorr NIZK Proof: Non-interactive Zero Knowledge Proof
4 for Discrete Logarithm," December 2013.
- 5 [draft-ietf-6lo-mesh-link-establishment-00] Kelsey, R., "Mesh Link Establishment,"
6 December 2015.
- 7 [RFC 4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6," April 2006.
- 8 [RFC 4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery
9 for IP version 6 (IPv6)," September 2007.
- 10 [RFC 6551] Vasseur, JP., Kim, M., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics
11 Used for Path Calculation in Low-Power and Lossy Networks," March 2012.
- 12

2.4 Terms and Definitions

13 Table 2-1 summarizes the key terms and definitions used in this specification.

14 **Table 2-1. Terms and Definitions**

Term	Definition
Anycast Locator (ALOC)	A Thread Anycast Locator (ALOC) is an IPv6 Anycast Address as defined in Section 5.2.2.2, Anycast Locator (ALOC), in Chapter 5, Network Layer.
Border Router	Any device capable of forwarding between a Thread Network and a non-Thread Network. The Border Router also serves as an interface point for the Commissioner when the Commissioner is on a non-Thread Network. The Border Router requires a Thread interface to perform and may be combined in any device with other Thread roles except the Joiner.
Commissioner	The currently elected authentication server for new Thread devices and the authorizer for providing the network credentials they require to join the network. A device capable of being elected as a Commissioner is called a Commissioner Candidate. Devices without Thread interfaces may perform this role, but those that have them may combine this role with all other roles except the Joiner. This device may be, for example, a cell phone or a server in the cloud, and typically provides the interface by which a human administrator manages joining a new device to the Thread Network.
Commissioner Candidate	A device that is capable of becoming the Commissioner, and either intends or is currently petitioning the Leader to become the Commissioner, but has not yet been formally assigned the role of Commissioner.

Term	Definition
End Device (ED)	A Thread device that transmits and receives unicast traffic only with a Parent device. FTDs may be a REED or a FED, MTDs may be a MED or a SED.
Endpoint Identifier (EID)	A Thread Endpoint Identifier is an IPv6 address that uniquely identifies a Thread interface within a Thread Network Partition and is independent from topology changes.
Exterior network(s)	A reference to other IP infrastructure (different from the actual Thread Network) to which a Thread Border Router participates. An exterior network can be a home local area network (LAN), a Virtual Private Network (VPN), or the wider Internet. An exterior network is generally accessed by the Border Router over an interface such as Wi-Fi, Ethernet, or cellular. The term is usually used in conjunction and opposed to the specific Thread Network Partition to which the Border Router also participates which is referred to as the Interior network.
External route	A route for an interior network IPv6 packet which must traverse a border router and be forwarded to an exterior network in order for it to reach the destination.
Frame counter	A value that is incremented with each new secured message and used to detect replayed messages.
Full End Device (FED)	An FTD acting as an FED that will always remain an ED. Unlike a REED, an FTD acting as an FED will never request to become a Router.
Full Thread Device (FTD)	A Thread device that can act as a Router, a REED, or a FED. Unlike an MTD, an FTD maintains links with neighboring Routers.
Interior network	A reference to a specific Thread Network Partition in which a Thread Border Router is active and for which it may provide prefix or address provisioning as well as packet forwarding and routing services. The term is usually used in conjunction and opposed to other IP network infrastructure different from the actual Thread Network to which the border router may participate referred to as Exterior network(s).
Joiner	The device to be added by a human administrator to a commissioned Thread Network. This role requires a Thread interface to perform and cannot be combined with another role in one device. The Joiner does not have network credentials.
Joiner Router	An existing Thread router or Router-eligible End Device (REED) on the secure Thread Network that is one radio hop away from the Joiner. The Joiner Router requires a Thread interface to perform, and may be combined in any device with other roles except the Joiner role.

Term	Definition
Leader	The device responsible for managing router ID assignment. The single distinguished device in any Thread Network Partition that currently acts as a central arbiter of network configuration state. The Leader requires a Thread interface to perform and may be combined in any device with other roles except the Joiner.
Link Cost	A positive integer indicating the cost of routing over a link; calculated from the link's incoming and outgoing Link Qualities.
Link Quality	An integer in [0, 3] that indicates the measured one-way quality of a link from a router to a neighboring router. A higher link quality indicates a more usable link, with 0 indicating that the link is non-existent or unusable.
Link-Local Scope (LL)	As defined in [RFC 4291] ; specifically, in a Thread Network, the scope boundaries of link-local scope are defined by the set of Thread interfaces that are reachable with a single radio transmission.
Mesh-Local Scope (ML)	In a Thread Network, the scope boundaries of Mesh-Local scope are defined by the set of Thread interfaces participating within the same Thread Network. For IPv6 multicast addresses, the Realm-Local scope [RFC 7346] is equivalent to Mesh-Local scope.
Minimal End Device (MED)	An MTD whose receiver is enabled all the time and that can communicate with its parent at any time. A MED does not need to explicitly synchronize with its parent to communicate.
Minimal Thread Device (MTD)	A Thread device that always acts as an ED. Unlike an FTD, an MTD does not maintain links with neighboring Routers.
Native Border Agent	A clarifying term for a device that is serving the role of Border Agent for a Native Commissioner. Such a device needs only an IEEE 802.15.4 radio [IEEE802154] to bridge between the unsecured 15.4 external neighbors and the secured Thread Network. As such, any Joiner Router MAY also serve as a Native Border Agent role.
Native Commissioner	A Commissioner or Commissioner Candidate that has the same interface used by the mesh. Unlike an On-mesh Commissioner, a Native Commissioner does not possess the Thread Network Credentials.
Network Credentials	All the security and operational parameters required for a device to be part of a Thread Network as contained in the Joiner Entrust message, including the Network Master Key, Mesh Local Prefix, etc. All devices on a Thread Network store the Network Credentials, including the Leader, Routers, and End Devices.

Term	Definition
Network Data	Aggregated data about a network's servers that is maintained and distributed by a network's leader.
on-mesh	A prefix is "on-mesh" if packets with on-mesh destinations are routed within the Thread subnet; all other packets are forwarded to a Border Router.
On-mesh Commissioner	A combined role for certain collapsed cases where the Commissioner has a 15.4 interface, and possesses the Thread Network Credentials. An On-mesh Commissioner is always both a Commissioner and its own Border Agent. An On-mesh Commissioner may also be a Joiner Router.
Router Eligible End Device (REED)	An FTD acting as an ED that may request to become a Router. A REED can have children and maintains links with neighboring Routers.
Router ID	An integer in [0, MAXIMUM_ROUTERS-1] assigned to a router as a network-wide unique identifier.
Routing Locator (RLOC)	A Thread Routing Locator (RLOC) is an IPv6 address that identifies the location of a Thread interface within a Thread Network Partition.
Server	A device that provides one or more services to a network. Border Routers are considered servers that provide a routing service.
Server Data	The data that describes a particular server's capabilities.
Sleepy End Device (SED)	An MTD whose receiver is normally disabled and that wakes periodically to communicate with its parent. A SED needs to explicitly synchronize with its parent before it can communicate.
Stable Network Data	Network data that is expected to be stable over periods of weeks to months. All devices maintain a local copy of the stable network data.
Temporary Network Data	Network data that is expected to be valid for less than MIN_STABLE_LIFETIME. Low-duty-cycle devices do not receive or maintain a local copy of the temporary network data.
Thread Network Partition	A connected group of nodes that operates independently of any other nodes in the network. Each Thread Network Partition has its own leader.
Valid	A prefix is "valid" if it is used to configure addresses within the Thread subnet.

2.5 Nomenclature for Tables Referencing Other Standards

For the tables presenting references to IEEE Standards and IETF RFCs, the requirement designation of each item from the reference document is defined using the following conventions in the "Designation" column (Table 2-2).

Table 2-2. Nomenclature for Tables Referencing Other Standards

Designation	Definition
I: Informative	Entries of the reference document with this designation are provided for information only.
N: Normative	Statements of this part of the reference document SHALL be taken as normative and mandatory parts of this Standard.
M: Modified	Statements of this part of the reference document SHALL be taken as part of this Standard (mandatory or optional), with the modifications and remarks noted under the part title.
S: Subset	Some portion of the statements in this part of the reference document SHALL be taken unmodified as part of this Standard, while other statements will be struck through.
N/R: Not Relevant	Statements of this part of the reference document SHALL be ignored for this Standard. An explanation may be given under the part title.
O: Optional	Statements of this part of the reference document SHALL be taken as optional parts of this Standard.
A: Additional	Statements with this designation are not present in the original reference document but are part of this Standard.

2.6 Acronyms

Table 2-3 defines the acronyms used in this specification.

Table 2-3. Acronyms and Definitions

Acronym	Definition
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
ALOC	Anycast Locator
ANSI	American National Standards Institute
API	Application Programming Interface
ASK	Amplitude Shift Key
BLE	Bluetooth Low Energy

Acronym	Definition
BPSK	Binary Phase-Shift Keying
CBC-MAC	Cipher Block Chaining Message Authentication Code
CCITT	Comité Consultatif International Téléphonique et Télégraphique (see ITU)
CCM	Counter with CBC-MAC
CID	Context ID
CoAP	Constrained Application Protocol
CPE	Customer Premises Equipment
CRC	Cyclic Redundancy Check
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
DHCPv6	Dynamic Host Configuration Protocol version 6
DNS	Domain Name System
DNS-SD	DNS Service Discovery
Dos	Denial of Service
DTLS	Datagram Transport Layer Security
DUID	DHCP Unique Identifier
DUID-EN	DUID Assigned by Vendor Based on Enterprise Number
DUID-LL	DUID Based on Link-layer Address
ED	End Device
EID	Endpoint Identifier
EUI-64	64-bit Extended Unique Identifier
FED	Full End Device
FTD	Full Thread Device
GPS	Global Positioning System
GTS	Guaranteed Time Slots
GUA	Global Unicast Address
HMAC-SHA256	Hash-based Message Authentication Code by using the SHA256 hash function
HNCP	HomeNet Control Protocol
IA	Identity Association
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
ID	Identifier

Acronym	Definition
IDR	Inverse Delivery Ratio
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IID	Interface Identifier
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
IS-IS	Intermediate System-to-Intermediate System
ITU	International Telecommunication Union
J-PAKE	Password Authenticated Key Exchange by Juggling
KA	Keep Alive
KEK	Key Encryption Key
kHz	kilohertz
L2TP	Layer 2 Tunneling Protocol
LAN	Local Area Network
LL16	Short Link Local Address
LoWPAN	Low power Wireless Personal Area Network
LQ	Leasequery
MAC	Media Access Control
mDNS	multicast Domain Name System
MeshCoP	Mesh Commissioning Protocol
MC	Manually Configured
MED	Minimal End Device
MIB	Management Information Base
MIC	Message Integrity Code
MLE	Mesh Link Establishment
ML-EID	Mesh-Local Endpoint Identifier
MLME	Media Access Control (MAC) Sublayer Management Entity
MLP	Mesh Local Prefix
MPL	Multicast Protocol for Low Power and Lossy Networks
MRC	Maximum Retransmission Count
MRT	Multicast Retransmission Timeout

Acronym	Definition
MTD	Minimal Thread Device
NAT	Network Address Translation
NIZK	Non-Interactive Zero-Knowledge
NIST	U.S. National Institute of Standards and Technology
NTP	Network Time Protocol
OSPF	Open Shortest Path First
OUI	Organizationally Unique Identifier as registered at IEEE
PAKE	Password Authenticated Key Exchange
PAN	Personal Area Network
PAT	Port Address Translation
PCP	Port Control Protocol
PD	Prefix Delegation
PHY	Physical Layer
PIB	PAN Information Base
PICS	Protocol implementation Conformance Statement
PPDU	Presentation Protocol Data Unit
PRBS	Pseudo Random Binary Sequence
PSKc	Pre-Shared Key for the Commissioner
PSKd	Pre-Shared Key for the Device
QPSK	Offset Quadrature Phase Shift Keying
QR code	Quick Response Code
REED	Router-Eligible End Device
RFC	Request for Comments
RIP	Routing Information Protocol
RLOC	Routing Locator
RLOC16	Last 16 bits of the RLOC address
RSSI	Received Signal Strength Indicator
SED	Sleepy End Device
SLAAC	Stateless Address Autoconfiguration
SoftAP	Software-enabled Access Point
SSCS	Service-specific Convergence Sublayer
SSID	Service Set Identifier
TCP	Transport Control Protocol

Acronym	Definition
TKL	Token Length
TLS	Transport Layer Security
TLV	Type-Length-Value
TMF	Thread Management Framework
TOS	Type of Service
TTL	Time To Live
UDP	User Datagram Protocol
UI	User Interface
ULA	Unique Local Address
URI	Uniform Resource Identifier
URT	Unicast Retransmission Timeout
USB	Universal Serial Bus
UTC	Coordinated Universal Time
VPN	Virtual Private Network
WAN	Wide Area Network
WPAN	Wireless Personal Area Network
ZKP	Zero Knowledge Proof

1 CHAPTER 3 PHY/MAC/6LOWPAN

2 Contents

3	3.1 Physical Layer.....	3-2
4	3.2 MAC Layer.....	3-2
5	3.3 6LoWPAN Adaptation Layer	3-10
6	3.4 Multiple Interfaces.....	3-13

3.1 Physical Layer

A Thread device MUST support at least one physical interface conforming to the 2450 MHz PHY (Physical layer) specifications defined in Chapter 6 of [\[IEEE802154\]](#) (see Table 3-1). If used for Thread, the physical interface SHALL be used exclusively by a set of MAC (Media Access Control), 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks), and Thread implementations as specified in the remainder of this document.

Table 3-1. Modifications and Statements to Chapter 6 of [\[IEEE802154\]](#)

Chapter / Section	Title and Remarks / Modifications	Status
6	PHY specification <ul style="list-style-type: none">Subset: Throughout the chapter, only the specifications which apply to the 2450 MHZ PHY are normative.	N,S
6.1	General requirements and definitions	N
6.2	PHY service specifications	N
6.3	PPDU format	N
6.4	PHY constants and PIB attributes	N
6.5	2450 MHZ PHY specifications	N
6.6	868/915 MHz band binary phase-shift keying (BPSK) PHY specifications	N/R
6.7	868/915 MHz band (optional) amplitude shift keying (ASK) PHY specifications	N/R
6.8	868/915 MHz band (optional) O-QPSK PHY specifications	N/R
6.9	General radio specifications	N

3.2 MAC Layer

A Thread device MUST implement a subset of [\[IEEE802154\]](#) MAC specifications as detailed in this section and in Table 3-2. The MAC layer operates as the next higher layer entity above the Thread physical interface specified in Section 3.1, **Physical Layer**. 6LoWPAN and the Thread Network layers consume the MAC services as its higher layer entities.

Any Thread device with Thread routing capability MUST be capable of acting as a MAC FFD (Full Function Device) with the set of restrictions specified below. A Thread device without

1 Thread routing capability MUST at a minimum act as a MAC RFD (Reduced Function Device)
2 but MAY implement FFD features.

3 Thread devices MUST have the following MAC capabilities:

- 4 • Employing the CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance) mechanism for channel access
- 5 • Scanning using active scan and Energy Detect (ED) scan types
- 6 • Generating and receiving MAC data and acknowledgement frames
- 7 • Generating Data request and Beacon request MAC command frames
- 8 • Generating MAC Beacon frames when Beacon requests are received by an FFD
- 9 • Providing a reliable link between two MAC entities
- 10 • Using MAC frame security based on the PIB security material attribute settings described in Section 7.2, MAC Security, in Chapter 7, Security.

13 Thread devices SHALL NOT be permitted to use the following MAC features when operating over the Thread physical interface:

- 15 • Periodic beacon-enabled mode of operation or synchronizing to network beacons
- 16 • GTS (Guaranteed Time Slots) mechanism
- 17 • Generating or interpreting these command frames: Association Request, Association Response, Disassociation Request, PAN ID Conflict, Orphan Notification, and Coordinator Realignment
- 18 • FFD PAN (Personal Area Network) Coordinator mode
- 19 • Any other MAC frame security options except as allowed above

22 **Table 3-2. Modifications and Statements to Chapter 7 and Annexes of [IEEE802154]**

Chapter / Section	Title and Remarks / Modifications	Status
7	<p>MAC Sublayer Specifications</p> <ul style="list-style-type: none"> • Subset: Throughout the chapter, only the specifications that apply to required MAC features are normative. • Modified: Throughout the chapter, the specifications applying to MAC features that are not permitted SHALL be ignored. 	N,S,M
7.1	<p>MAC Sublayer service specification</p> <ul style="list-style-type: none"> • Modified: Thread devices SHOULD NOT implement the sub-clauses marked as Not Relevant. • Modified: Implementations MAY optimize and modify the internal service interface provided by the MAC to the higher layers as long as the specified device functionality is still realized. 	N,M
7.1.1	MAC data service	N

Chapter / Section	Title and Remarks / Modifications	Status
7.1.2	MAC management service	N
7.1.3	Association primitives	N/R
7.1.4	Disassociation primitives	N/R
7.1.5	Beacon notification primitive	N
7.1.6	Primitives for reading PIB attributes	N
7.1.7	GTS management primitives	N/R
7.1.8	Primitives for orphan notification	N/R
7.1.9	Primitives for resetting the MAC sublayer	N
7.1.10	Primitives for specifying the receiver enable time	N
7.1.11	Primitives for channel scanning <ul style="list-style-type: none"> • Modified: All Thread devices MUST be capable of performing ED scans and active scans. • Modified: Thread devices MUST NOT use orphan and passive scan types. Specifications referring to orphan and passive scan SHALL be ignored. 	N,M
7.1.12	Communication status primitive	N
7.1.13	Primitives for writing PIB attributes	N
7.1.14	Primitives for updating the superframe configuration	N/R
7.1.15	Primitives for synchronization with a coordinator	N/R
7.1.16	Primitives for requesting data from a coordinator	N
7.1.17	MAC enumeration description	N
7.2	MAC frame formats <ul style="list-style-type: none"> • Modified: Use of addressing mode 00 MUST NOT be used for the Destination addressing mode and Source 	N,M

Chapter / Section	Title and Remarks / Modifications	Status
	Addressing mode subfields (7.2.1.1.6 and 7.2.1.1.8) when the Frame Type is Data or MAC Command.	
7.3	MAC command frames <ul style="list-style-type: none"> • Modified: Thread devices MUST be capable to generate the Data Request and Beacon Request command frames when indicated to the MAC by the higher layers. • Modified: Thread devices MUST NOT generate MAC command frame of any other types. 	N,M
7.3.1	Association request command	N/R
7.3.2	Association response command	N/R
7.3.3	Disassociation notification command	N/R
7.3.4	Data request command	N
7.3.5	PAN ID conflict notification command	N/R
7.3.6	Orphan notification	N/R
7.3.7	Beacon request <ul style="list-style-type: none"> • Modified: Thread devices acting as RFDs MUST support the generation of the Beacon request command frame. 	N,M
7.3.8	Coordinator realignment	N/R
7.3.9	GTS request	N/R
7.4	MAC constants and PIB attributes <ul style="list-style-type: none"> • Modified: A Thread device MAY omit support for constants and PIB attributes related strictly to MAC functionality that is marked as Not Relevant or is not permitted. 	N,M
7.4.1	MAC constants <ul style="list-style-type: none"> • Modified: For improved privacy and security, the value of the <i>aExtendedAddress</i> constant marked as "Device specific" in [IEEE802154] is to be provisioned 	N,M

Chapter / Section	Title and Remarks / Modifications	Status
	<p>for a Thread device in relation to the factory-assigned global IEEE EUI-64 value for the Thread interface as follows:</p> <ul style="list-style-type: none"> ▪ When the Thread device is configured to obtain Thread Network security credentials using Thread Commissioning as specified in Chapter 8, Mesh Commissioning Protocol, the specific value provided by the device to the MAC sub-layer to represent the <i>aExtendedAddress</i> prior to obtaining the network security credentials MUST be the first 64 bits of the result of computing SHA-256 over the universally administered IEEE EUI-64 value that has been factory assigned to the Thread interface instead of the actual IEEE EUI-64 value. ▪ When the Thread device is configured to obtain Thread Network security credentials using an out-of-band method instead of Thread Commissioning as specified in Chapter 8, Mesh Commissioning Protocol, the specific value provided by the device to the MAC sub-layer to represent the <i>aExtendedAddress</i> prior to obtaining the network security credentials MAY be the factory-assigned IEEE EUI-64 value. ▪ After the Thread Network security credentials have been successfully obtained using either Thread Commissioning or out-of-band pre-configuration, the specific value provided by the device to the MAC sublayer to represent the <i>aExtendedAddress</i> MUST be a random generated 64-bit value different from the universally administered IEEE EUI-64 value that has been factory assigned to the Thread interface. The random value SHALL be generated from a true random number generation source to ensure a high probability of uniqueness. ▪ If the 64-bit value resulting after either SHA-256 computation or random generation has the second least significant bit of octet 0 (also called U/L or X bit in IEEE EUI-64 terminology) set to 0, the value provided to the MAC sub-layer as the <i>aExtendedAddress</i> SHALL be modified to have the respective bit set to 1 in order to adhere to a locally administered EUI-64 format. 	
7.4.2	<p>MAC PIB attributes</p> <ul style="list-style-type: none"> • Modified: The <i>macBeaconOrder</i> PIB attribute is read-only and set to value '15'. 	N,M

Chapter / Section	Title and Remarks / Modifications	Status
	<ul style="list-style-type: none"> • Modified: The <i>macMaxFrameRetries</i> PIB attribute is read-only and set to value '3'. • Modified: The <i>macPromiscuousMode</i> PIB attribute is read-only and set to value 'FALSE'. 	
7.5	MAC functional description <ul style="list-style-type: none"> • Modified: A Thread device MAY disable MAC functionality marked as Not Relevant and SHALL disable MAC functionality implementing features that are not permitted. 	N,M
7.5.1	Channel access <ul style="list-style-type: none"> • Subset: Superframe structures MUST NOT be used. 	N,S
7.5.1.1	Superframe structure	N/R
7.5.1.2	Incoming and outgoing superframe timing	N/R
7.5.1.3	Interframe spacing	N
7.5.1.4	CSMA-CA algorithm	N
7.5.2	Starting and maintaining PANs <ul style="list-style-type: none"> • Modified: Functionality is modified as specified by the sub-clauses below. 	N,M
7.5.2.1	Scanning through channels <ul style="list-style-type: none"> • Modified: Thread devices MUST NOT perform passive or orphan channel scans. 	N,M
7.5.2.1.1	ED channel scan	N
7.5.2.1.2	Active channel scan	N
7.5.2.1.3	Passive channel scan	N/R
7.5.2.1.4	Orphan channel scan	N/R
7.5.2.2	PAN identifier conflict resolution	N/R
7.5.2.3	Starting and realigning a PAN	N/R

Chapter / Section	Title and Remarks / Modifications	Status
7.5.2.4	Beacon generation <ul style="list-style-type: none"> • Modified: The address used in the Source Address field of the beacon frame generated by a Thread device SHALL contain the value of <i>aExtendedAddress</i> for all values of <i>macShortAddress</i>. 	N,M
7.5.2.5	Device discovery	N/R
7.5.3	Association and disassociation	N/R
7.5.4	Synchronization <ul style="list-style-type: none"> • Modified: synchronization using periodic beacons and orphaned device realignment MUST NOT be used. 	N,M
7.5.4.1	Synchronization with beacons <ul style="list-style-type: none"> • Subset: only the incoming beacon processing rules that apply for a nonbeacon-enabled mode are normative. 	N,S
7.5.4.2	Synchronization without beacons	N
7.5.4.3	Orphaned device realignment	N/R
7.5.5	Transaction handling	N
7.5.6	Transmission, reception, and acknowledgment	N
7.5.6.1	Transmission	N
7.5.6.2	Reception and rejection	N
7.5.6.3	Extracting pending data from a coordinator	N
7.5.6.4	Use of acknowledgments and retransmissions	N
7.5.6.5	Promiscuous mode	N/R
7.5.6.6	Transmission scenarios	N
7.5.7	GTS allocation and management	N/R

Chapter / Section	Title and Remarks / Modifications	Status
7.5.8	Frame security	N
7.6	Security suite specifications <ul style="list-style-type: none"> • Subset: Thread devices MAY implement strictly the security functionality needed to use the specific PIB security material configuration described in Section 7.2, "MAC Security" in Chapter 7, Security. 	N,S
7.6.1	PIB security material <ul style="list-style-type: none"> • Modified: The PIB security material content SHALL be restricted to the values described in Sections 7.2.1, MAC Default Key Source, 7.2.2, MAC Key Table, 7.2.3, MAC Device Table, and 7.2.4, MAC Security Level Table, in Chapter 7, Security. 	N,M
7.6.2	Auxiliary security header <ul style="list-style-type: none"> • Modified: Auxiliary security header field values SHALL be restricted to the configuration described in Section 7.2.5, MAC Auxiliary Security Header Format in Chapter 7, Security. 	N,M
7.6.3	Security operations	N
7.7	Message sequence charts illustrating MAC-PHY interaction <ul style="list-style-type: none"> • Modified: The following figures SHALL be ignored: Figure 78—PAN start message sequence chart—PAN coordinator; Figure 80—Association message sequence chart—device; Figure 81—Association message sequence chart—coordinator; Figure 82—Passive scan message sequence chart; Figure 86—Orphaned device realignment message sequence chart. 	N,M
Annex A	Service-specific convergence sublayer (SSCS)	N/R
Annex B	CCM* mode of operation	N
Annex C	Test vectors for cryptographic building blocks	N
Annex D	Protocol implementation conformance statement (PICS) proforma	N/R

Chapter / Section	Title and Remarks / Modifications	Status
Annex E	Coexistence with other IEEE standards and proposed standards	I
Annex F	IEEE 802.15.4 regulatory requirements	I

3.3 6LoWPAN Adaptation Layer

- A Thread device MUST implement [\[RFC 4944\]](#) and [\[RFC 6282\]](#) (see Table 3-3).
 6LoWPAN (IPv6 over Low-power Wireless Personal Area Networks) is used as an adaptation layer to fragment and reassemble IPv6 packets to and from IEEE 802.15.4 MAC data frame payloads that may be smaller in size.
 In addition, the Mesh Header option as described in Chapter 11 of [\[RFC 4944\]](#) is used in conjunction with the Thread Routing mechanisms detailed in this specification to provide optimized IP mesh routing with link layer forwarding for packet transmissions within the Thread Network. For more information, see Chapter 5, Network Layer.

Table 3-3. Modifications and Statements to [\[RFC 4944\]](#)

Chapter / Section	Title and Remarks / Modifications	Status
1.	Introduction	I
1.1.	Requirements Notation	N
1.2.	Terms Used	N
2.	IEEE 802.15.4 Mode for IP	N
3.	Addressing Modes <ul style="list-style-type: none"> • Modified: Mapping of IPv6 multicast addresses to 802.15.4 16-bit multicast addresses per Section 9 is not supported. • Modified: Interfaces in Thread Networks use alternate mechanisms for hosts to learn IPv6 prefixes than specified in [RFC 4861]. 	N,M
4.	Maximum Transmission Unit	N
5.	LoWPAN Adaptation Layer and Frame Format	N
5.1.	Dispatch Type and Header <ul style="list-style-type: none"> • Modified: ESC dispatch type is redefined by [RFC 6282]. • Modified: Additional dispatch values are added by [RFC 6282]. 	N,M,S

	<ul style="list-style-type: none"> Subset: The following dispatch values are used by Thread interfaces as specified in [RFC 4944] and [RFC 6282]: <ul style="list-style-type: none"> 01 1xxxxxx - IPHC - LOWPAN_IPHC compressed IPv6 10 xxxxxxx - MESH - Mesh Header 11 000xxx - FRAG1 - Fragmentation Header (first) 11 100xxx - FRAGN - Fragmentation Header (subsequent) 	
5.2.	<p>Mesh Addressing Type and Header</p> <ul style="list-style-type: none"> Subset: Only short 16-bit addresses are used in conjunction with the Mesh Header. The V field and F field are set to '1'. 	N,S
5.3.	Fragmentation Type and Header	N
6.	Stateless Address Autoconfiguration	N
7.	IPv6 Link Local Address	N
8.	<p>Unicast Address Mapping</p> <p>Note: As permitted in this section of [RFC 4944], interfaces in Thread Networks use alternate mechanisms for non-multicast address mapping than those specified in [RFC 4861].</p>	N
9.	Multicast Address Mapping	N/R
10.	<p>Header Compression</p> <p>Note: In Thread Networks, the Header Compression mechanism described in [RFC 6282] is used instead.</p>	N/R
11.	<p>Frame Delivery in a Link-Layer Mesh</p> <p>Note: Packets with Mesh Addressing headers are forwarded using information from the link-layer routing table. See Section 5.10.1.3, Forwarding of Packets with Mesh Address Headers, in Chapter 5, Network Layer, for a description of how this table is populated.</p>	N
11.1.	LoWPAN Broadcast	N/R
12.	IANA Considerations	N/R
13.	Security Considerations	I
14.	Acknowledgements	I
15.	References	I
15.1.	Normative References	I
15.2.	Informative References	I
Appendix A	Alternatives for Delivery of Frames in a Mesh	I

- 1 The Header Compression algorithm described in [\[RFC 4944\]](#) is deprecated and has been
2 obsoleted by the improved Header Compression algorithm described in [\[RFC 6282\]](#). A
3 Thread device MUST implement [\[RFC 6282\]](#) (see Table 3-4).

4 **Table 3-4. Modifications and Statements to [\[RFC 6282\]](#)**

Chapter / Section	Title and Remarks / Modifications	Status
1.	Introduction	I
1.1.	Requirements Language	N
2.	Specific Updates to [RFC 4944]	N
3.	IPv6 Header Compression	N
3.1.	LOWPAN_IPHC Encoding Format	N
3.1.1.	Base Format	N
3.1.2.	Context Identifier Extension	N
3.2.	IPv6 Header Encoding	N
3.2.1.	Traffic Class and Flow Label Compression	N
3.2.2.	Deriving IIDs from the Encapsulating Header	N
3.2.3.	Stateless Multicast Address Compression	N
3.2.4.	Stateful Multicast Address Compression	N
4.	IPv6 Next Header Compression	N
4.1.	LOWPAN_NHC Format	N
4.2.	IPv6 Extension Header Compression <ul style="list-style-type: none"> • Modified: When there is a single trailing Pad1 or PadN option of 7 octets or less and the containing header is a multiple of 8 octets, the trailing Pad1 or PadN option SHOULD be elided by the compressor. 	N,M
4.3.	UDP Header Compression	N

Chapter / Section	Title and Remarks / Modifications	Status
4.3.1.	Compressing UDP Ports	N
4.3.2.	Compressing UDP Checksum	N
4.3.3.	UDP LOWPAN_NHC Format	N
5.	IANA Considerations	N
6.	Security Considerations	I
7.	Acknowledgements	I
8.	References	I
8.1.	Normative References	N
8.2.	Informative References	I

3.4 Multiple Interfaces

- 1 A Thread device MAY include multiple physical and media access control interfaces available
2 for radio frequency or wired connectivity. For more information, see Chapter 9, Border
3 Router, for a description of provisioning IP (Internet Protocol) over multiple interfaces.
4
- 5 A Thread device MAY expose more than one PHY and MAC interface adhering to IEEE
6 802.15.4 standards. However, at least one of these interface MUST comply with Section 3.1,
7 **Physical Layer**, Section 3.2, **MAC Layer**, and Section 3.3, **6LoWPAN Adaptation Layer**.

CHAPTER 4 MESH LINK ESTABLISHMENT

Contents

4	4.1 Introduction.....	4-3
5	4.2 Overview.....	4-3
6	4.2.1 Link Configuration	4-3
7	4.2.2 Parameter Dissemination	4-4
8	4.2.3 Neighbor Detection.....	4-4
9	4.3 Security Formats.....	4-4
10	4.4 Command Format.....	4-5
11	4.5 TLV Formats.....	4-6
12	4.5.1 Source Address TLV.....	4-7
13	4.5.2 Mode TLV	4-7
14	4.5.3 Timeout TLV	4-9
15	4.5.4 Challenge TLV.....	4-9
16	4.5.5 Response TLV	4-9
17	4.5.6 Link-layer Frame Counter TLV.....	4-9
18	4.5.7 Link Quality TLV.....	4-9
19	4.5.8 Network Parameter	4-10
20	4.5.9 MLE Frame Counter TLV.....	4-10
21	4.5.10 Route64 TLV.....	4-10
22	4.5.11 Address16 TLV.....	4-10
23	4.5.12 Leader Data TLV	4-10
24	4.5.13 Network Data TLV	4-11
25	4.5.14 TLV Request TLV	4-11
26	4.5.15 Scan Mask TLV.....	4-11
27	4.5.16 Connectivity TLV	4-11
28	4.5.17 Link Margin TLV	4-13
29	4.5.18 Status TLV	4-13
30	4.5.19 Version TLV	4-13
31	4.5.20 Address Registration TLV.....	4-13
32	4.5.21 Channel TLV	4-14
33	4.5.21.1 Supported Channel Pages	4-15
34	4.5.21.2 Supported Channels.....	4-15
35	4.5.22 PAN ID TLV	4-15

1	4.5.23	Active Timestamp TLV	4-16
2	4.5.24	Pending Timestamp TLV.....	4-16
3	4.5.25	Active Operational Dataset TLV	4-16
4	4.5.26	Pending Operational Dataset TLV	4-16
5	4.5.27	Thread Discovery TLV	4-17
6	4.6	Leader and Network Data	4-17
7	4.7	Network Attaching	4-17
8	4.7.1	Attaching to a Parent.....	4-17
9	4.7.2	Parent Selection	4-21
10	4.7.3	Child Update Request and Child Update Response Messages	4-21
11	4.7.4	Message Buffering for Children.....	4-22
12	4.7.5	Timing Out Children.....	4-22
13	4.7.6	Child Synchronization after Reset	4-23
14	4.7.7	Link Synchronization.....	4-24
15	4.7.7.1	Initial Router Synchronization	4-24
16	4.7.7.2	New Router Neighbor Synchronization.....	4-25
17	4.7.7.3	Router Synchronization after Reset	4-25
18	4.7.7.4	REED and FED Synchronization	4-26
19	4.8	Operational Dataset Announcements	4-26
20	4.8.1	Processing Announcements	4-27
21	4.9	Message Transmission	4-27
22	4.10	Processing of Incoming Messages.....	4-28
23	4.11	Parameters and Constants	4-29
24	4.12	IANA Notes	4-30
25	4.12.1	Security Suites	4-30
26	4.12.2	Command Types	4-31
27	4.12.3	TLV Types	4-32
28	4.12.4	Security Considerations.....	4-36

4.1 Introduction

The configuration of individual links in IEEE 802.15.4 mesh networks falls into a gap between standards. [\[IEEE802154\]](#) provides for static point-to-point and star topologies while the routing (L3) protocols used in multi-hop mesh networks assume that the L2 links are already up and running. Effective mesh networking using IEEE 802.15.4 requires identifying, configuring, and securing usable links to neighboring devices as the network's membership and physical environment change. Newly usable links need to be identified and configured automatically, where configuration values can include link-layer addresses, transmit and receive modes, security parameters, and so forth.

Security configuration is particularly important, because IEEE 802.15.4's replay protection applies only between a joining device and the IEEE 802.15.4 coordinator by which it joins the network. Replay protection with other neighbors requires a synchronization step that is not specified by IEEE 802.15.4.

One of the most important properties of a radio link—how reliably the two neighbors can communicate—often cannot be determined unilaterally by either neighbor. Many 802.15.4 links are asymmetric, where messages traveling one way across the link are received more or less reliably than messages traveling in the opposite direction. There is a chicken-and-egg problem here. It is a waste of effort to configure a link that does not have sufficient two-way reliability to be useful, but the two-way reliability cannot be determined without exchanging messages over the link. MLE (Mesh Link Establishment) resolves this by allowing a node to periodically multicast an estimate of the quality of its links. This allows a node to determine if it has a usable radio link to a neighbor without first configuring that link.

4.2 Overview

MLE adds two capabilities to IEEE 802.15.4:

- Dynamically configuring and securing radio links
- Detecting unreliable links before any effort is spent configuring them

All MLE messages are sent using UDP (User Datagram Protocol). While UDP is not an obvious choice for a protocol used for L2 configuration, Thread chose it to simplify integration of MLE into existing systems.

4.2.1 Link Configuration

Link configuration is done using link-local unicasts to exchange IEEE 802.15.4 radio parameters (addresses, node capabilities, and frame counters) between neighbors. Link configuration messages are either a request that the link be configured, or an acceptance or rejection of such a request.

IEEE 802.15.4 security uses frame counters to detect replayed messages. MLE uses a two-message challenge and response protocol to ensure that the MLE message containing a neighbor's frame counter is not itself a replayed message.

4.2.2 Parameter Dissemination

Devices receive a variety of link local data from MLE messages. A device that does not have the current network values, either because it has just joined the network or for any other reason, can send a unicast request to a neighbor. The neighbor will respond by sending the current network values.

4.2.3 Neighbor Detection

IEEE 802.15.4 radio links can be asymmetric, a link between neighboring devices may be much more reliable in one direction than in the other. This limits the usefulness of unilateral link quality detection: a link that looks strong to one device may not be usable because it works poorly in the other direction. To avoid wasting effort configuring unusable links, devices can use MLE to send link-local multicasts containing their local link quality estimates. Neighboring nodes can then form an estimate of the two-way quality of their link to the sender.

4.3 Security Formats

One of the main functions of MLE is to initialize link-layer security. This means that MLE itself cannot rely on link-layer security. To avoid the cost and complexity of adding a second security suite, MLE reuses [\[AES\]](#) in Counter with CBC-MAC Mode [\[CCM\]](#) as described in [\[IEEE802154\]](#).

An MLE message begins with a single byte indicating the security suite used in that message. If that initial byte is "255", no security is used and the message has no additional security data. An initial byte of '0' indicates that the message is secured (encrypted and authenticated) as described in [\[IEEE802154\]](#).

As a result, MLE messages have the following format illustrated in Figure 4-1.

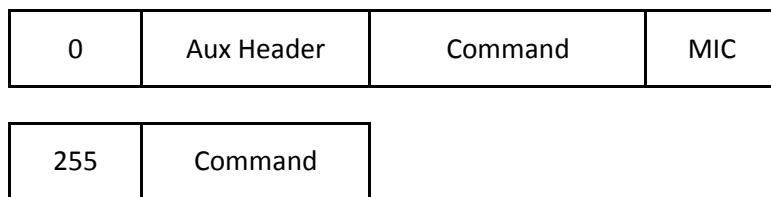


Figure 4-1. MLE Message Format

Aux Header

Auxiliary Security Header is described in [IEEE802154].

Command

For more information on the MLE command format, see Section 4.4, [Command Format](#).

MIC

Message Integrity Code is described in [IEEE802154](#).

- 1 MLE security MUST NOT use any key that is being used by the link (or any other) layer.
2 [\[CCM\]](#) requires that each key and nonce pair be used exactly once, which is most easily
3 achieved by using different keys.
- 4 Each device MUST maintain an outgoing MLE frame counter for use in securing outgoing
5 packets in compliance with [\[CCM\]](#). This MAY be the same frame counter used for securing
6 802.15.4 frames. The outgoing MLE frame counter MUST be handled as required by [\[CCM\]](#).
7 In particular, frame counters MUST NOT be reused for any given key; if the outgoing MLE
8 frame counter reaches its maximum value (0xFFFFFFFF), secured MLE messages MUST NOT
9 be sent until a new key is available, at which point the outgoing MLE frame counter MAY be
10 set back to zero.

4.4 Command Format

MLE messages consist of a command type and a series of TLV (Type-Length-Value) parameters as illustrated in Figure 4-2.



Figure 4-2. MLE Message Command Format

Command Type

An eight-bit unsigned integer identifying the type of message. Table 4-1 defines the command types in this chapter.

Table 4-1. MLE Message Command Types

Command Type	Definition
0 Link Request	A request to establish a link to a neighbor.
1 Link Accept	Accept a requested link.
2 Link Accept and Request	Accept a requested link and request a link with the sender of the original request.
3 Link Reject	Reject a link request.
4 Advertisement	Inform neighbors of network information and a device's link state.
5 Update	Not used in Thread Networks.
6 Update Request	Not used in Thread Networks.
7 Data Request	A request, typically containing a TLV Request TLV that indicates which TLV(s) are being requested.
8 Data Response	A response to a request, containing whatever TLVs were requested.
9 Parent Request	A multicast request used to find neighboring devices that can act as a Parent.
10 Parent Response	Response to Parent Request, identifying a potential Parent.

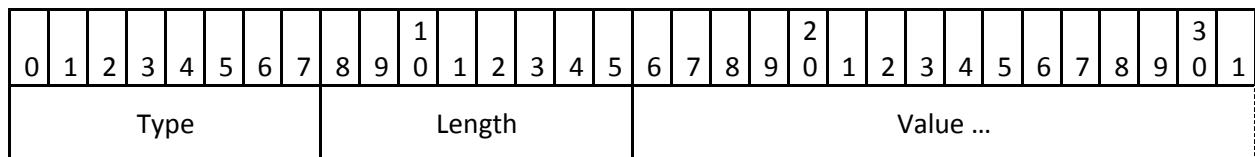
Command Type	Definition
11 Child ID Request	Request for a Child ID sent by a device to a Router or Router-Eligible End Device (REED).
12 Child ID Response	Response from a Router to a device assigning it a 16-bit network ID.
13 Child Update Request	Request by Child to update parameters on Parent.
14 Child Update Response	Response from Parent on Child request to update parameters.
15 Announce	A multicast message used to notify neighboring devices of the Thread Network's current Channel, PAN ID, and Active Timestamp.
16 Discovery Request	A multicast message used to discover networks.
17 Discovery Response	Response from a device on the Thread Network to a Discovery request.

- 1 The first four commands—Link Request, Link Accept, Link Accept and Request, and Link
2 Reject—are collectively referred to as **link configuration messages**.

3 **4.5 TLV Formats**

4 Values are encoded using a TLV format, where the type and length are one byte each and
5 the length field contains the length of the value in bytes. TLVs are stored serially with no
6 padding between them. They are byte-aligned but are not aligned in any other way such as
7 on 2- or 4-byte boundaries. All values in TLVs are in network-byte order. Unless otherwise
8 stated, values are unsigned integers. Signed integers are represented in two-complement
9 notation. There are two TLV formats—a base format and an extended format.

10 Figure 4-3 defines the base TLV format.



11 **Figure 4-3. Base TLV Format**

12 Type

13 An eight-bit unsigned integer giving the type of the value, as defined in [Section 4.12.2, Command Types](#).
14

15 Length

16 An eight-bit unsigned integer giving the length of the Value field from 0 to 254 octets. The
17 value of 255 is reserved to indicate a 16-bit length following. The lengths of the Type and
18 Length fields are not counted in the Length field.

19 Value

20 Octets of value, formatted as defined for the Type.

1 Figure 4-4 defines the extended format that allows for longer TLVs.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type										1	1	1	1	1	1	1	Length										Value ...							

2 **Figure 4-4. Extended TLV Format**

3 Type

4 An eight-bit unsigned integer giving the type of the value, from IANA registry. For more
5 information, see [Section 4.12, IANA Notes](#).

6 Length

7 A 16-bit unsigned integer giving the length of the Value field in octets. The lengths of
8 the Type, 0b11111111 escape octet and Length fields are not counted in the Length
9 field.

10 Value

11 Octets of value, formatted as defined for the Type.

12 With the exceptions of the Source Address TLV and Parameter TLV, an MLE message MUST
13 NOT contain two or more TLVs of the same type. To allow devices to have multiple source
14 addresses, an MLE message MAY contain two or more Source Address TLVs.

15 To allow for future extensions, when processing incoming TLVs defined as having a fixed
16 Length in this version of the specification, then the TLV MUST NOT be discarded and the
17 Value field(s) spanning the specified fixed Length MUST still be processed as valid even
18 when the actual incoming TLV Length is larger than the specified fixed Length.

19 **4.5.1 Source Address TLV**

20 The Source Address TLV (TLV Type 0) has a Value containing the sender's 16-bit MAC
21 address. This TLV MUST be included whenever the sender has a valid 16-bit MAC address.

22 **4.5.2 Mode TLV**

23 The Mode TLV (TLV Type 1) has a Value containing a byte string representing the mode in
24 which this link is used by the source of the message. Figure 4-5 defines the Mode TLV
25 format.

26

0	1	2	3	4	5	6	7
Reserved	R	S	D	N			

27 **Figure 4-5. Mode TLV Format**

28 R (Receiver) on when idle

- 1 Set to '1' if the sender has its receiver on when not transmitting; otherwise, set to '0'.
- 2 Only an MTD acting as a SED will set this bit to '0'.

- 1 S (Secure) data requests
- 2 Set to '1' if the sender will use 802.15.4 [\[IEEE802154\]](#) to secure all data requests;
3 otherwise, set to '0'.
- 4 D (Device type)
- 5 Set to '1' if the sender is a Full Thread Device (FTD); set to '0' if a Minimal Thread
6 Device (MTD).
- 7 N (Network data)
- 8 Set to '1' if the sender requires the full Network Data; set to '0' if the sender only needs
9 the stable Network Data.
- 10 Reserved
- 11 Set to '0' by the sender and ignored by the receiver.

4.5.3 Timeout TLV

The Timeout TLV (TLV Type 2) has a Value containing a 32-bit unsigned integer. The value is the expected maximum interval between transmissions by the sender, in seconds. This allows the receiver to more accurately time out a link to a neighbor that polls for its incoming messages.

4.5.4 Challenge TLV

The Challenge TLV (TLV Type 3) has a Value containing a randomly-chosen byte string that is used to determine the freshness of any reply to this message. The recommendations in [\[RFC 4086\]](#) apply with regard to generation of the challenge value. The byte string MUST be at least 4 bytes in length and a new value MUST be chosen for each Challenge TLV transmitted. An important part of replay protection is determining if a newly-heard neighbor is actually present or is a set of recorded messages. This is done by sending a random challenge value to the neighbor and then receiving that same value in a Response TLV sent by the neighbor. The maximum size of the Challenge TLV is 8 bytes.

4.5.5 Response TLV

The Response TLV (TLV Type 4) has a Value containing a byte string copied from a Challenge TLV.

4.5.6 Link-layer Frame Counter TLV

The Link-layer Frame Counter TLV (TLV Type 5) has a Value containing the sender's current outgoing link-layer Frame Counter, encoded as an N-bit unsigned integer. For 802.15.4, this is a 32-bit value.

4.5.7 Link Quality TLV

The Link Quality TLV (TLV Type 6) is not used in Thread Networks.

1 **4.5.8 Network Parameter**

2 The Parameter TLV (TLV Type 7) is not used in Thread Networks.

3 **4.5.9 MLE Frame Counter TLV**

4 The MLE Frame Counter TLV (TLV Type 8) has a Value containing the sender's current
5 outgoing MLE Frame Counter, encoded as a 32-bit unsigned integer.

6 **4.5.10 Route64 TLV**

7 The Route64 TLV (TLV Type 9) is used for distributing active Router IDs and routing
8 information. For more information, see Chapter 5, Network Layer.

9 **4.5.11 Address16 TLV**

10 The Address16 TLV (TLV Type 10) contains a 16-bit MAC address. This is sent by a Parent to
11 a new Child to tell the Child the address the Parent has assigned it. It is also used to
12 reconfirm a Router neighbor's 16-bit MAC address.

13 **4.5.12 Leader Data TLV**

14 The Leader Data TLV (TLV Type 11) contains the sender's current Network Leader Data.
15 Figure 4-6 defines the Leader Data TLV format.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Partition ID																																		
Weighting				Data Version				Stable Data Version				Leader Router ID																						

16 **Figure 4-6. Leader Data TLV Format**

17 Partition ID

18 A 32-bit unsigned integer identifying the Thread Network Partition to which the sender
19 belongs.

20 Weighting

21 An 8-bit unsigned integer weighting value for the Thread Network Partition to which the
22 sender belongs. Given a choice between two or more Thread Network Partitions, a node
23 will attach itself to the one with the maximum weight. If two or more Thread Network
24 Partitions share the maximum weight, a node will join the one with the maximum
25 Partition ID.

26 Data Version

27 The sender's VN_version value. For more information, see Chapter 5, Network Layer.

- 1 Stable Data Version
- 2 The sender's VN_stable_version value. For more information, see Chapter 5, Network Layer.
- 4 Leader Router ID
- 5 The Router ID assigned to the Leader of the Thread Network Partition to which the sender belongs.

4.5.13 Network Data TLV

- 8 The Network Data TLV (TLV Type 12) contains the sender's current Network Data, encoded as TLVs as described in Section 5.18, Network Data Encoding, in Chapter 5, Network Layer.

4.5.14 TLV Request TLV

- 11 The TLV Request TLV (TLV Type 13) contains a list of TLV codes, indicating which TLVs the sender is requesting. Figure 4-7 defines the Leader Data TLV format.

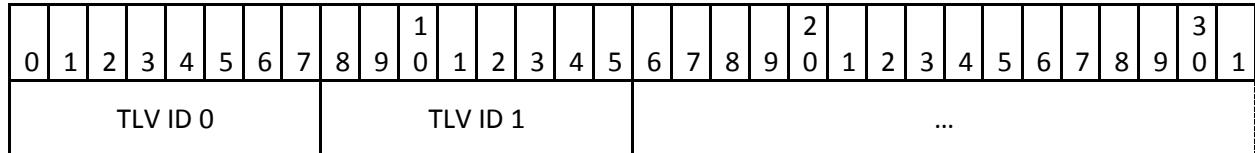


Figure 4-7. TLV Request TLV Format

4.5.15 Scan Mask TLV

- 15 The Scan Mask TLV (TLV Type 14) contains flags that indicate the types of devices that are to respond to a multicast Request. Figure 4-8 defines the Scan Mask TLV format.

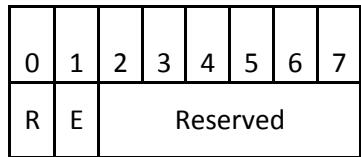


Figure 4-8. Scan Mask TLV Format

- 18 R (Router)
19 Active Routers MUST NOT respond if the R flag is not set.
- 20 E (End device)
21 REEDs MUST NOT respond if the E flag is not set.

4.5.16 Connectivity TLV

- 23 The Connectivity TLV (TLV Type 15) shows how well the sender is connected to other Routers and Children. Figure 4-9 defines the Connectivity TLV format.

1

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
PP	Reserved						Link Quality 3						Link Quality 2						Link Quality 1															
Leader Cost						ID Sequence						Active Routers						SED Buffer Size																
SED Buffer Size																SED Datagram Count																		

2 **Figure 4-9. Connectivity TLV Format**

3 PP - Parent Priority

4 2-bit signed integer indicating the priority of the sender as a parent:

5 1 (01) High

6 0 (00) Medium (default)

7 -1 (11) Low

8 -2 (10) Reserved - MUST NOT be used

9 Link Quality N

10 The number of neighboring device with which the sender shares a link of quality N. See
11 Chapter 5, Network Layer for a description of link quality.

12 Leader Cost

13 The sender's routing cost to the Leader. This field is zero if the sender is the Leader
14 itself. If the actual routing cost is infinite (normally represented as zero), no message is
15 sent.

16 ID Sequence

17 The most recent ID sequence number received by the sender.

18 Active Routers

19 8-bit unsigned integer indicating the number of active Routers in the sender's Thread
20 Network Partition.

21 SED Buffer Size

22 Optional 16-bit unsigned integer indicating the guaranteed buffer capacity in octets for
23 all IPv6 datagrams destined to a given SED.

24 SED Datagram Count

25 Optional 8-bit unsigned integer indicating the guaranteed queue capacity in number of
26 IPv6 datagrams destined to a given SED.

27 The SED Buffer Size and SED Datagram Count fields are included as a pair if the
28 corresponding Parent Request originated from a SED and either the indicated buffer size or
29 the datagram count is greater than the minimum value stated in the Thread Conformance
30 specification.

4.5.17 Link Margin TLV

The Link Margin TLV (TLV Type 16) contains the sender's calculated link margin in dB for the destination, expressed as an unsigned eight-bit value. Figure 4-10 defines the Link Margin TLV format.

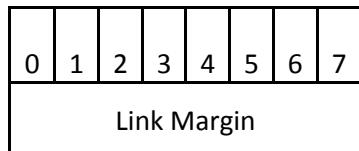


Figure 4-10. Link Margin TLV Format

4.5.18 Status TLV

The Status TLV (TLV Type 17) contains a status response to a request. Figure 4-11 defines the Status TLV format.

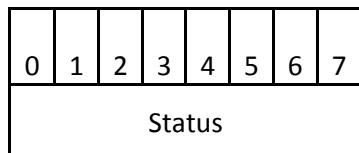


Figure 4-11. Status TLV Format

The Status byte is an unsigned integer that holds the following values:

Value	Status Meaning	Notes
1	Error	

4.5.19 Version TLV

The Version TLV (TLV Type 18) contains the version number of the Thread protocol implemented by the sender as an unsigned 16-bit integer. Figure 4-12 defines the Version TLV format.

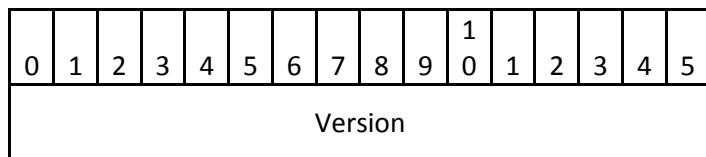


Figure 4-12. Version TLV Format

For this version of the specification (Thread 1.1), the Version TLV contains the value of '2'.

4.5.20 Address Registration TLV

The Address Registration TLV (TLV Type 19) contains zero or more addresses that have been configured by the source of the MLE message that contains it. The Address

1 Registration TLV is used for registration of valid unicast addresses. When sent by an rx-off-
2 when-idle device, the Address Registration TLV MAY contain IPv6 multicast addresses. An
3 Address Registration TLV sent by rx-off-when-idle Children serves to notify Parents that
4 IEEE 802.15.4 indirect transmissions SHOULD be used to forward packets destined to the
5 addresses contained in the Address Registration TLV.

6 IPv6 addresses can be represented in either of two forms. In the first format the full IPv6
7 address is present. In the second format the 64 bit prefix is replaced with a 6LoWPAN
8 context identifier. Both formats can be used in a single TLV. IPv6 unicast addresses can be
9 encoded using either form. However, IPv6 multicast addresses MUST be encoded using the
10 uncompressed form.

11 In the compressed form, the prefix of the IPv6 address is encoded using the appropriate
12 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) context ID for the prefix
13 along with the interface ID of the addresses. The full IPv6 address is used when the context
14 IDs are not known. The Mesh Local Prefix can always be encoded using context ID zero.

15 Figure 4-13 defines the Address Registration TLV formats.
16

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
0	Reserved																															

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
1	Re-served																															

18 **Figure 4-13. Address Registration TLV Formats**

19 IPv6 Address

20 The 16-byte IPv6 address

21 Reserved

22 Reserved bits; MUST be set to '0' (zero) on transmission and ignored upon reception.

23 CID

24 The context ID of the prefix

25 IID

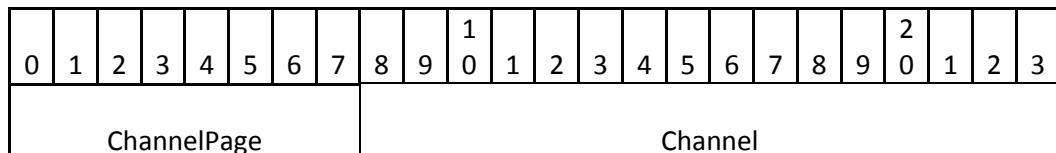
26 The 8-byte interface ID

27 **4.5.21 Channel TLV**

28 The Channel TLV (TLV Type 20) contains the channel page and channel of an adjacent
29 Thread Network Partition operating on a different Active Operational Dataset. Figure 4-14
30 defines the Channel TLV format.

31

1



2 **Figure 4-14. Channel TLV Format**

3 ChannelPage

4 An unsigned 8-bit value that specifies the channel page.

5 Channel

6 An unsigned 16-bit value that identifies the channel within the channel page.

7 **4.5.21.1 Supported Channel Pages**

8 The following channel pages are supported.

Channel Page Value	Description
0	2.4 GHz O-QPSK PHY as defined in Section 6.5 of [IEEE802154]
1 – 255	Reserved

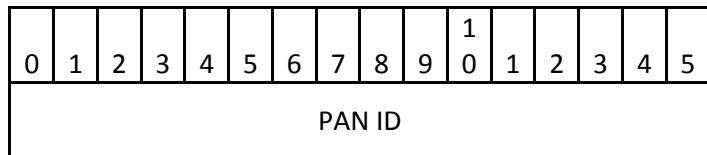
9 **4.5.21.2 Supported Channels**

10 The following channels are supported.

Channel Value	Description
0-10	Reserved
11-26	2.4 GHz channels as defined in Section 6.1.2.1 of [IEEE802154]
27 – 65535	Reserved

11 **4.5.22 PAN ID TLV**

12 The PAN ID TLV (TLV Type 21) contains the PAN ID of an adjacent Thread Network Partition operating on a different Active Operational Dataset. Figure 4-15 defines the PAN ID TLV format.



15 **Figure 4-15. PAN ID TLV Format**

16 PAN ID

17 An unsigned 16-bit value that specifies the PAN ID.

4.5.23 Active Timestamp TLV

The Active Timestamp TLV (TLV Type 22) contains an Active Timestamp. Figure 4-16 defines the Active Timestamp TLV format.

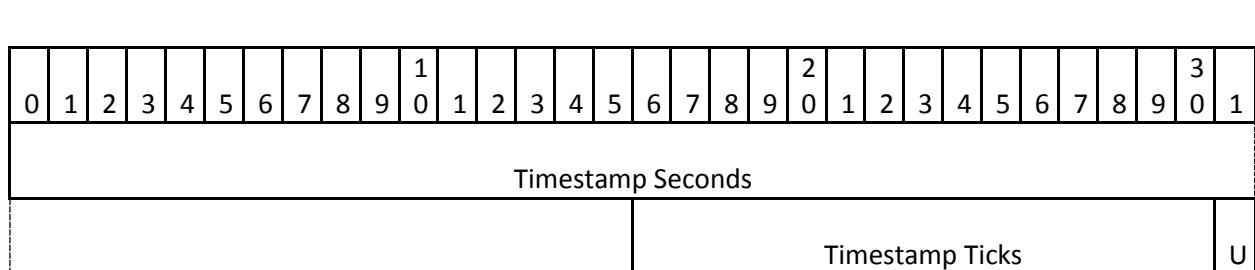


Figure 4-16. Active Timestamp TLV Format

Timestamp Seconds

A 48-bit unsigned integer that encodes a Unix Time value.

Timestamp Ticks

A 15-bit unsigned integer that encodes the fractional Unix Time value in 32.768 kHz resolution.

U bit

A 1-bit flag that indicates the time was obtained from an authoritative source: either NTP (Network Time Protocol), GPS (Global Positioning System), cell network, or other method. Note that the U bit is the least significant bit when the entire timestamp is represented as a single 64-bit number.

4.5.24 Pending Timestamp TLV

The Pending Timestamp TLV (TLV Type 23) contains a Pending Timestamp. The format of the Pending Timestamp TLV is identical to that of the Active Timestamp TLV.

4.5.25 Active Operational Dataset TLV

The Active Operation Dataset TLV (TLV Type 24) contains the sender's Active Operational Dataset encoded as a series of Network Management TLVs (see Section 8.10.1, Network Management TLVs, in Chapter 8, Mesh Commissioning Protocol). The Active Timestamp is not included, and MUST be sent using a separate Active Timestamp TLV.

4.5.26 Pending Operational Dataset TLV

The Pending Operation Dataset TLV (TLV Type 25) contains the sender's Pending Operational Dataset encoded as a series of Network Management TLVs (see Section 8.10.1, Network Management TLVs, in Chapter 8, Mesh Commissioning Protocol). The Pending Timestamp is not included and MUST be sent a separate Pending Timestamp TLV.

4.5.27 Thread Discovery TLV

The Thread Discovery TLV (TLV Type 26) contains a series of Mesh Commissioning Protocol TLVs used for Thread network discovery on IEEE 802.15.4 interfaces (see Section 8.4.4.1.1, Native Discovery Messages, in Chapter 8, Mesh Commissioning Protocol).

4.6 Leader and Network Data

Advertisement, Link Request, Link Accept, and Link Accept and Request messages sent by nodes connected to a network MUST contain a Leader Data TLV. This is used to avoid establishing a link between nodes in different Thread Network Partitions. A node MUST NOT send a Link Accept or Link Accept and Request message in response to a Request containing a Leader Data TLV whose contents do not match the node's own Leader data, except in the case of Child Link Requests during partitioning. For more information, see Chapter 5, Network Layer.

A node receiving a Data Request asking for a Network Data TLV MUST send a response containing the current Network Data.

4.7 Network Attaching

4.7.1 Attaching to a Parent

An end device attaches to a new Parent using a four-message exchange.

The sequence of messages is designed so that no device commits any resources to the exchange before receiving a Response TLV which demonstrates that the sender possesses the current MLE key. In addition to beginning the handshake with Challenge and Response TLVs, the first two messages also perform a network discovery function. This exchange assumes that the Child device knows the network's current channel and PAN ID. If it does not, it will need to use the Discovery Request message to locate devices on the network before beginning the exchange.

The full attaching handshake is as follows:

1. Child multicasts an MLE Parent Request.

This message is multicast to the Link-Local All Routers multicast address (FF02::2) with a hop limit of 255. It contains the following TLVs:

- Mode TLV
- Challenge TLV
- Scan Mask TLV

The Scan Mask TLV indicates whether only active Routers or all Routers and REEDs should respond. Connections MUST be attempted with an active Router first, with a second request sent to both Routers and REEDs (both the R and E bits set) only if the first request fails to receive a reply within

MLE_PARENT_REQ_SCANMASK_R_TIMEOUT seconds, or the replies from Routers do not contain Routers with the highest link quality. For the second Parent Request (both the R and E set), the timeout to wait for a response is MLE_PARENT_REQ_SCANMASK_RE_TIMEOUT.

- 1 • Version TLV

1 2. Routers and REEDs unicast MLE Parent Response.

2 All Routers and REEDs that receive the Child's Request and match the Scan Mask flags
3 respond with an MLE Parent Response message. The only change to the internal state of
4 the Router or REED is to record the value sent in the Challenge TLV. The Parent
5 Response contains these TLVs:

- 6 • Source Address TLV
- 7 • Leader Data TLV
- 8 • Link-layer Frame Counter TLV
- 9 • [MLE Frame Counter TLV]
- 10 • Response TLV
- 11 • Challenge TLV
- 12 • Link Margin TLV
- 13 • Connectivity TLV
- 14 • Version TLV

15 The MLE Frame Counter TLV MAY be omitted if the sender uses the same internal counter
16 for both link-layer and MLE security. The Link Margin TLV contains the estimated link margin
17 for the Child's initial Request message. The value in the Parent Priority field of the
18 Connectivity TLV SHOULD be -1 if the sender has less than 1/3 of its total child capacity
19 remaining. Otherwise, the value SHOULD be 0. Other factors MAY be used in determining
20 the value of the Parent Priority field.

21 Parent Response messages MUST be delayed by a random amount up to
22 MLE_PARENT_RSP_ROUTER_JITTER seconds if only the R flag is set and up to
23 MLE_PARENT_RSP_REED_JITTER seconds if the E flag is set. A Router or REED MUST NOT
24 send an MLE Parent Response if:

- 25 • It has no available Child capacity (if Max Child Count minus Child Count would be
26 equal to zero)
- 27 OR
- 28 • It is disconnected from its Thread Network Partition (that is, it has not received an
29 updated ID sequence number within LEADER_TIMEOUT seconds)
- 30 OR
- 31 • Its current routing path cost to the Leader is infinite.

32 A Router or REED MUST include the SED Buffer Size and SED Datagram Count fields in the
33 MLE Parent Response if the values it supports for either field are greater than the minimum
34 required by the Thread Conformance specification. A Thread Router MUST honor the SED
35 Buffer Size and SED Datagram Count for the duration that the SED is attached to it.

36 3. Child unicasts MLE Child ID Request.

37 A Thread device processes the Connectivity TLV contained in MLE Parent Responses to
38 select a Parent for attaching. The SED Buffer Size and SED Datagram Count provide
39 information about the Parent's buffering capabilities to an attaching SED. When
40 processing a Connectivity TLV with a length of 7 bytes, the SED Buffer Size and SED
41 Datagram Count are assumed to be the minimum required by the Thread Conformance
42 specification. When processing a Connectivity TLV with a length of 10 bytes, both the
43 SED Buffer Size and the SED Datagram Count are indicated in the respective fields. The
44 Child uses the link margin, parent priority, and connectivity data to select a Parent, in
45 that order. A better link margin takes precedence over a better parent priority and a
46 better parent priority has precedence over a better network connectivity. The Child then
47 unicasts to the chosen Parent an MLE Child ID Request containing the following TLVs:

- 1 • Response TLV
- 2 • Link-layer Frame Counter TLV
- 3 • [MLE Frame Counter TLV]
- 4 • Mode TLV
- 5 • Timeout TLV
- 6 • Version TLV
- 7 • [Address Registration TLV]
- 8 • [TLV Request TLV: Address16 (Network Data and/or Route)]
- 9 • [Active Timestamp TLV]
- 10 • [Pending Timestamp TLV]

11 Again, the MLE Frame Counter TLV MAY be omitted if the sender uses the same internal
12 counter for both link-layer and MLE security.

13 If the Child ID Request does not contain an MLE Active Timestamp TLV, or if it contains
14 an MLE Active Timestamp TLV that does not match the Parent's Active Timestamp, the
15 Parent MUST include an Active Operational Dataset TLV in the Child ID Response. If the
16 Parent has a valid Pending Operational Dataset and the Child ID Request does not
17 contain a MLE Pending Timestamp TLV, or if contains an MLE Pending Timestamp TLV
18 that does not match the Parent's PendingTimestamp, the Parent MUST include a Pending
19 Operational Dataset TLV in the Child ID Response.

20 MTDs MUST include an Address Registration TLV containing their ML-EID along with any
21 other addresses they wish to register with their parent. FTDs MUST NOT include an
22 Address Registration TLV.

23 If the Child has no network data, or has been out of communication for an extended
24 period, or if its stored Leader data does not match that sent by the Parent, it MUST
25 request a Network Data TLV in addition to requesting an Address16 TLV. A REED MAY
26 also request a Route64 TLV as an aid in determining whether or not it should become an
27 active Router. The sender MUST include its Active Timestamp and MUST include its
28 Pending Timestamp if it has a valid Pending Operational Dataset and maintains a valid
29 Delay Timer. If the Mode TLV indicates that the sender is an rx-off-when-idle device, it
30 MUST begin sending IEEE 802.15.4 data requests after sending the Child ID request. If
31 the selected Parent is a REED, as determined from its RLOC16, the data requests MUST
32 use the Parent's MAC Extended Address as the MAC destination. The Parent will become
33 an active router with a new RLOC16 before it sends the MLE Child ID Response.

34 4. Parent unicasts MLE Child ID Response.

35 The Parent can now safely allocate a Child ID for the new Child and add the Child to its
36 Child Table. It also includes any other TLVs requested by the Child. If the Parent device
37 is a REED, it MUST become an active Router by acquiring a Router ID before responding
38 (see Section 5.9.10, Router ID Assignment, in Chapter 5, Network Layer). The Parent
39 then sends a Child ID Response containing these TLVs:

- 40 • Source Address TLV
- 41 • Leader Data TLV
- 42 • Address16 TLV
- 43 • [Network Data TLV]
- 44 • [Route64 TLV]
- 45 • [Address Registration TLV]
- 46 • [Active Operational Dataset TLV]
- 47 • [Pending Operational Dataset TLV]

If the Child is an rx-off-when-idle device, as indicated by the contents of the Mode TLV included in the Child ID Request, this message, and all subsequent messages sent to the Child, MUST be sent as IEEE 802.15.4 indirect transmissions.

If the Child ID Request included an Address Registration TLV, the Parent echoes back any addresses other than the ML-EID that it has stored. Addresses not registered with the Parent, either because they are not (as known to the Parent) Mesh-Local addresses or because the Parent does not have sufficient storage capacity are not echoed back. The ML-EID address is always registered and does not need to be acknowledged in this fashion.

If the Child ID Request does not contain an MLE Active Timestamp TLV, or if it contains an MLE Active Timestamp TLV that does not match the Parent's Active Timestamp, the Parent MUST include an Active Operational Dataset TLV in the Child ID Response. If the Parent has a valid Pending Operational Dataset and the Child ID Request does not contain a MLE Pending Timestamp TLV, or if contains an MLE Pending Timestamp TLV that does not match the Parent's PendingTimestamp, the Parent MUST include a Pending Operational Dataset TLV in the Child ID Response.

If the Child receives a Parent Response message secured using a key sequence number that is greater than the Child's key sequence number, the Child MUST set its key sequence number to the higher value. If the Parent receives a Child ID Request message secured using a key sequence number that is greater than the Parent's key sequence number, the Parent MUST set its key sequence number to the higher value.

4.7.2 Parent Selection

An attaching device compares the Parent Responses received in response to a Parent Request to select the best candidate Parent. It uses Thread Network Partition priority, the link margin, and connectivity TLVs in the Parent Responses, as well as the RSSI of the messages, to make the choice. A REED MUST select a Parent from the highest priority Thread Network Partition (see Section 5.16.3, Merging Thread Network Partitions, in Chapter 5, Network Layer) from which a Parent Response is received. Non-REEDs ignore Thread Network Partition priority when choosing a Parent.

For a given Parent Response, the Link Margin TLV and the RSSI of the message can be used to calculate the link quality in each direction. The minimum of the incoming and outgoing link qualities is the two-way link quality. Parents with better two-way link quality are preferred. If two Parents have the same two-way link quality, an active Router is preferred over a REED. If two Parents of the same type have the same two-way link quality, the one with the most high quality links to neighbors, as indicated by the Link Quality 3 fields of the connectivity TLV, is preferred. If multiple Parents have the same number of Link Quality 3 neighbors, the Child can select any one of them as a Parent.

4.7.3 Child Update Request and Child Update Response Messages

The Child Update Request and Child Update Response messages are used to synchronize values across the link between a Parent and a Child. A Child Update Request may be sent by a Child in the following circumstances:

- To modify its mode, timeout duration, and/or address registration.
- After resetting, to verify its address assignment and obtain its Parent's frame counter(s) and current network data.

- 1 • To reset its timeout on the Parent (for rx-on-when-idle devices only).

2 A Child MUST NOT use the Child Update Request to modify its mode to one requiring
3 increased buffering capacity from the Parent compared to when the Child initially attached.
4 A Child that has attached to its Parent indicating it is an FTD MUST NOT use Child Update
5 Request to modify its mode to MTD. A Child that has attached to the Parent indicating it is
6 an rx-on-when-idle device MUST NOT use Child Update Request to modify its mode to rx-
7 off-when-idle. Instead, the Child SHOULD change to the mode requiring increased buffering
8 capacity by reattaching, either to the current Parent or another, using the process described
9 in Section 4.7.1, **Attaching to a Parent**.

10 A Child Update Request may be sent by a Parent:

- 11 • After resetting, to obtain the Child's timeout duration, address registration, and
12 frame counter(s).
13 • To send new network data to the Child.

14 Regardless of why a Child Update Request was sent, the recipient always performs these
15 same actions:

- 16 • A Parent stores any mode, timeout duration, and address registration values it
17 received and echoes them back in the response.
18 • A Child that receives Leader and network data stores them and echoes the Leader
19 data in the response.
20 • If the Child Update Request contains a Challenge TLV, the response contains a
21 Response TLV along with the sender's current frame counter(s).
22 • Any TLVs listed in a TLV Request TLV in the Child Update Request are included in the
23 response. A Child MAY request Address16, Leader Data, and Network Data TLVs. A
24 Parent MAY request Timeout and Address Registration TLVs.
25 • A Parent that receives a Child Update Request resets the sender's timeout timer.

26 A Child MUST include a Mode TLV in every Child Update Request and a Leader Data TLV if it
27 has valid Leader data. A Parent MUST include a Source Address TLV and Leader Data TLV in
28 every Child Update Request and Child Update Response.

29 The only exception to the above is that a Parent MAY send a Child Update Request or Child
30 Update Response containing only a Source Address TLV and a Status TLV containing the
31 value 'error'. This message informs the Child that the sender is no longer its Parent. This
32 message is sent by a Router when it receives a Child Update or a MAC Data Request that
33 can be authenticated but the sender is not a Child of the Router.

34 **4.7.4 Message Buffering for Children**

35 Requirements on Parent buffering of Child messages, such as minimum payload sizes and
36 hold times, are detailed in the Thread Conformance specification.

37 **4.7.5 Timing Out Children**

38 Children send periodic keep-alive messages to their Parents. If a Parent does not receive a
39 keep-alive message within a Child's timeout period, the Child is considered to be no longer
40 present and the Parent can reclaim any resources allocated to the Child.

41 A Child specifies its timeout period in the Timeout TLV included in the Child ID Request sent
42 to the Parent. Parents maintain a separate timeout timer for each Child. This timer is reset

1 when a keep-alive message is received from the Child. If the timer reaches the value of the
2 Timeout TLV sent by the Child, that Child is considered to have left and the Parent MUST
3 stop responding to Address Query or other messages on the Child's behalf. It MAY also
4 reclaim any resources (table entries, buffers, and so on) that have been allocated to the
5 Child. Non-sleepy End Devices SHOULD use a timeout of MLE_END_DEVICE_TIMEOUT. They
6 MAY use another value for application or operational reasons.

7 If a Child does not receive a response to a keep-alive message from their Parent, the Child
8 should retry three times before attempting to reattach to a new Parent. All four attempts
9 MUST be completed within the child's timeout period. The keep-alive message for an rx-off-
10 when-idle Child is an 802.15.4 MAC Data Request command. Rx-off-when-idle Children
11 SHOULD also periodically (at time intervals indicated by the application) resynchronize by
12 sending a Child Update Request to their Parent to avoid cases when the MAC Data Request
13 is being acknowledged erroneously by a receiver that is not a valid Parent. The keep-alive
14 message for an rx-on-when-idle Child is a Child Update Request containing the following
15 TLVs:

- 16 • Mode TLV
- 17 • Source Address TLV
- 18 • Leader Data TLV
- 19 • [Address Registration TLVs]

20 The Address Registration TLVs are a complete list of any non-link-local, non-mesh-local
21 addresses that the Child has configured.

22 If a Child receives 802.15.4 status NO_ACK for FAILED_CHILD_TRANSMISSIONS
23 consecutive unicast MCPS-DATA.requests to its Parent, it MUST treat this as if the Parent
24 had timed out and attempt to reattach.

25 If a Parent receives 802.15.4 status NO_ACK for FAILED_CHILD_TRANSMISSIONS
26 consecutive unicast MCPS-DATA.requests to a Child, the link is considered down and the
27 Parent MUST stop responding to Address Query or other messages on the Child's behalf.
28 However, it MUST also retain its state for the Child until the actual timeout, so as to be able
29 to respond appropriately if the Child reappears before then.

30 **4.7.6 Child Synchronization after Reset**

31 If a device Child resets or otherwise allows an unknown period of time to pass, it must
32 verify its relationship with its Parent. This is done by sending a Child Update Request
33 containing the Child current state including a Mode TLV and, if relevant, an Address
34 Registration TLV. A sleepy Child must then resume polling.

35 There are three possible outcomes:

- 36 • The Parent sends an MLE Child Update Response echoing the Child's data back. The
37 link is then established and the Child can resume its normal functioning.
- 38 • The Parent sends an MLE Child Update Response containing the Status TLV with a
39 status of "Error". The Parent is present but no longer considers the Child its Child.
40 The Child MUST then reattach, either to this Parent or another, using the process
41 described in Section 4.7.1, **Attaching to a Parent**.
- 42 • No response is received, the Child SHOULD then resend the Child Update Request. If
43 no response is received after a total of 3 transmissions, the Child MUST then
44 reattach, either to this Parent or another, using the process described in Section
45 4.7.1, **Attaching to a Parent**.

1 If a reattachment is unsuccessful (no new Parent is found), the Child MUST assume the
2 network is not currently present. The device MAY then make further attempts at
3 reattaching, or take some other application-dependent action.

4 4.7.7 Link Synchronization

5 Link synchronization is done using a two-message exchange for each direction in a link, a
6 Link Request message sent by one device followed by a Link Accept sent in reply. As an
7 optimization, bidirectional synchronization can be done in three messages, where the
8 second device's Link Accept reply can be combined with a Link Request as a single Link
9 Accept and Request message. If the recipient of a Link Request message has a valid frame
10 counter for the sender, it sends a Link Accept in reply; if not, it sends a Link Accept and
11 Request.

12 Link synchronization is performed in the following circumstances:

- 13 • As part of the attachment process, described in Section 4.7.1, **Attaching to a**
Parent.
- 14 • When a REED becomes an active Router.
- 15 • When a Router receives a message from a neighboring device for which it does not
16 have a stored frame counter.
- 17 • After a hardware reset or other event that causes a loss of local data and/or allows
18 an indeterminate amount of time to pass.

20 4.7.7.1 Initial Router Synchronization

21 Initial Router synchronization uses a two-message exchange as follows:

22 1. Device sends a Multicast Link Request.

23 When it first becomes an active Router a device MUST send a Multicast Link Request
24 Message to the Link-Local All Routers multicast address (FF02::2) containing the
25 following TLVs:

- 26 • Source Address TLV
- 27 • Leader Data TLV
- 28 • Challenge TLV
- 29 • Version TLV
- 30 • TLV Request TLV: Link Margin

31 2. Device replies with a Unicast Link Accept.

32 The response is a unicast Link Accept or Link Accept And Request message containing:

- 33 • Source Address TLV
- 34 • Leader Data TLV
- 35 • Response TLV
- 36 • Link-layer Frame Counter TLV
- 37 • Version TLV
- 38 • Link Margin TLV
- 39 • [MLE Frame Counter TLV]
- 40 • [Challenge TLV]
- 41 • [TLV Request TLV: Link Margin]

1 The Challenge TLV and TLV Request TLV are included if the response is an Accept and
2 Request message. Responses to Multicast Link Requests MUST be delayed by a random time
3 interval up to MAX_RESPONSE_DELAY_TIME. The new Router uses the value in the Link
4 Margin TLV that it receives to calculate an initial outgoing quality for that neighbor. This
5 avoids the need to wait for the neighbor to send an MLE Advertisement before establishing a
6 routing cost to the neighbor.

7 **4.7.7.2 New Router Neighbor Synchronization**

8 If a Router hears from a neighboring Router for which it has no stored frame counter, it
9 performs the same exchange as in Section 4.7.7.1, **Initial Router Synchronization**, with
10 two differences:

- 11 • The Link Request is unicast instead of multicast.
- 12 • If synchronization was triggered by a message containing a Leader Data TLV whose
13 Data Version is more recent than that of the requester, new Network Data SHOULD
14 be requested after the Link Synchronization procedure. To request the new Network
15 Data, MLE Data Request is transmitted to the neighbor Router as described in
16 Section 5.15, Network Data and Propagation, in Chapter 5, Network Layer.

17 **4.7.7.3 Router Synchronization after Reset**

18 Link synchronization after reset uses a two-message exchange as follows:

19 1. Device sends a Link Request.

20 The synchronization exchange after a device resets is again similar, except that the
21 initial Link Request message does not contain a Leader Data TLV or Source Address TLV,
22 and the requested TLVs are the Address16 TLV and Route64 TLV. The Address16 TLV
23 request is used to determine if the device's 16-bit network ID is still valid. The initiating
24 Link Request contains the following TLVs:

- 25 • Challenge TLV
- 26 • Version TLV
- 27 • TLV Request TLV: Address16, Route64 TLV

28 The Link Request is multicast to the Link-Local All Routers multicast address (FF02::2).

29 2. Neighboring Routers reply with a Link Accept.

30 Neighboring Routers respond only if they have a valid Router ID assignment for the
31 sender. The Link Accept (or Link Accept and Request) response contains the following
32 TLVs:

- 33 • Source Address TLV
- 34 • Leader Data TLV
- 35 • Response TLV
- 36 • Link-layer Frame Counter TLV
- 37 • [MLE Frame Counter TLV]
- 38 • Address16 TLV
- 39 • Version TLV
- 40 • Route64 TLV
- 41 • [Challenge TLV]

42 The Challenge TLV is included if the response is an Accept and Request message.

43 Responses to multicast Link Requests MUST be delayed by a random time interval up to
44 MAX_RESPONSE_DELAY_TIME.

1 Following link synchronization, if there is newer Network Data, that MUST be requested
2 using an MLE Data Request transmitted to the neighbor as described in Section 5.15
3 Network Data and Propagation, in Chapter 5, Network Layer.

4.7.7.4 REED and FED Synchronization

5 An End Device normally only sends and receives messages via its Parent. For a non-link-
6 local multicast, which is retransmitted by every Router, requiring a REED or FED to receive
7 the multicast from its Parent is both inefficient and error-prone. For this reason, REEDs or
8 FEDs MUST process incoming Advertisements and MUST attempt to maintain one-way
9 frame-counter synchronization with at least 3 of their neighboring Routers, if present. The
10 frame counters received in the Link Accept messages MUST be stored and used for MAC
11 security processing of incoming multicast messages.

12 This one-way synchronization uses a two-message exchange as follows:

13 1. REED or FED sends a Link Request.

14 This Link Request is unicast and contains the following TLVs:

- 15 • Source Address TLV
- 16 • Leader Data TLV
- 17 • Challenge TLV
- 18 • Version TLV

19 2. Router responds with a Link Accept.

20 Any Router that receives a Link Request MUST check the RLOC16 in the Source Address
21 TLV to see if the sender is a Router, a REED, or a FED. If the sender is a Router, the
22 recipient SHOULD proceed as described in the previous sections. If the sender is a REED
23 or FED, the recipient sends a Link Accept message containing the following TLVs:

- 24 • Source Address TLV
- 25 • Response TLV
- 26 • Version TLV
- 27 • Link-layer Frame Counter TLV
- 28 • [MLE Frame Counter TLV]

29 The recipient of the Link Request does not make any change to its local state and MUST
30 NOT reply with a Link Accept and Request message as is done in the three-message
31 exchange.

32 Upon receipt of the Link Accept message, the REED or FED saves the link-layer
33 addresses and frame counters of its Router neighbor and uses them in authenticating
34 incoming multicast messages sent by that neighbor. Unicast messages arriving from the
35 neighboring Router MUST be discarded.

36 If, subsequently, no advertisement is heard from that Router neighbor for a
37 MAX_NEIGHBOR_AGE (see Table 5-8 in Chapter 5, Network Layer) period, the link-layer
38 addresses and frame counter received in the Link Accept are discarded. If this reduces
39 the number of neighboring Routers with which the REED or FED has synchronized to less
40 than 3, the REED or FED MUST then synchronize with another neighboring Router,
41 should one be available.

4.8 Operational Dataset Announcements

43 A Thread device announces its Channel and PAN ID values by sending an Announce
44 message. Announce messages contain the following TLVs:

- 1 • Channel TLV: identifies the sender's Active Operational Dataset Channel.
- 2 • Active Timestamp TLV: identifies the sender's Active Operational Dataset Timestamp.
- 3 • PAN ID TLV: identifies the sender's Active Operational Dataset PAN ID.

4 Announce messages MUST have the Destination PAN ID in the IEEE 802.15.4 MAC header
5 set to the Broadcast PAN ID (0xFFFF) and MUST be secured using Key ID Mode 2 as
6 described in Section 7.2.1.3, Key ID Mode 2, in Chapter 7, Security.

7 Announce messages MUST be secured at the MLE layer and set MLE Key Identifier Mode to
8 2. A Thread device MUST transmit Announce messages on the appropriate PHY channels as
9 specified in the MGMT_ANNOUNCE_BEGIN.ntf message.

10 **4.8.1 Processing Announcements**

11 When receiving an Announce message, the Thread device compares the received Active
12 Timestamp TLV with its locally stored value in the Active Operational Dataset. If the
13 received value is newer and the channel and/or PAN ID in the Announce message differ
14 from those currently in use, the receiving device attempts to attach using the channel and
15 PAN ID received from the Announce message.

16 If the attachment is successful, the Thread device MUST retrieve the updated Operational
17 Datasets as described in Section 8.4.3.5.1, Synchronizing Active Operational Datasets and
18 Section 8.4.3.5.2, Synchronizing Pending Operational Datasets, in Chapter 8, Mesh
19 Commissioning Protocol. Once the Thread device receives the new Active Commissioning
20 Dataset, the device MUST transmit its own Announce messages on the channel it was on
21 prior to the attachment. This transmission is done even if the channel prior to the
22 attachment is the same channel as the current one. In this case, the Thread device SHOULD
23 use the following parameters:

- 24 • Count: 3 transmissions
- 25 • Period: 1 second

26 If the received Active Timestamp is older than the locally stored value in the Active
27 Operational Dataset, the receiving device sends its own Announce message once on the
28 channel indicated in the received Channel TLV.

29 **4.9 Message Transmission**

30 MLE messages SHOULD be sent using the assigned UDP port number (19788) as both the
31 source and destination port. Link configuration and advertisement messages MUST be sent
32 with an IP Hop Limit of 255, either to a link-local unicast address or to the Link-Local All
33 Nodes (FF02::1) or all-routers (FF02::2) multicast addresses.

34 With the exception of the Discovery Request and Discovery Response all other MLE
35 messages MUST be secured using the procedure specified in [\[AES\]](#) and [\[CCM\]](#) using the
36 auxiliary security header as described in [\[IEEE802154\]](#).

37 The authenticated data consists of the following three concatenated values:

- 38 • IP source address
- 39 • IP destination address
- 40 • Auxiliary security header

41 The secured data consists of the message body following the auxiliary security header (the
42 command ID and TLVs). The security suite identifier is not included in either the
43 authenticated data or the secured data.

- 1 MLE messages that fit within a single 802.15.4 frame MUST NOT use 802.15.4 security. MLE
2 messages that require fragmentation by 6LoWPAN MUST use 802.15.4 security.
- 3 With the exception of the Discovery Response, a message sent in response to a multicast
4 request, such as a Multicast Link Request, MUST be delayed by a random time between 0
5 and MLE_MAX_RESPONSE_DELAY seconds, with a resolution of at least 1 ms.
- 6 Unless the retry sequence and delay values for a request or process are otherwise specified,
7 if no response is received to a unicast or multicast request, the request MAY be
8 retransmitted using a simple timeout mechanism. Unicast requests are not relayed, which
9 avoids the need for a more elaborate mechanism. Table 4-2 summarizes the parameters
10 used for MLE in Thread Networks.
- 11 For each transmission, the appropriate MLE_UNICAST_RETRANSMISSION_DELAY or
12 MLE_MULTICAST_RETRANSMISSION_DELAY value is multiplied by a random number chosen
13 with a uniform distribution between 0.9 and 1.1 with a resolution of at least 1 ms. The
14 randomization factor is included to minimize synchronization of messages transmitted. A
15 message MUST NOT be transmitted more than MLE_MAX_TRANSMISSION_COUNT times.

16 **4.10 Processing of Incoming Messages**

- 17 Any incoming link configuration or advertisement message, or an incoming update sent to a
18 link-local address, whose IP Hop Limit is not 255, may have been forwarded by a Router
19 and MUST be discarded.
- 20 Incoming messages whose Command Type is a reserved value MUST be ignored. Any TLVs
21 in an incoming message whose TLV Type has a reserved value MUST be ignored.
- 22 With the exception of the Discovery Request and Discovery Response, incoming messages
23 that are not secured with either MLE or link-layer security SHOULD be ignored. Secured
24 incoming messages are decrypted and authenticated using the procedures specified in [\[AES\]](#)
25 and [\[CCM\]](#), with security material obtained from the auxiliary security header as described
26 in [\[IEEE802154\]](#). The key source may be obtained either from the link layer source address
27 or from the auxiliary security header.
- 28 A device MUST maintain a separate incoming MLE frame counter for each neighbor with
29 which it establishes a link. Any MLE message received with a frame counter the same or
30 lower than that of a previously received and authenticated message from the same source
31 MUST be discarded. Messages for which no previous frame counter are available MAY be
32 processed, but their counter value MUST be saved for comparison with later messages.

4.11 Parameters and Constants

Table 4-2 lists the parameters used for MLE in Thread networks, their definitions, and their default values.

Table 4-2. MLE Parameters

Parameter Name	Definition	Default
FAILED_CHILD_TRANSMISSIONS	The number of consecutive MCPS.DATA-Confirms having Status NO_ACK that cause a Child-to-Parent link to be considered broken.	4 (up to 16 unacknowledged individual frame transmissions and retries)
MLE_UNICAST_RETRANSMISSION_DELAY	Base delay before retransmitting an MLE unicast. This value is multiplied by a random number between 0.9 and 1.1 to get the actual delay.	1 second
MLE_MULTICAST_RETRANSMISSION_DELAY	Base delay before retransmitting an MLE multicast. This value is multiplied by a random number between 0.9 and 1.1 to get the actual delay.	5 seconds
MLE_MAX_TRANSMISSION_COUNT	The maximum number of times an MLE message may be transmitted.	3
MLE_MAX_RESPONSE_DELAY	Maximum delay before responding to a multicast request.	1 second
MLE_END_DEVICE_TIMEOUT	Child timeout specified by End Devices when attaching.	240 seconds

Parameter Name	Definition	Default
MLE_PARENT_REQ_SCANMASK_R_TIMEOUT	Time an attaching node MUST wait for a Parent Response from Active Routers to respond to an MLE Parent Request sent with Scan Mask bit R set.	0.75 second
MLE_PARENT_REQ_SCANMASK_RE_TIMEOUT	Time an attaching node MUST wait for a Parent Response from Active Routers or REEDs to respond to an MLE Parent Request sent with Scan Mask R and E bits sent.	1.25 seconds
MLE_PARENT_RSP_ROUTER_JITTER	Maximum amount of random delay before sending an MLE Parent Response when only active routers are specified in the Scan Mask.	0.5 second
MLE_PARENT_RSP_REED_JITTER	Maximum amount of random delay before sending an MLE Parent Response when REEDs are included in the Scan Mask.	1.0 second

4.12 IANA Notes

- 1
- 2 IANA has assigned UDP port 19788 to MLE.
- 3 IANA will be requested to establish a new top-level registry, called "MLE: Mesh Link Establishment," to contain all MLE objects, codepoints, and sub-registries.

4.12.1 Security Suites

- 5 IANA will be requested to create a sub-registry called "Security Suites." Values range from 0 to 255. Table 4-3 summarizes the MLE Security Suites.

1

Table 4-3. MLE Security Suites

Value	Meaning	For more information
0	802.15.4 Security	This chapter
255	No Security	This chapter

2 Values 1-254 are currently unassigned.

4.12.2 Command Types

4 IANA will be requested to create a sub-registry called "Command Types." Values range from 5 0 to 255. Table 4-4 summarizes the MLE Command Types.

6 **Table 4-4. MLE Command Types**

Value	Meaning	For more information
0	Link Request	This chapter
1	Link Accept	This chapter
2	Link Accept and Request	This chapter
3	Link Reject	This chapter
4	Advertisement	This chapter
5	Update	Not used in Thread Networks
6	Update Request	Not used in Thread Networks
7	Data Request	This chapter
8	Data Response	This chapter
9	Parent Request	This chapter
10	Parent Response	This chapter
11	Child ID Request	This chapter
12	Child ID Response	This chapter
13	Child Update Request	This chapter
14	Child Update Response	This chapter
15	Announce	This chapter
16	Discovery Request	Chapter 8, Mesh Commissioning Protocol
17	Discovery Response	Chapter 8, Mesh Commissioning Protocol

7 Values 18-255 are currently unassigned.

4.12.3 TLV Types

2 IANA will be requested to create a sub-registry called "TLV Types." Values range from 0 to
3 255. Table 4-5 summarizes the MLE TLV Types.

Table 4-5. MLE TLV Types

Value	Meaning	For more information
0	Source Address TLV	This chapter
1	Mode TLV	This chapter
2	Timeout TLV	This chapter
3	Challenge TLV	This chapter
4	Response TLV	This chapter
5	Link-layer Frame Counter TLV	This chapter
6	Link Quality TLV	Not used in Thread Networks
7	Network Parameter TLV	Not used in Thread Networks
8	MLE Frame Counter TLV	This chapter
9	Route64 TLV	Chapter 5, Network Layer
10	Address16 TLV	This chapter
11	Leader Data TLV	This chapter
12	Network Data TLV	Chapter 5, Network Layer
13	TLV Request TLV	This Chapter
14	Scan Mask TLV	This chapter
15	Connectivity TLV	This chapter
16	Link Margin TLV	This chapter
17	Status TLV	This chapter
18	Version TLV	This chapter
19	Address Registration TLV	This chapter
20	Channel TLV	This chapter
21	PAN ID TLV	This chapter
22	Active Timestamp TLV	This chapter
23	Pending Timestamp TLV	This chapter
24	Active Operational Dataset TLV	This chapter
25	Pending Operational Dataset TLV	This chapter
26	Thread Discovery TLV	This chapter. Sub-TLVs are defined in Chapter 8. Mesh Commissioning Protocol.

- 1 Values 27-255 are currently unassigned.
- 2 Table 4-6 summarizes the MLE message command types and included TLVs.
- 3

Table 4-6. MLE Message Command Types and Included TLVs

Command Type	Included TLVs
Link Request	Source Address TLV Challenge TLV Leader Data TLV Version TLV TLV Request TLV: Link Margin
Link Accept	Source Address TLV Response TLV Link-layer Frame Counter TLV [MLE Frame Counter TLV] Version TLV Leader Data TLV [Challenge TLV] Link Margin TLV
Link Accept and Request	Source Address TLV Leader Data TLV Response TLV Link-layer Frame Counter TLV Version TLV Link Margin TLV [MLE Frame Counter TLV] [Challenge TLV] [TLV Request TLV: Link Margin]
Link Reject	Status TLV
Advertisement	Source Address TLV Route64 TLV Leader Data TLV
Update	Not used in Thread Networks
Update Request	Not used in Thread Networks
Data Request	TLV Request TLV (requesting Network Data TLV) Active Timestamp TLV [Pending Timestamp TLV]
Data Response	Source Address TLV Leader Data TLV [Network Data TLV]

Command Type	Included TLVs
	[Active Operational Dataset TLV] [Pending Operational Dataset TLV] [Active Timestamp TLV] [Pending Timestamp TLV]
Parent Request	Mode TLV Challenge TLV Scan Mask TLV Version TLV
Parent Response	Source Address TLV Challenge TLV Response TLV Link-layer Frame Counter TLV [MLE Frame Counter TLV] Leader Data TLV Connectivity TLV Link Margin TLV Version TLV
Child ID Request	Mode TLV Timeout TLV Response TLV Link-layer Frame Counter TLV [MLE Frame Counter TLV] Version TLV [Address Registration TLV] [TLV Request TLV] [Active Timestamp TLV] [Pending Timestamp TLV]
Child ID Response	[Route64 TLV] Address16 TLV Leader Data TLV [Network Data TLV] Source Address TLV [Address Registration TLV] [Active Timestamp TLV] [Pending Timestamp TLV] [Active Operational Dataset TLV] [Pending Operational Dataset TLV]
Child Update Request (from a Child)	Mode TLV [Source Address TLV] (if known) [Leader Data TLV] (if known)

Command Type	Included TLVs
	<ul style="list-style-type: none"> [Challenge TLV] [Timeout TLV] [Address Registration TLV] [TLV Request TLV] (may request Address16 TLV, Leader Data TLV, and/or Network Data TLV)
Child Update Request (from Parent)	<ul style="list-style-type: none"> Source Address TLV [Leader Data TLV] [Network Data TLV] [Challenge TLV] [TLV Request TLV] (may request Timeout TLV, and/or Address Registration TLV)
Child Update Request (rejection from Parent)	<ul style="list-style-type: none"> Source Address TLV Status TLV
Child Update Response (from Child)	<ul style="list-style-type: none"> [Source Address TLV] (if known) [Timeout TLV] (if requested) [Address Registration TLV] (if requested) [Response TLV] (in response to Challenge TLV) [Link-layer Frame Counter TLV] (in response to Challenge TLV) [MLE Frame Counter TLV] (in response to Challenge TLV) Leader Data TLV (optional, see Section 4.7.3, <u>Child Update Request and Child Update Response Messages</u>)
Child Update Response (from Parent)	<ul style="list-style-type: none"> Source Address TLV Mode TLV Timeout TLV (echoed from Child Update Request) Address Registration TLV (echoed from Child Update Request) [Address16 TLV] (if requested) [Leader Data TLV] (if requested) [Network Data TLV] (if requested) [Response TLV] (in response to Challenge TLV) [Link-layer Frame Counter TLV] (in response to Challenge TLV) [MLE Frame Counter TLV] (in response to Challenge TLV)
Discovery Request	Thread Discovery TLV
Discovery Response	Thread Discovery TLV

1 **4.12.4 Security Considerations**

- 2 In general, MLE has the strengths and weaknesses of the link layer security that it inherits.
3 The one exception is that MLE's operation requires accepting and acting on incoming
4 Advertisement and Link Request messages for which the receiver has no prior knowledge of
5 the sender's MLE frame counter. Because of this, implementers must be careful in how they
6 use information obtained from these possibly-replayed messages. For example, information
7 from unsecured messages SHOULD NOT be used to modify any stored information obtained
8 from secured messages.
- 9 The Hop Limit field of received packets other than multi-hop update messages is verified to
10 contain 255, the maximum legal value. Because Routers decrement the Hop Limit on all
11 packets they forward, received packets containing a Hop Limit of 255 must have originated
12 from a neighbor. This technique is borrowed from IPv6 ND (Neighbor Discovery)
13 [\[RFC 4861\]](#).

CHAPTER 5 NETWORK LAYER**Contents**

3	5.1 IPv6 Addressing.....	5-5
4	5.2 IPv6 Addressing Architecture	5-8
5	5.2.1 Address Scopes.....	5-8
6	5.2.1.1 Link-Local Scope	5-8
7	5.2.1.2 Realm-Local Scope	5-8
8	5.2.2 Unicast Addressing	5-8
9	5.2.2.1 Routing Locator (RLOC).....	5-9
10	5.2.2.2 Anycast Locator (ALOC).....	5-9
11	5.2.2.3 Endpoint Identifier (EID)	5-11
12	5.2.2.4 Link-Local Addresses.....	5-11
13	5.2.2.5 Mesh-Local Addresses	5-11
14	5.2.2.6 Global Addresses	5-12
15	5.2.3 Multicast Addressing	5-12
16	5.2.3.1 Link-Local Scope	5-12
17	5.2.3.2 Realm-Local Scope	5-13
18	5.2.3.3 Other Multicast Scopes	5-13
19	5.3 IPv6 Address Configuration	5-13
20	5.3.1 Link-Local Addresses	5-13
21	5.3.2 Mesh-Local Addresses.....	5-14
22	5.3.3 Global Addresses.....	5-14
23	5.4 EID-to-RLOC Mapping	5-14
24	5.4.1 Information Base	5-15
25	5.4.1.1 Local Address Set	5-15
26	5.4.1.2 MTD Child Address Set	5-15
27	5.4.1.3 Address Query Set.....	5-15
28	5.4.2 Address Query	5-15
29	5.4.2.1 Transmission of Address Query Messages	5-15
30	5.4.2.2 ADDR_QRY.qry – Address Query	5-16
31	5.4.2.3 Receipt of Address Query Messages	5-16
32	5.4.2.4 ADDR_NTF.ans - Address Notification.....	5-17
33	5.4.2.5 Receipt of Address Notification Messages.....	5-17
34	5.4.3 Proactive Address Notifications	5-17
35	5.4.3.1 ADDR_NTF.ntf - Proactive Address Notification	5-18
36	5.4.3.2 Receipt of Proactive Address Notifications	5-18
37	5.5 EID-to-RLOC Map Cache.....	5-18
38	5.5.1 Information Base	5-19

1	5.5.1.1	EID-to-RLOC Set	5-19
2	5.5.2	Managing EID-to-RLOC Map Cache Entries	5-19
3	5.5.2.1	Cache Errors	5-19
4	5.5.2.2	Optimizations.....	5-20
5	5.6	Duplicate IPv6 Address Detection.....	5-20
6	5.6.1	Address Queries	5-20
7	5.6.2	Proactive Address Notifications	5-20
8	5.6.3	Creation of Address Error Notifications.....	5-21
9	5.6.3.1	ADDR_ERR.ntf - Address Error Notification.....	5-21
10	5.6.4	Receipt of Address Error Notifications	5-21
11	5.7	DHCPv6 Services.....	5-22
12	5.8	ICMPv6	5-22
13	5.9	Routing Protocol	5-23
14	5.9.1	Routing Database.....	5-24
15	5.9.1.1	Router ID Set.....	5-24
16	5.9.1.2	Link Set.....	5-24
17	5.9.2	Route Set	5-24
18	5.9.3	Leader Database	5-24
19	5.9.3.1	ID Assignment Set.....	5-25
20	5.9.4	Link Margins and Link Metrics	5-25
21	5.9.5	Routing Cost and Next Hop.....	5-26
22	5.9.6	Loops and Loop Detection	5-26
23	5.9.7	Sending Advertisements.....	5-26
24	5.9.8	Processing Route64 TLVs.....	5-27
25	5.9.9	Router ID Management.....	5-27
26	5.9.10	Router ID Assignment.....	5-29
27	5.9.10.1	ADDR_SOL.req – Address Solicit Request	5-29
28	5.9.10.2	ADDR_SOL.rsp – Address Solicit Response.....	5-30
29	5.9.10.3	ADDR_REL.ntf – Address Release Notification.....	5-31
30	5.10	Unicast Packet Forwarding.....	5-31
31	5.10.1	Unicast Packet Forwarding inside the Thread Network.....	5-31
32	5.10.1.1	Full Thread Device Forwarding.....	5-31
33	5.10.1.2	Minimal Thread Device Forwarding.....	5-32
34	5.10.1.3	Forwarding of Packets with Mesh Address Headers.....	5-32
35	5.10.2	Unicast Packet Forwarding outside the Thread Network.....	5-33
36	5.11	Multicast Packets Forwarding	5-33
37	5.11.1	Link-Local Scope	5-33
38	5.11.2	Realm-Local and Larger Scopes.....	5-34
39	5.12	Selection of Link-Layer Destination Addresses.....	5-38
40	5.13	Thread Network Data	5-38

1	5.13.1	Version Number Set	5-39
2	5.13.2	Valid Prefix Set	5-39
3	5.13.3	External Route Set	5-40
4	5.13.4	6LoWPAN Context ID Set	5-40
5	5.13.5	Server Set.....	5-41
6	5.13.6	Commissioning Data Set	5-41
7	5.13.7	Provisioning Domain	5-41
8	5.14 Stable Thread Network Data	5-42	
9	5.15 Network Data and Propagation	5-43	
10	5.15.1	Modifying Network Data	5-43
11	5.15.2	Propagation to rx-on-when-idle Devices	5-43
12	5.15.3	Propagation of Operational Datasets	5-44
13	5.15.4	Propagation to rx-off-when-idle Devices	5-45
14	5.15.5	Leader and Thread Network Data	5-45
15	5.15.6	Server Behavior	5-46
16		5.15.6.1 SVR_DATA.ntf – Server Data Notification.....	5-46
17	5.15.7	Router Behavior	5-47
18	5.15.8	Host Behavior	5-48
19		5.15.8.1 ND_DATA.req (/nd) – Neighbor Discovery Data Request..	5-48
20		5.15.8.2 ND_DATA.rsp (/nd) – Neighbor Discovery Data Response	5-49
21	5.15.9	6LoWPAN Contexts.....	5-49
22	5.16 Thread Network Partitions	5-49	
23	5.16.1	Losing Connectivity	5-50
24	5.16.2	Starting a New Thread Network Partition.....	5-51
25	5.16.3	Merging Thread Network Partitions	5-51
26	5.16.4	Resetting Thread Network Partition Data	5-52
27	5.16.5	Loss of Leader	5-52
28	5.16.6	Children	5-53
29	5.17 Protocol Parameters and Constants	5-53	
30	5.18 Network Data Encoding.....	5-57	
31	5.18.1	Has Route TLV	5-58
32	5.18.2	Prefix TLV	5-59
33	5.18.3	Border Router TLV.....	5-60
34	5.18.4	6LoWPAN ID TLV.....	5-61
35	5.18.5	Commissioning Data TLV.....	5-61
36	5.18.6	Service TLV	5-61
37	5.18.7	Server TLV	5-62
38	5.19 Network Layer TLVs	5-63	
39	5.19.1	Target EID TLV	5-64
40	5.19.2	MAC Extended Address TLV	5-64

1	5.19.3	RLOC16 TLV	5-64
2	5.19.4	ML-EID TLV	5-65
3	5.19.5	Status TLV	5-65
4	5.19.6	Time Since Last Transaction TLV	5-65
5	5.19.7	Router Mask TLV	5-66
6	5.19.8	ND Option TLV	5-66
7	5.19.9	ND Data TLV.....	5-66
8	5.19.10	Thread Network Data TLV.....	5-67
9	5.20 MLE Routing TLV	5-67	
10	5.20.1	MLE Route64 TLV Format	5-67

5.1 IPv6 Addressing

A Thread-compliant device MUST implement [\[RFC 2460\]](#). A Thread device is not required to implement all the available IPv6 Network Layer features. IPv6 fragmentation and reassembly is not required for Thread devices. A Border Router MAY implement IPv6 packet reassembly.

Table 5-1 summarizes the modifications and statements to [\[RFC 2460\]](#) for Thread Networks. (See Section 2.5, Nomenclature for Tables Referencing Other Standards, in Chapter 2, Supporting Information, for an explanation of the “Status” column.)

Table 5-1. Modifications and Statements to [\[RFC 2460\]](#)

Chapter / Section	Title and Remarks / Modifications	Status
1	Introduction	I
2	Terminology	I
3	IPv6 Header Format	N
4	IPv6 Extension Headers	N,M
4.1	Extension Header Order	N
4.2	Options	N
4.3	Hop-by-Hop Options Header	N
4.4	Routing Header	N/R
4.5	Fragment Header	N/R
4.6	Destination Options Header	N
4.7	No Next Header	N
5	Packet Size Issues <ul style="list-style-type: none">• Path MTU Discovery is not required.• Only Border Routers are required to receive packets as large as 1500 bytes.• IPv6-to-IPv4 tunneling and fragmentation is not required.	M

Chapter / Section	Title and Remarks / Modifications	Status
6	Flow Labels <ul style="list-style-type: none"> Set to 0 on transmission and ignored on reception. 	N/R
7	Traffic Classes <ul style="list-style-type: none"> Set to 0 on transmission and ignored on reception. 	N/R
8	Upper-Layer Protocol Issues	M
8.1	Upper-Layer Checksums	N
8.2	Maximum Packet Lifetime	N
8.3	Maximum Upper-Layer Payload Size	N
8.4	Responding to Packets Carrying Routing Headers	N/R
Appendix A	Semantics and Usage of the Flow Label Field	N/R
Appendix B	Formatting Guidelines for Options	N

- 1 A Thread device MUST implement the IPv6 addressing architecture specified in [\[RFC 4291\]](#).
 2 A Thread device MUST support 1 link-local address, at least 2 Mesh-Local addresses (used
 3 for communication inside the Thread Network), and MAY support additional IPv6 addresses
 4 (e.g., ULA or Global Unique Addresses (GUA)) based on its available resources.
 5 The Devices MUST also support a number of link-local and realm-local multicast addresses,
 6 as defined in Section 5.2.4, **Multicast Addressing**.
 7 Table 5-2 summarizes the modifications and statements to [\[RFC 4291\]](#) for Thread
 8 Networks. (See Section 2.5, Nomenclature for Tables Referencing Other Standards, in
 9 Chapter 2, Supporting Information, for an explanation of the "Status" column.)

10 **Table 5-2. Modifications and Statements to [\[RFC 4291\]](#)**

Chapter / Section	Title and Remarks / Modifications	Status
1	Introduction	I
2	IPv6 Addressing	N
2.1	Addressing Model	N

Chapter / Section	Title and Remarks / Modifications	Status
2.2	Text Representation of Addresses	N
2.3	Text Representation of Address Prefixes	N
2.4	Address Type Identification	N
2.5	Unicast Addresses	M
2.5.1	Interface Identifiers	N
2.5.2	The Unspecified Address	N
2.5.3	The Loopback Address	N
2.5.4	Global Unicast Addresses	N
2.5.5	IPv6 Addresses with Embedded IPv4 Addresses	N/R
2.5.6	Link-Local IPv6 Unicast Addresses	N
2.5.7	Site-Local IPv6 Unicast Addresses	N/R
2.6	Anycast Addresses	M
2.6.1	Required Anycast Address	N
2.7	Multicast Addresses	N
2.7.1	Pre-Defined Multicast Addresses	N
2.8	A Node's Required Addresses <ul style="list-style-type: none"> • A Node MAY recognize the loopback address. 	N,M
3	Security Considerations	I
4	IANA Considerations	I
5	Acknowledgements	I
6	References	I

Chapter / Section	Title and Remarks / Modifications	Status
6.1	Normative References	I
6.2	Informative References	I
Appendix A	Creating Modified EUI-64 Format Interface Identifiers	N/R
Appendix B	Changes from RFC 3513	I

1 **5.2 IPv6 Addressing Architecture**

2 **5.2.1 Address Scopes**

3 **5.2.1.1 Link-Local Scope**

4 In a Thread Network, the scope boundaries of link-local scope are defined by the set of
5 Thread interfaces that are reachable with a single radio transmission.

6 **5.2.1.2 Realm-Local Scope**

7 In a Thread network, the scope boundaries of Realm-Local scope [RFC 7346] are defined by
8 the set of Thread interfaces participating within the same Thread Network. All Thread
9 interfaces within the same Realm-Local scope zone share the same Thread commissioning
10 parameters (that is, Master Key, Extended Personal Area Network ID (PAN ID), and Mesh-
11 Local Prefix) and are actively participating in the same Thread Network Partition. Thread
12 devices form Mesh-Local Addresses out of the Mesh-Local Prefix.

13 **5.2.2 Unicast Addressing**

14 All Thread devices use a number of different IPv6 unicast addresses for communication. For
15 addresses that have scope larger than link-local, Thread defines three different types of
16 address:

- 17 • Routing Locator (RLOC)
- 18 • Anycast Locator (ALOC)
- 19 • Endpoint Identifier (EID)

20 Thread also uses three scopes for unicast addressing:

- 21 • Link-local
- 22 • Mesh-local
- 23 • Global

5.2.2.1 Routing Locator (RLOC)

A Thread Routing Locator (RLOC) is an IPv6 address that identifies the location of a Thread interface within a Thread Network Partition. RLOCs are used for communicating control traffic and delivering IPv6 datagrams to their intended destinations. In general, applications do not see or use RLOCs.

A RLOC is formed out of the Mesh-Local Prefix and encodes the Router ID and Child ID in the IID (Interface Identifier). The prefix is the Mesh-Local Prefix and the IID has the form 0000:00FF:FE00:xxxx where 'xxxx' is a 16-bit value that embeds the Router ID and Child ID. Bits 0 through 5 encode the Router ID. Bits 7 through 15 encode the Child ID. Bit 6 is reserved and MUST be set to '0' (zero) as illustrated in Figure 5-1:

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5
Router ID					R	Child ID										

Figure 5-1. Unicast Routing Locator Bits

Child ID 0 is reserved for the Thread router that is assigned the Router ID. Any Child of the Thread Router identified by Router ID can have a Child ID in the range of 1 through 511.

Because the RLOC embeds the Router ID and Child ID, the RLOC for a Thread interface may change whenever the Thread Network Partition topology changes (for example, a Child moves to a new Parent or Parent receives a new Router ID).

In this specification, the RLOC16 refers to the 16-bit encoding above that embeds the Router ID and Child ID. When an RLOC is assigned to a Thread interface, the MAC layer sets the IEEE 802.15.4 Short Address to the RLOC16.

Router ID 63 is reserved for Anycast Locators.

5.2.2.2 Anycast Locator (ALOC)

A Thread Anycast Locator (ALOC) is an IPv6 address that identifies the location of one or more Thread interfaces within a Thread Network Partition. ALOCs are used where the specific RLOC of a destination is not known at the originator. In general, applications do not see or use ALOCs.

An ALOC is formed out of the Mesh-Local Prefix and encodes the Anycast Locator destination in the IID. The prefix is the Mesh-Local Prefix and the IID has the form 0000:00FF:FE00:FCxx where 'FCxx' is a 16-bit value that indicates the ALOC destination.

In this specification, the ALOC16 refers to the 16-bit encoding of the Anycast Locator destination, where the ALOC16 is defined in Table 5-3. When forwarding messages to an ALOC destination, Routers use shared network information, obtained from Network Data, Leader Data, or Commissioning Data, to map the ALOC to an appropriate RLOC for a device on the Thread Network. When multiple devices have autoconfigured an ALOC, typically the nearest, that is, the one with the lowest path cost, is chosen.

Table 5-3 details the current allocation of the ALOC16 space.

1

Table 5-3. Allocation of the ALOC Space

ALOC16	Name	Description and Unicast RLOC lookup method
0xFC00	Leader	This address references the Leader of the partition. The Unicast RLOC corresponding to the Leader can be obtained from the Leader Data.
0xFC01 – 0xFC0F	DHCPv6 Agent	These addresses reference the nearest DHCPv6 Agent for the prefix associated with a context ID in the range 1 - 15. The ALOC16 is equal to 0xFC00 + context ID. The Unicast RLOC corresponding to the DHCPv6 Agent can be obtained from the Network Data.
0xFC10 – 0xFC2F	Service	These addresses reference the nearest server for a Service Type ID in the range 0 - 31. The ALOC16 is equal to 0xFC10 + Service Type ID. The Unicast RLOC corresponding to the server can be obtained from the Network Data.
0xFC30 – 0xFC37	Commissioner	This address references an active Commissioner using an index in the range 0 - 7. The ALOC16 is equal to 0xFC30 + index, where the index is equal to Commissioner Session ID mod 8. The Unicast RLOC corresponding to the Commissioner or Border Agent can be obtained from the Commissioner Dataset.
0xFC40 – 0xFC4E	Neighbor Discovery Agent	These addresses reference the nearest Neighbor Discovery Agent for the prefix associated with a context ID in the range 1 - 15. The ALOC is equal to 0xFC3F + context ID. The RLOC corresponding to the Neighbor Discovery Agent can be obtained from the Network Data.
0xFC38 – 0xFC3F, 0xFC4F – 0xFF	Reserved	Reserved for future use.

2 5.2.2.1 Leader ALOC

3 Thread defines a Leader ALOC to reach the Leader of the current partition. This Anycast Locator is mainly used to provide reachability to the Leader by the Commissioner. The Leader ALOC16 has the value 'FC00'.

6 5.2.2.2 DHCPv6 Agent ALOC

7 Thread defines a DHCPv6 (Dynamic Host Configuration Protocol version 6) Agent ALOC to reach DHCPv6 Agents serving a given valid prefix. The DHCPv6 Agent ALOC16 has the form 'FC0n', where 'n' is the context ID.

10 For more information, see Section 5.14, **Stable Network Data** for details on the use of the 11 DHCPv6 ALOC.

1 **5.2.2.2.3 Service ALOC**

2 Thread defines a Service ALOC to reach servers within a Thread Network. A Service ALOC
3 provides a convenient way for MTDs to reach a given type of server. The Service ALOC16
4 has the form 'FCmn', where mn' is the Service Type ID plus 16.

5 For more information, see Section 5.14, [Stable Network Data](#) for details on the use of the
6 Service ALOC.

7 **5.2.2.2.4 Commissioner ALOC**

8 Thread defines a Commissioner ALOC to reach the active Commissioner. The Commissioner
9 ALOC16 has the form '0xFC3N', where N is Commissioner Session ID for the corresponding
10 Commissioner modulo 8.

11 **5.2.2.2.5 Neighbor Discovery ALOC**

12 Thread defines a Neighbor Discovery ALOC to reach Neighbor Discovery Agents serving a
13 given valid prefix. The Neighbor Discovery Agent ALOC16 has the form 'FC4n', where 'n' is
14 the context ID minus 1.

15 For more information, see Section 5.15.8, [Host Behavior](#), for details on the use of the
16 Neighbor Discovery ALOC.

17 **5.2.2.3 Endpoint Identifier (EID)**

18 A Thread EID is an IPv6 address that uniquely identifies a Thread interface within a Thread
19 Network Partition and is independent from topology changes. All addresses, other than
20 RLOCs or ALOCs, that are configured using either the Mesh-Local Prefix or a prefix offered
21 by a Border Router for address configuration are considered EIDs. An EID is independent
22 from the position of the Thread interface within a Thread Network Partition and does not
23 change in response to Thread Network Partition topology changes.

24 An EID provides a stable identifier for the Thread interface within a Thread Network
25 Partition. However, an EID is not directly routable, because the Thread routing protocol only
26 exchanges route information for RLOCs. To deliver an IPv6 datagram with an EID as the
27 IPv6 Destination Address, a Thread device must perform an EID-to-RLOC mapping lookup.
28 This is handled by the network itself and applications do not need to be concerned about
29 this mapping.

30 Because EIDs remain stable in the face of topology changes, applications SHOULD use EIDs
31 when communicating with Thread interfaces. However, applications MAY use RLOCs directly
32 for simple request-response communication patterns within a Thread Network Partition.

33 **5.2.2.4 Link-Local Addresses**

34 Thread uses link-local addresses to reach Thread interfaces within range of a single radio
35 transmission. Thread devices use link-local addresses to discover neighbors, configure links,
36 and exchange routing information. A Thread device MUST assign a link-local IPv6 address
37 where the interface identifier is set to the MAC Extended Address with the universal/local bit
38 inverted. Note that the MAC Extended Address is not the factory-assigned IEEE EUI-64, as
39 described in Chapter 3.

40 **5.2.2.5 Mesh-Local Addresses**

41 Thread uses mesh-local addresses to reach Thread interfaces within the same Thread
42 Network Partition. When attached to a Thread Network Partition, a Thread interface
43 autoconfigures two Mesh-Local addresses as follows:

- 1 • Routing Locator (RLOC): an RLOC where the prefix is the Mesh-Local Prefix and the
2 interface identifier encodes the Router and Child IDs.
- 3 • Mesh-Local Endpoint Identifier (ML-EID): an EID where the prefix is the Mesh-Local
4 Prefix and the interface identifier (IID) is chosen at random. The IID MUST not have
5 the form 0000:00FF:FE00:xxxx as this is reserved for RLOCs and ALOCs. Note that
6 this IID is persistent over reboots (see Table 10-10). The ML-EID Interface Identifier
7 and the IEEE 802.15.4 Extended Address MUST be independent and one MUST NOT
8 be derived from the other to allow for duplicate address detection.

9 A Thread interface on the Leader of a Thread Network Partition additionally autoconfigures
10 the Leader ALOC.

11 **5.2.2.6 Global Addresses**

12 Thread interfaces MAY assign additional IPv6 unicast addresses for valid prefixes assigned
13 to the Thread Network. Such addresses can be used to enable communication with devices
14 outside the Thread Network Partition. The Thread Network Data indicates what valid
15 prefixes are assigned to the Thread Network and available for configuring IPv6 addresses.

16 **5.2.3 Multicast Addressing**

17 Thread devices subscribe to a number of mandatory well-known IPv6 multicast addresses.
18 Because rx-off-when-idle devices use IEEE 802.15.4 indirect transmissions to receive
19 messages reliably, Thread defines which IPv6 multicast addresses are to be forwarded to rx-
20 off-when-idle devices using IEEE 802.15.4 indirect transmissions.

21 **5.2.3.1 Link-Local Scope**

22 All Thread interfaces MUST subscribe to the Link-Local All Nodes multicast address
23 (FF02::1). All Thread interfaces operating in a Router, REED, or Border Router mode MUST
24 subscribe to the Link-Local All Routers multicast address (FF02::2).

25 By default, Parents MUST NOT use IEEE 802.15.4 indirect transmissions to forward
26 multicast packets destined to the Link-Local All Nodes (FF02::1) and Link-Local All Routers
27 (FF02::2) to their rx-off-when-idle Children.

28 To better support multicast messaging to rx-off-when-idle Children, all Thread interfaces
29 MUST subscribe to the Link-Local All Thread Nodes multicast address.

30 The Link-Local All Thread Nodes multicast address is a link-local Unicast Prefix-Based
31 Multicast Address [\[RFC 3306\]](#), with:

- 32 • flgs set to 3 (P = 1 and T = 1)
- 33 • scop set to 2
- 34 • plen set to the Mesh Local Prefix length
- 35 • network prefix set to the Mesh Local Prefix
- 36 • group ID set to 1

37 Parents MUST use IEEE 802.15.4 indirect transmissions to forward multicast packets
38 destined to the link-local all-Thread-nodes to their rx-off-when-idle Children.

39 Thread devices MAY subscribe to additional link-local multicast addresses. Rx-off-when-idle
40 Children MUST register such IPv6 multicast addresses with their Parent to receive them via
41 IEEE 802.15.4 indirect transmissions.

5.2.3.2 Realm-Local Scope

All Thread interfaces MUST subscribe to the realm-local all-nodes multicast address (FF03::1). All Thread interfaces operating in a Router, REED, or Border Router mode MUST subscribe to the realm-local all-routers multicast address (FF03::2). To forward multicast packets that have scope larger than realm-local, all Thread devices MUST receive and process MPL Data Messages sent to the realm-local ALL_MPL_FORWARDERS address (FF03::FC).

By default, Parents MUST NOT use IEEE 802.15.4 indirect transmissions to forward multicast packets destined to the realm-local all-nodes (FF03::1) and realm-local all-routers (FF03::2) to their rx-off-when-idle Children.

To better support multicast messaging to rx-off-when-idle Children, all Thread interfaces MUST subscribe to the Realm-Local All Thread Nodes multicast address.

The Realm-Local All Thread Nodes multicast address is a realm-local Unicast Prefix-Based Multicast Address [\[RFC 3306\]](#), with:

- flgs set to 3 (P = 1 and T = 1)
- scop set to 3
- plen set to the Mesh Local Prefix length
- network prefix set to the Mesh Local Prefix
- group ID set to 1

Parents MUST use IEEE 802.15.4 indirect transmissions to forward multicast packets destined to the realm-local all-Thread-nodes to their rx-off-when-idle Children.

Thread devices MAY subscribe to additional realm-local multicast addresses. Rx-off-when-idle Children MUST register such IPv6 multicast addresses with their Parent to receive them via IEEE 802.15.4 indirect transmissions.

5.2.3.3 Other Multicast Scopes

Thread devices MAY subscribe to additional multicast addresses of arbitrary scope. Rx-off-when-idle Children MUST register such IPv6 multicast addresses with their Parent to receive them via IEEE 802.15.4 indirect transmissions.

A Border Router SHOULD NOT forward multicast packets from other interfaces into a Thread Network. However, a Border Router MAY selectively forward multicast packets from other interfaces into a Thread Network when such multicasts are appropriate for a Thread Network. The Border Router SHOULD NOT forward multicast packets from higher bandwidth networks.

5.3 IPv6 Address Configuration

5.3.1 Link-Local Addresses

A Thread device MUST autoconfigure a link-local IPv6 address where the interface identifier is set to the MAC Extended Address with the universal/local bit inverted. Note that the MAC Extended Address is not the factory-assigned IEEE EUI-64, as described in Chapter 3.

1 5.3.2 Mesh-Local Addresses

2 When attached to a Thread Network Partition, a Thread interface MUST autoconfigure an
3 RLOC. When operating as a Child, the device generates the full IPv6 RLOC from the RLOC16
4 obtained during the Attach process. When operating as a Router, the device obtains the full
5 IPv6 RLOC from the RLOC16 assigned by the Leader.
6 A Thread interface MUST autoconfigure an ML-EID with an interface identifier chosen at
7 random. The ML-EID Interface Identifier MUST be independent from the MAC layer's IEEE
8 802.15.4 Extended Address and one MUST NOT be derived from the other. The ML-EID
9 Interface Identifier MUST NOT have the form as those used for RLOCs. If a randomly chosen
10 value has the form 0000:00ff:fe00:xxxx, the Thread device MUST choose another value at
11 random.

12 5.3.3 Global Addresses

13 The Thread Network Data MAY indicate one or more valid prefixes assigned to the network.
14 The values of P_slaac and P_dhcp in the Valid Prefix Set (see Section 5.13.2, **Valid Prefix
15 Set**) indicate whether addresses for a given prefix may be configured using Stateless
16 Address Autoconfiguration (SLAAC) or DHCPv6.

17 If P_slaac is 'true', then a Thread device MAY assign a global address out of the associated
18 P_prefix. Thread devices SHOULD implement and employ [\[RFC 7217\]](#) as the default scheme
19 for generating stable IPv6 addresses when SLAAC is allowed. If P_slaac is 'false', the Thread
20 device MUST NOT perform IPv6 address autoconfiguration.

21 If P_dhcp is 'true', then a Thread device MAY configure a global IPv6 address for the
22 associated P_prefix by using a P_border_router as a DHCPv6 server.

23 In addition to stateless- and DHCPv6-based autoconfiguration, a Thread device MUST
24 support the ability to manually configure an arbitrary IPv6 address for a valid prefix
25 included in the Network Data, in addition to IPv6 addresses that may be configured via
26 SLAAC or DHCPv6 as indicated by the Network Data. To support Manually Configured IPv6
27 Addresses, Parents MUST NOT restrict the IPv6 addresses registered by Children other than
28 (a) the IPv6 address MUST use an valid prefix; and (b) a Child MAY be limited to having no
29 more than four registered addresses (inclusive of the ML-EID). In particular, a Child MAY
30 register multiple addresses with the same valid prefix.

31 Source addresses SHOULD be chosen in accordance with the rules in Section 5 of
32 [\[RFC 6724\]](#), except in the case of conflicting application requirements. Note that, because
33 Thread uses a multihop subnet, "next-hop" in rule 5.5 MUST be interpreted as meaning the
34 Border Router through which a packet will be routed, which may not be the next hop within
35 the Thread subnet.

36 5.4 EID-to-RLOC Mapping

37 Because EIDs are not directly routable, a Full Thread Device (FTD) must perform EID-to-
38 RLOC mapping to forward a packet to a destination identified by an EID.

1 **5.4.1 Information Base**

2 **5.4.1.1 Local Address Set**

3 The Local Address Set records the set of EIDs assigned to a Thread interface.

4 **5.4.1.2 MTD Child Address Set**

5 The MTD Child Address Set records the set of EIDs assigned to each MTD Child's Thread
6 interface. The MTD Child Address Set consists of AddressSets, one per MTD Child as follows:

- 7 • AddressSet - a set of IPv6 addresses assigned to the MTD Child's Thread interface.
8 Each MTD Child Address Set MUST support at least four IPv6 addresses, which
9 includes the ML-EID and any other non-link-local IPv6 unicast addresses assigned to
10 the corresponding MTD Child's Thread interface.

11 **5.4.1.3 Address Query Set**

12 The Address Query Set records the state for each EID-to-RLOC mapping that is being
13 performed by an FTD. The Address Query Set consists of Address Query Tuples, one per
14 EID, as follows:

15 (EID, AQ_Timeout, AQ_Failures, AQ_Retry_Timeout)

16 where:

- 17 • EID is the IPv6 address for which an RLOC is desired.
- 18 • AQ_Timeout is the time remaining for waiting for responses to an Address Query, or
19 zero if there is no outstanding Address Query.
- 20 • AQ_Failures is the number of consecutive Address Query messages for which no
21 corresponding response was received before AQ_Timeout expires.
- 22 • AQ_Retry_Timeout is the time a device must wait before sending another Address
23 Query message.

24 **5.4.2 Address Query**

25 FTDs use Address Query to look up an EID-to-RLOC mapping. All Thread interfaces on FTDs
26 MUST subscribe to the Realm-Local All-Routers address (FF03::2) and process Address
27 Query messages. When an FTD needs to look up the RLOC for a given EID, it multicasts an
28 Address Query message to the Realm-Local All-Routers address (FF03::2). Any and all
29 recipients that have the EID configured, or have an MTD Child that has that EID registered,
30 respond with a unicast Address Notification message.

31 FTDs MUST perform EID-to-RLOC lookups for ML-EIDs and for EIDs formed using on-mesh
32 prefixes and MUST NOT perform EID-to-RLOC lookups on EIDs whose prefixes are not on-
33 mesh. Border Routers that offer a valid prefix MAY perform EID-to-RLOC lookups for EIDs
34 that use that prefix, whether or not the prefix is on-mesh. MTDs MUST NOT perform EID-to-
35 RLOC lookups.

36 **5.4.2.1 Transmission of Address Query Messages**

37 When an FTD determines that it needs to perform an EID-to-RLOC lookup, it first
38 determines whether or not the EID already exists in the Address Query Set.

- 1 If there is no entry that includes the EID, then a new entry is added with the following
2 values:
3 AQ_Timeout = ADDRESS_QUERY_TIMEOUT
4 AQ_Retry_Timeout = ADDRESS_QUERY_INITIAL_RETRY_DELAY
5 AQ_Failures = 0
6 If there is an entry that includes the EID, then the requestor determines its action based on
7 the values of AQ_Timeout and AQ_Retry_Timeout as follows:
8 • If AQ_Timeout is non-zero, then the packet is delayed pending the completion of the
9 address discovery that is already underway. No other action is taken.
10 • If AQ_Timeout and AQ_Retry_Timeout are both zero, a new address query is
11 initiated and the packet is delayed pending the completion of the address query.
12 • If AQ_Timeout is zero and AQ_Retry_Timeout is nonzero, then a prior address query
13 for the EID failed to elicit a response and no RLOC for the EID is known. Any packet
14 whose destination is the EID MUST be dropped. A new discovery will be initiated
15 when a packet is sent to this EID after AQ_Retry_Timeout has reached zero.

5.4.2.2 ADDR_QRY.qry – Address Query

Address Query messages are unreliable queries containing the EID for which an RLOC is desired.

CoAP Request URI

coap://[FF03::2]:MM/a/aq

Transaction Pattern

Ntf/Qry/Ans_NON

CoAP Payload

Target EID TLV

The IPv6 Source address MUST be the RLOC of the originator. The IPv6 Destination address MUST be the Realm-Local All-Routers multicast address (FF03::2). The Target EID TLV MUST contain the EID for which an RLOC is desired.

Upon transmitting an Address Query message, the requestor MUST set the AQ_Timeout in the corresponding Address Query Tuple to AQ_TIMEOUT.

A device that is sending an Address Query message MAY send an initial multicast with an IPv6 Hop Limit of 1 (for Routers) or 2 (for REEDs). This multicast has much lower overhead than a full Realm-Local multicast and is sufficient if the target device or its Parent is within one hop of the sender (or its Parent). If this initial multicast fails to elicit a response, a full Realm-Local multicast is sent.

5.4.2.3 Receipt of Address Query Messages

All Thread interfaces on FTDs MUST subscribe to the Realm-Local All-Routers address (FF03::2) and process Address Query messages. Upon receipt of a valid Address Query message, an FTD determines whether or not the IPv6 address contained within the Target EID TLV exists within the Local Address Set.

If the IPv6 address contained within the Address Query message exists within the Local Address Set or within the MTD Child Address Set, the FTD sends a message containing the

1 RLOC for itself, specifically an Address Notification message. An FTD MUST NOT send CoAP
2 responses to multicast ADDR_QRY.qry CoAP requests.

3 **5.4.2.4 ADDR_NTF.ans - Address Notification**

4 Address Notifications are answer messages sent in response to Address Query messages
5 and are formatted as follows:

6 CoAP Request URI

7 `coap://[<Address Query Source>]:MM/a/an`

8 Transaction Pattern

9 Ntf/Qry/Ans_CON

10 CoAP Payload

11 Target EID TLV

12 RLOC16 TLV

13 ML-EID TLV

14 [Time Since Last Transaction TLV]

15 The IPv6 Source address MUST be the RLOC of the originator. The IPv6 Destination address
16 MUST be the RLOC of the destination.

17 The responder copies the Target EID TLV from the Address Query message into the Address
18 Notification message. The RLOC16 TLV contains the RLOC16 of the FTD responding to the
19 Address Query. The ML-EID TLV contains the ML-EID associated with the Target EID and is
20 included to support duplicate detection of the Target EID. The Time Since Last Transaction
21 TLV is only present when the message is sent by an FTD Parent on behalf of a Child. FTDs
22 responding for themselves MUST NOT include the Time Since Last Transaction TLV. The
23 Time Since Last Transaction TLV contains the number of seconds since a keep-alive
24 message (as defined in Section 4.7.5, Timing Out Children, in Chapter 4, Mesh Link
25 Establishment) was last received from the Child.

26 **5.4.2.5 Receipt of Address Notification Messages**

27 FTDs respond to Address Query messages with Address Notification messages. If one or
28 more Address Notification messages are received for the same EID before AQ_Timeout for
29 the EID expires, the Address Query completed successfully. If multiple Address Notification
30 messages are received with the same ML-EID, the message with the lowest Time Since Last
31 Transaction is used. If multiple Address Notification messages are received with different
32 ML-EIDs, the FTD detects that the Target EID is in use by more than one Thread interface
33 and acts to resolve the duplicate, as described in Section 5.6, **Duplicate IPv6 Address
34 Detection**.

35 If AQ_Timeout for the EID expires and no successful Address Query response was received,
36 the device increments AQ_Failures and AQ_Retry_Timeout is set to:

37 $\text{ADDRESS_QUERY_INITIAL_RETRY_DELAY} * (2 ^ (\text{AQ_Failures} - 1))$

38 with a maximum value of ADDRESS_QUERY_MAX_RETRY_DELAY.

39 **5.4.3 Proactive Address Notifications**

40 In addition to responding to Address Queries, Thread devices MAY also proactively send
41 Address Notification messages to notify other devices of their EID and RLOC. For example, a

- 1 Child that attaches to a new Parent, and thus obtains a new RLOC, MAY notify Border
2 Routers of the change and thus avoid the overhead of multicast Address Queries.

3 **5.4.3.1 ADDR_NTF.ntf - Proactive Address Notification**

4 Proactive Address Notifications are identical to ADDR_NTF.ans sent in answer to Address
5 Query messages, except that they are sent proactively instead of in answer to a query.

6 CoAP Request URI

7 `coap://[<peer address>]:MM/a/an`

8 Transaction Pattern

9 Ntf/Qry/Ans_CON

10 CoAP Payload

11 Target EID TLV

12 RLOC16 TLV

13 ML-EID TLV

14 The IPv6 Source address MUST be the RLOC of the originator. The IPv6 Destination address
15 MUST be the RLOC of the destination.

16 The Target EID TLV contains an EID configured by the sender. The RLOC16 TLV contains the
17 sender's RLOC16 if the sender is an FTD, or the sender's Parent's RLOC16 if the sender is an
18 MTD. The ML-EID TLV contains the ML-EID in use by the sender and is included to support
19 duplicate detection of the Target EID.

20 **5.4.3.2 Receipt of Proactive Address Notifications**

21 Any Address Notification message received for a Target EID that is not in the Address Query
22 Set MUST be treated as a Proactive Address Notification. Upon receipt of a Proactive
23 Address Notification message, the recipient determines whether or not there is an entry for
24 the Target EID in its EID-to-RLOC Set. If there is such an entry, the recipient MUST replace
25 the corresponding RLOC with the RLOC derived from the value in the RLOC16 TLV. If there
26 is no such entry and sufficient resources are available, the recipient MAY add a new entry to
27 its EID-to-RLOC Set containing that same EID and RLOC.

28 **5.5 EID-to-RLOC Map Cache**

29 To avoid repeatedly sending Address Query messages when forwarding datagrams, every
30 FTD maintains an EID-to-RLOC Map Cache. Whenever an FTD needs to perform an EID-to-
31 RLOC lookup, it first checks the Map Cache to see if an EID-to-RLOC mapping already
32 exists. If so, the FTD simply uses the corresponding RLOC. If not, the FTD MUST send an
33 Address Query as specified in Section 5.4.2, [Address Query](#).

34 FTDs MUST maintain an EID-to-RLOC Map Cache for ML-EIDs and for EIDs formed using on-
35 mesh prefixes and MUST NOT include EIDs whose prefixes are not on-mesh. Border Routers
36 that offer a valid prefix MAY include EID-to-RLOC EIDs that use that prefix in their cache,
37 whether or not the prefix is on-mesh. MTDs do not maintain EID-to-RLOC Map Caches.

1 **5.5.1 Information Base**

2 **5.5.1.1 EID-to-RLOC Set**

3 The EID-to-RLOC Set records the mappings from EIDs to RLOCs. The EID-to-RLOC Set
4 consists of EID-to-RLOC Tuples as follows:

5 (EID, RLOC, Age)

6 where:

- 7 • EID is the IPv6 address for which the RLOC is desired.
8 • RLOC is the RLOC for the corresponding EID.
9 • Age indicates how recently the EID-to-RLOC entry was used relative to other entries
10 in the same EID-to-RLOC Set.

11 A Thread interface maintains one EID-to-RLOC Set for itself and one EID-to-RLOC Set for
12 each of the MTD Children attached to it. Each EID-to-RLOC Set MUST support at least four
13 non-link-local unicast IPv6 addresses.

14 **5.5.2 Managing EID-to-RLOC Map Cache Entries**

15 Upon a successful completion of an Address Query, an FTD MUST update the appropriate
16 EID-to-RLOC Set. If the datagram that triggered the Address Query was generated locally,
17 then the FTD updates the EID-to-RLOC Set corresponding to its Thread interface. If the
18 datagram that triggered the Address Query was generated by an MTD Child attached to it,
19 then the Parent Thread device updates the EID-to-RLOC Set corresponding to the MTD
20 Child.

21 If no entry exists for the corresponding EID, the FTD MUST create a new entry. If no entries
22 are available, the FTD MUST evict the least-recently-used entry in the corresponding EID-
23 to-RLOC Set.

24 **5.5.2.1 Cache Errors**

25 The RLOC assigned to a device can change over time, causing existing address cache
26 entries to become incorrect. These errors need to be detected and corrected.

27 A cache error is detected if a device receives a packet containing a 6LoWPAN mesh header
28 and an EID destination IPv6 address where the mesh destination is the device's RLOC16
29 address but the EID does not belong to the receiver or, if the receiver is a router, the EID
30 has not been registered by any Child of the receiver.

31 When a cache error is detected, the receiving device must respond by sending an ICMPv6
32 Destination Unreachable message with the ICMPv6 Code set to 0 (No route to destination)
33 to the packet's mesh source. The ICMPv6 message's source is the device's RLOC address
34 and its destination is the RLOC address derived from the RLOC16 mesh source of the
35 original message. The ICMPv6 message payload MUST contain at least the full IPv6 header
36 of the original message.

37 When an ICMPv6 Destination Unreachable message is received, the recipient must extract
38 the EID address from the original IPv6 header contained in the ICMPv6 message payload
39 and locate the EID-to-RLOC tuple with the matching EID. If there is such an EID-to-RLOC
40 tuple, and its RLOC value matches the RLOC IP source of the ICMP message, that EID-to-
41 RLOC tuple must be removed from the cache.

1 Also, when an updated Router ID Set is received, all EID-to-RLOC tuples whose RLOC values
2 contain an unassigned Router ID must be removed from the cache.

3 **5.5.2.2 Optimizations**

4 FTDs MAY update EID-to-RLOC Map Cache entries by inspecting packets being sent/received
5 via a Thread interface. An FTD MAY add/update EID-to-RLOC Set entries by associating
6 RLOCs and EIDs contained within the same packet. An FTD MAY also remove EID-to-RLOC
7 Set entries by observing ICMPv6 error messages.

8 Whenever the RLOC for a Thread device changes, the Thread device SHOULD proactively
9 notify other FTDs that likely maintain an EID-to-RLOC mapping for it. This can be done with
10 any IPv6 packet that includes the EID as the IPv6 Source and the RLOC as the Mesh Source.

11 **5.6 Duplicate IPv6 Address Detection**

12 Thread implements a form of Optimistic Duplicate Address Detection, where IPv6 addresses
13 may be assigned and used before any attempt to detect and resolve duplicates takes place.

14 Thread uses Address Notification messages to detect duplicate EIDs. Thread devices receive
15 Address Notification messages in response to Address Queries or asynchronously in the case
16 of Proactive Address Notifications. Thread devices perform duplicate detection in both
17 scenarios.

18 Thread's duplicate detection relies on devices having unique ML-EIDs to identify whether an
19 EID is in use by more than one device. Thread devices generate ML-EIDs at random and no
20 attempt is made to guarantee uniqueness of ML-EIDs. However, the probability can be
21 easily estimated, and the probability of collision is exceedingly small [\[RFC 4429\]](#).

22 **5.6.1 Address Queries**

23 When the same EID is assigned to two or more Thread interfaces in the same Thread
24 Network Partition, an Address Query generates multiple Address Notification messages. The
25 receiver of the Address Notification messages compares the ML-EID TLVs of the received
26 Address Notification messages to determine whether or not the multiple replies are due to
27 multiple interfaces using the same IPv6 address or a single interface migrating from Router
28 to Router.

29 If two or more received Address Notification messages have the same EID but different ML-
30 EIDs, the receiving device determines that the EID is in use by more than one device and
31 MUST multicast an Address Error Notification message to the Realm-Local All-Routers
32 multicast address (FF03::2).

33 **5.6.2 Proactive Address Notifications**

34 An FTD that receives a Proactive Address Notification message compares the RLOC16 of the
35 Address Notification message to the RLOC16 contained in its EID-to-RLOC Map Cache. If an
36 EID-to-RLOC Map Cache entry exists for the EID and the RLOC16 differs, the FTD
37 determines that the EID may be in use by more than one device and MUST unicast an
38 Address Error Notification to the RLOC16 contained in the EID-to-RLOC Map Cache entry
39 and then update the EID-to-RLOC Map Cache entry with the newly received RLOC16. This
40 unicast message serves to check whether the old RLOC is truly a duplicate user of the EID
41 or simply stale data.

1 The receiver of a unicast Address Error Notification message determines whether the Target
2 EID belongs to itself or, in the case of a Router, one of its Children and compares the ML-
3 EID associated with the Target EID in its local state and the ML-EID in the Address Error
4 Notification message. If the ML-EIDs differ, the FTD determines that the Target EID is in use
5 by more than one device and MUST multicast an Address Error Notification message to the
6 Realm-Local All-Routers multicast group (FF03::2).

7 **5.6.3 Creation of Address Error Notifications**

8 **5.6.3.1 ADDR_ERR.ntf - Address Error Notification**

9 An FTD sends Address Error Notification messages to notify other Thread interfaces that a
10 duplicate EID may be in use. Address Error Notification messages are CoAP POST requests
11 containing the detected duplicate IPv6 address and the ML-EID contained in the Address
12 Notification that triggered the duplicate address detection.

13 CoAP Request URI

14 `coap://[<peer address>]:MM/a/ae`

15 Transaction Pattern

16 Ntf/Qry/Ans_CON (when unicast) or Ntf/Qry/Ans_NON (when multicast)

17 CoAP Payload

18 Target EID TLV

19 ML-EID TLV

20 The IPv6 Source address MUST be the RLOC of the originator. The IPv6 Destination address
21 is either the Realm-Local All-Routers multicast address (FF03::2) when sent via multicast or
22 the RLOC of the destination when sent via unicast. A Thread device MUST NOT send CoAP
23 responses to multicast ADDR_ERR.ntf CoAP requests.

24 The Target EID TLV and the ML-EID TLV are copied from the Address Notification or Address
25 Error Notification message that triggered this message.

26 **5.6.4 Receipt of Address Error Notifications**

27 Upon receiving an Address Error Notification message, the Thread device checks whether
28 the Target EID is assigned to its Thread interface. If the Target EID is assigned to its Thread
29 interface and the ML-EID differs, then a duplicate is detected and the device MUST stop
30 using the Target EID.

31 If the recipient of an Address Error Notification is an FTD, the FTD checks if the Target EID
32 is assigned to one of its MTD' Children's Thread interfaces. If the Target EID is assigned to
33 an MTD Child's Thread interface and the ML-EID differs, then a duplicate is detected and the
34 device MUST unicast an Address Error Notification to the corresponding MTD Child and the
35 Router MUST remove the Target EID from the MTD Child Address Set.

36 If the recipient of a unicast Address Error Notification is an FTD, it MUST multicast an
37 Address Error Notification to the Realm-Local All-Routers multicast address (FF03::2).

5.7 DHCPv6 Services

Thread Networks do not depend on the use of DHCPv6 and Thread devices need not implement DHCPv6. Border Routers MAY act as DHCPv6 Agents and assign addresses and/or provide information about available DNS and other servers. Thread devices that wish to take advantage of any available DHCPv6 Agents need to be able to act as DHCPv6 Clients.

5.8 ICMPv6

A Thread-compliant device MUST implement [\[RFC 4443\]](#). Table 5-4 summarizes the modifications and statements to [\[RFC 4443\]](#) for Thread-compliant devices. (See Section 2.5, Nomenclature for Tables Referencing Other Standards, in Chapter 2, Supporting Information, for an explanation of the "Status" column.)

Table 5-4. Modifications and Statements to [\[RFC 4443\]](#)

Chapter / Section	Title and Remarks / Modifications	Status
1.	Introduction	I
2.	ICMPv6 (ICMP for IPv6)	N
2.1	Message General Format	N
2.2	Message Source Address Determination	N
2.3	Message Checksum Calculation	N
2.4	Message Processing Rules	N
3.	ICMPv6 Error Messages	N
3.1	Destination Unreachable Message	N
3.2	Packet Too Big Message	N
3.3	Time Exceeded Message <ul style="list-style-type: none">Fragment reassembly time exceeded message is only required on the Border Router because the other device types do not support IPv6 reassembly.	M
3.4	Parameter Problem Message	N

Chapter / Section	Title and Remarks / Modifications	Status
4	ICMPv6 Informational Messages	N
4.1	Echo Request Message	N
4.2	Echo Reply Message	N
5	Security Considerations	I
5.1	Authentication and Confidentiality of ICMP Messages	I
5.2	ICMP Attacks	I
6	IANA Considerations	I
6.1	Procedure for New ICMPV6 Type and Code Value Assignments	I
6.2	Assignments for This Document	I
7	References	I
7.1	Normative References	I
7.2	Informative References	I
8	Acknowledgements	I
Appendix A	Changes since RFC 2463	I

5.9 Routing Protocol

This section describes the routing protocol used in Thread Networks. The Thread routing protocol is a simple distance vector routing protocol. The main goal of the protocol is to maximize the amount of routing information that can be communicated in a single message. For this reason, the protocol is limited to networks containing MAX_ROUTERS or fewer Routers.

Note: More than MAX_ROUTERS routing addresses MAY be used to allow time for timing out of Router addresses before they are reused.

Routers periodically advertise their routing costs to all other Routers in the Thread Network partition and the quality of their one-hop links to neighbors. The routing cost to any particular destination is the minimum of what all neighbors advertise to that destination plus

1 your cost to that neighbor. The rate at which advertisements are sent is determined by an
2 instance of the Trickle algorithm [[RFC 6206](#)].

3 **5.9.1 Routing Database**

4 A Router maintains a routing database that records information for each separate interface
5 that uses Thread distance-vector routing. These databases contain the data sets described
6 in the following sub-sections.

7 **5.9.1.1 Router ID Set**

8 An interface's Router ID Set contains two values:

- 9 • ID_set is the current set of valid Router IDs.
- 10 • ID_sequence_number is the sequence number assigned to the current ID_set.

11 **5.9.1.2 Link Set**

12 An interface's Link Set records information about other Routers that are, or recently were,
13 neighbors on that interface. A Router maintains a tuple for each link to a neighbor as
14 follows:

15 ($L_{\text{router_id}}$, $L_{\text{link_margin}}$, $L_{\text{incoming_quality}}$, $L_{\text{outgoing_quality}}$, L_{age})

16 where:

- 17 • $L_{\text{router_id}}$ is the Router ID assigned to that neighbor.
- 18 • $L_{\text{link_margin}}$ is the measured link margin for messages received from the neighbor.
- 19 • $L_{\text{incoming_quality}}$ is the incoming link quality metric as calculated from
 $L_{\text{link_margin}}$.
- 20 • $L_{\text{outgoing_quality}}$ is the incoming link quality metric reported by the neighbor for
messages arriving from this Router.
- 21 • L_{age} is the elapsed time since an advertisement was received from the neighbor.

24 **5.9.2 Route Set**

25 An interface's Route Set records route cost and next hop information about other Routers
26 that are, or recently were, reachable (had a finite routing cost) from this one. A Router
27 maintains a tuple for each reachable Router as follows:

28 ($R_{\text{destination}}$, $R_{\text{next_hop}}$, $R_{\text{route_cost}}$)

29 where:

- 30 • $R_{\text{destination}}$ is the Router ID assigned to the destination of this route.
- 31 • $R_{\text{next_hop}}$ is the Router ID assigned to the next hop along the route.
- 32 • $R_{\text{route_cost}}$ is the routing cost reported by $R_{\text{next_hop}}$ for $R_{\text{destination}}$.

33 **5.9.3 Leader Database**

34 Routers acting as Leaders maintain an additional database for tracking Router ID
35 assignments. Similar to the Routing Database, a Leader maintains a separate Leader
36 Database for each interface on which a device is acting as a Leader.

5.9.3.1 ID Assignment Set

An interface's Assignment Set records the state of each Router ID.

A Leader maintains a tuple for each Router ID as follows:

(ID_id, ID_owner, ID_reuse_time)

where:

- ID_id is a Router ID.
- ID_owner is the IEEE 802.15.4 Extended Address of the Router to which this Router ID is currently assigned, or, if not assigned, the IEEE 802.15.4 Extended Address of the Router to which it was most recently assigned.
- ID_reuse_time is the time at which this Router ID MAY be reassigned, if it is not currently assigned.

The Leader records whether or not a Router ID is currently assigned in the Router ID Set.

5.9.4 Link Margins and Link Metrics

The L_link_margin values are running averages of the received relative signal strength from neighboring Routers. The relative signal strength is recorded as dB above the local noise floor, called the **link margin**. The average link margin is converted to a one-way link quality per Table 5-5.

Table 5-5. Average Link Margin Conversion to One-Way Link Quality

Link Margin	Quality	Link Cost
> 20dB	3	1
> 10dB	2	2
> 2dB	1	4
≤ 2dB	0	infinite

The one-way link quality is shared with neighbors using Route64 TLVs in MLE advertisements. The link cost is determined by the minimum of the two one-way link qualities for a link. For example, a link having a link quality of 2 in one direction and 1 in the other direction has a minimum link quality of 1, and therefore a link cost of 4.

Having only four possible link quality values minimizes the overhead of communicating link qualities to neighbors.

Links with less than 10dB of margin are unlikely to be stable, and so have a high routing cost. The intent is to use them only if there is no alternate path.

The link margin may be determined directly by measuring both incoming signal strength and noise floor. Alternatively, it may be derived from one or more other link quality measurements for which a correlation with the link margin has been determined. Which measurements are used and what the correlation is depends on the hardware platform.

Regardless of how the link margin is determined, averaging MUST be used to smooth out short-term volatility. Thread devices MUST maintain an exponentially weighted moving average of the link margin for each neighbor, which is updated (sampled) every time a new MAC frame is received from a neighbor; with a weighting of 1/8 or 1/16 for the new sample value. This weight choice is implementation-specific. For example, if the average link margin

1 for a neighbor is A and a message arrives from that neighbor with link margin X, the new
2 average becomes $(A * 7/8) + (X * 1/8)$. A hysteresis MUST also be applied when
3 determining the quality of a link, so that a link margin near the boundary of two qualities
4 does not cause frequent changes to L_incoming_quality. The value of the hysteresis is 2dB
5 for the thresholds between link qualities 1-2 and link qualities 2-3. The value of the
6 hysteresis is 1dB for the threshold between link qualities 0-1. For example, the average link
7 margin must increase to at least 12dB to change an existing quality 1 link to a quality 2 link
8 and to be at least 3dB to change an existing quality 0 link to a quality 1 link.

9 If a Router receives 802.15.4 status NO_ACK for FAILED_ROUTER_TRANSMISSIONS
10 consecutive unicast MCSP.DATA-Requests to a particular Router neighbor, it MUST set the
11 link quality to that Router neighbor to 0. As described in Section 5.9.7, **Sending**
12 **Advertisements**, the Router will then reset its Trickle timer. Note that failures to transmit
13 due to failed clear channel assessments do not count as a failure to receive an ACK.

14 **5.9.5 Routing Cost and Next Hop**

15 The routing cost for Router ID (R) is the minimum of the cost to send directly to R and the
16 minimum multihop route cost.

17 More specifically, if there is an entry in the Link Set with L_router_id = R, Cost_direct is the
18 link cost associated with the minimum of L_incoming_quality and L_outgoing_quality. If
19 there is no such entry, Cost_direct is infinite.

20 If there is an entry in the Route Set with R_destination = R, and R_route_cost plus the link
21 cost to R_next_hop is less than or equal to MAX_ROUTE_COST, then Cost_multihop is that
22 sum. If there is no such entry, or if the sum is greater than MAX_ROUTE_COST, then
23 Cost_multihop is infinite.

24 The routing cost to R is the minimum of Cost_direct and Cost_multihop. If both Cost_direct
25 and Cost_multihop are infinite, then R is unreachable. If Cost_direct is finite and less than
26 or equal to cost_multihop, then the next hop on the route is R itself. Otherwise, the next
27 hop is R_next_hop.

28 **5.9.6 Loops and Loop Detection**

29 A loop in the routing graph to certain destinations causes the cost to that destination to
30 count above MAX_ROUTE_COST.

31 There is no 'split horizon' loop avoidance. If a Router receives a packet from a neighbor
32 when that neighbor is the next hop for the packet's destination, the Route Set entry for the
33 destination MUST be removed from the Route Set. This detects and removes two-hop loops.

34 **5.9.7 Sending Advertisements**

35 Routers send MLE Advertisements as described in this section. REEDs send periodic MLE
36 Advertisements as described in Section, 5.16.6, **Children**. All MLE Advertisements MUST be
37 sent to the Link-Local All Nodes multicast address (FF02::1).

38 The rate at which advertisements are sent is determined by using the Trickle algorithm
39 [[RFC 6206](#)] using ADVERTISEMENT_I_MIN and ADVERTISEMENT_I_MAX as parameters
40 I_MIN and I_MAX. The Trickle parameter k is set to infinity. There is no consistency
41 checking for advertisements.

- 1 The Trickle timer is reset when the reported routing cost to one or more destinations
- 2 changes to or from infinite (represented as value 0). The Trickle timer SHOULD also be
- 3 reset if removing a Router ID for which there is a non-zero cost.
- 4 When sending an advertisement, a Router includes the L_incoming_quality,
- 5 L_outgoing_quality, and the routing cost for each Router ID in ID_SET. If the routing cost to
- 6 a Router ID is less than MAX_ROUTE_COST, then the Route value is that cost. Otherwise,
- 7 the Route value is infinite (represented as value 0).

8 **5.9.8 Processing Route64 TLVs**

- 9 The first step in processing an incoming Route64 TLV is to compare the Router ID sequence
- 10 number from the Route64 TLV with ID_sequence_number. If the Route64 TLV's sequence
- 11 number is higher, then the ID_sequence_number and ID_set values MUST BE replaced with
- 12 the sequence number and ID set from the Route64 TLV. ID sequence X is more recent than
- 13 ID sequence Y if: $((x-y) \bmod 256) \leq 127$
- 14 If the Route64 TLV's sequence number is the same or lower than ID_sequence_number,
- 15 then ID_sequence_number and ID_set MUST remain unchanged.
- 16 If the receiving device has a Router ID and the advertisement includes an incoming link
- 17 quality for that ID, then the L_outgoing_quality for the sender MUST be set to that incoming
- 18 link quality.
- 19 Finally, for each Router ID (R) other than the sender's ID or receiver's ID that appears in
- 20 both ID_set and the Route64 TLV's Assigned Router ID Mask, let Cost_reported be the route
- 21 cost for R in the Route64 TLV.

22 Then:

- 23 • If Cost_reported = 0 and R_next_hop == TLV sender's ID, then remove R from the
- 24 Route Set.
- 25 • Otherwise, if $0 < Cost_{reported}$ and
 - 26 ■ there is no entry for Router ID R in the Route Set
 - 27 OR
 - 28 ■ $R_{next_hop} ==$ TLV sender's ID
 - 29 OR
 - 30 ■ sender's link cost + Cost_reported < R_next_hop's link cost + R_route_cost,
 - 31 Then add or update an entry for R by setting R_next_hop to the sender's ID and
 - 32 R_route_cost to Cost_reported.

33 A REED MUST process Route64 TLVs to obtain the number of Routers on the Thread

34 Network, for use in determining whether or not to become an active Router, as described in

35 Section 5.9.9, **Router ID Management**. It MUST also obtain its Parent's cost to the Leader

36 from the Parent's Route64 TLV. If this cost is infinite, the Child sets a timer for

37 PARENT_ROUTE_TO_LEADER_TIMEOUT seconds. If it has not received a new Route64 TLV

38 from the Parent when the timer expires, it attempts to find and attach to a new Parent with

39 a finite cost to the Leader.

40 **5.9.9 Router ID Management**

- 41 A Leader is responsible for managing Router IDs. Like all Routers, the Leader maintains an
- 42 ID_Set showing the assigned Router IDs. Unlike other Routers, the Leader also maintains an

1 ID_Assignment_Set containing the IEEE 802.15.4 Extended Address of each device that has
2 an assigned ID and, for currently unassigned IDs, the time at which the ID may be
3 reassigned.

4 Whenever the Leader allocates or deallocates a Router ID, it changes its ID_set, increments
5 its ID_sequence_number, and updates the Router ID's entry in the ID_Assignment_Set. To
6 reclaim unused IDs, the Leader also increments ID_sequence_number whenever that value
7 has remained unchanged for ID_SEQUENCE_PERIOD seconds. If a Router goes for
8 NETWORK_ID_TIMEOUT seconds without receiving a new ID_sequence_number from a
9 neighbor, it MUST consider itself disconnected from the Leader and stop using its current
10 Router ID.

11 A Leader MAY deallocate a Router ID at any time. A Leader SHOULD deallocate a Router ID
12 if the Router ID has had an infinite routing cost for INFINITE_COST_TIMEOUT. When the
13 Leader deallocates a Router ID, it removes the ID from its ID_Set, increments
14 ID_sequence_number, and sets the Router ID's Reuse_Time to the current time plus
15 ID_REUSE_DELAY. The Leader SHOULD NOT reassign the Router ID to a new Router for at
16 least ID_REUSE_DELAY seconds.

17 If the number of active Routers on the Thread Network partition, as reported in the Route64
18 TLV of neighboring nodes, is less than ROUTER_UPGRADE_THRESHOLD, a REED MUST wait
19 a random period between 0 and ROUTER_SELECTION_JITTER from the time it detected the
20 condition and then, if the condition still holds, MUST attempt to become an active Router by
21 requesting a Router ID from the Leader. This helps prevent many Children from requesting
22 Router IDs at the same time.

23 If the number of active Routers on the network exceeds
24 ROUTER_DOWNGRADE_THRESHOLD, an active Router that meets all of the following criteria
25 MUST attempt to release its Router ID and become a REED. Let M be the set of neighboring
26 Routers for which the Router has a two-way link quality of 2 or better.

27

- Have at least MIN_DOWNGRADE_NEIGHBORS neighbors in set M.
- Have at least one neighbor that has as good or better quality links to all Routers in
M.
- Have fewer Children than three times the number of excess Routers. In other words,
if there are N active Routers and the Router has C Children,
then $C < 3 * (N - \text{ROUTER_DOWNGRADE_THRESHOLD})$.
- Not be the Leader.
- Not be the sole Border Router for a particular Provisioning Domain (see Section
5.13.7, **Provisioning Domain**). If multiple Border Routers for the same Provisioning
Domain are eligible to downgrade they MAY use some external mechanism to select
which do so.

38 To determine if the second criterion is satisfied, an active Router inspects the Route64 TLVs
39 from the MLE advertisements of neighboring Routers. Suppose the device D receives an MLE
40 advertisement from neighbor R. By examining the incoming and outgoing cost fields of the
41 Route64 TLV, D can determine R's neighborhood, that is, the set of Routers S to which R
42 has two-way links. If S contains every Router in D's for which D has a quality 2 or better
43 link with a quality at least as good as D's, then the second criterion has been satisfied.

44 If the decision to become a Child was triggered by an increase in the number of Routers,
45 the device MUST first delay a random period between 0 and ROUTER_SELECTION_JITTER
46 seconds from the time it detected the increase, then recheck the total number of Routers.
47 This helps prevent many Routers from becoming Children at the same time.

1 **5.9.10 Router ID Assignment**

- 2 A REED that wants to become a Router MUST send a CoAP POST Address Solicit message to
3 request a Router ID from the Leader. The Leader's Router ID is obtained from the shared
4 Thread Network Data.

5 **5.9.10.1 ADDR_SOL.req – Address Solicit Request**

- 6 The Address Solicit Request message is as follows:

- 7 CoAP Request URI

8 `coap://[<leader address>]:MM/a/as`

1 Transaction Pattern
2 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)
3 CoAP Payload
4 MAC Extended Address TLV
5 [RLOC16 TLV]
6 Status TLV
7 The MAC Extended Address TLV is included so the Leader can keep track of the mapping
8 between Thread devices and RLOC16s. The Status TLV indicates the reason the request was
9 sent. It contains TOO_FEW_ROUTEERS if the reason was that the number of active Routers is
10 less than ROUTER_UPGRADE_THRESHOLD. It contains HAVE_CHILD_ID_REQUEST if the
11 reason is that the sender has received an MLE Child ID Request message (see Section 4.7.1,
12 Attaching to a Parent, in Chapter 4, Mesh Link Establishment). It contains
13 PARENT_PARTITION_CHANGE if a Router having children has moved to a different partition
14 (see Section 5.16.4, **Resetting Thread Network Partition Data**). The RLOC16 TLV is
15 included only if the sender was previously a Router, possibly within another Partition, and
16 wishes to retain its original Router ID. The Leader MUST NOT allocate an RLOC16 to more
17 than one device as identified by the ML-EID.
18 If the status is HAVE_CHILD_ID_REQUEST or PARENT_PARTITION_CHANGE, the Leader
19 assigns a Router ID to the REED if one is available. If the status is TOO_FEW_ROUTEERS, the
20 Leader assigns a Router ID if the number of active Routers is less than
21 ROUTER_UPGRADE_THRESHOLD. If the REED requested a particular Router ID and that
22 Router ID is available, that Router ID is chosen. If no Router ID is available for assignment,
23 the Leader sends an Address Solicit Response containing the status 'No Address Available'.
24 Otherwise the Leader sends this response containing the status 'Success'.
25 **5.9.10.2 ADDR_SOL.rsp – Address Solicit Response**
26 The Address Solicit Response message is formatted as follows:
27 CoAP Response Code
28 2.04 Changed
29 Transaction Pattern
30 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)
31 CoAP Payload
32 Status TLV(Success, No Address Available)
33 [RLOC16 TLV]
34 [Router Mask TLV]
35 The RLOC16 TLV contains the assigned RLOC16. The Router Mask TLV contains an updated
36 current ID Sequence and Router ID mask that reflect the new ID assignment. The RLOC16
37 TLV and Router Mask TLV are included only if the status is 'Success'.
38 If a Router wishes to return to being a Child, it MUST first reconnect to the network as a
39 Child and then MUST send a CoAP POST Address Release Notification Message to the Leader
40 containing its no-longer used Router ID. Reconnecting before releasing avoids contention
41 with the Leader having to both mark the Router ID as unassigned and use it to send a
42 response.

5.9.10.3 ADDR_REL.ntf – Address Release Notification

The Address Release Notification message is a reliable notification formatted as follows:

CoAP Request URI

coap://[<leader address>]:MM/a/ar

Transaction Pattern

Ntf/Qry/Ans_CON

CoAP Payload

MAC Extended Address TLV

RLOC16 TLV

If a Router wishes to return to being a Child, it MUST first reconnect to the network as a Child and then MUST send an ADDR_REL.ntf to the Leader containing its RLOC16 Router address that it is no longer using.

5.10 Unicast Packet Forwarding

The Thread Specification supports two types of unicast packets forwarding:

- Packet forwarding when the source and destination devices are part of the same Thread Network
- Packet forwarding when the source or destination device is outside the Thread Network

5.10.1 Unicast Packet Forwarding inside the Thread Network

There will be a maximum of MAX_ROUTERS Routers at one time that share their routing information as described in Section 5.9, **Routing Protocol**. Inside the Thread Network, only Routers can perform multi-hop packet forwarding.

5.10.1.1 Full Thread Device Forwarding

Routing inside a Thread Network is done only using RLOCs. Multi-hop packet forwarding within Thread occurs at the link layer using the 6LoWPAN Mesh Header with RLOC16s. FTDs are required to resolve EID-to-RLOC address mappings and maintain an EID-to-RLOC Map Cache. FTDs also respond directly to EID-to-RLOC Map queries. FTDs perform IPv6 packet forwarding as follows:

- Obtain the IPv6 destination's RLOC. If the IPv6 destination address is an RLOC, the device uses the IPv6 destination address to route directly. If the IPv6 destination address is an ALOC, the device uses the lookup method for that type of ALOC to determine the RLOC of an appropriate destination to which to forward the message. If the IPv6 destination address is not an RLOC, the device SHOULD check its neighbor tables and EID-to-RLOC Map Cache to see if an RLOC is already known. If the device is not already known, the device initiates an EID-to-RLOC lookup to obtain the RLOC for the IPv6 destination address.
 - Determine if the RLOC is a neighboring device. If the RLOC belongs to a neighboring device, the 6LoWPAN layer encodes the 6LoWPAN frame without

- 1 using a 6LoWPAN Mesh Header and sets the IEEE 802.15.4 Destination using the
2 RLOC16.
- 3 ▪ If the RLOC does not belong to a neighboring device, the 6LoWPAN layer encodes
4 the 6LoWPAN frame with a 6LoWPAN Mesh Header, sets the V and F bits to '1',
5 the Originator Address field using the local RLOC16, and the Final Address field
6 using the destination's RLOC16.
7 The hopsLft field of the 6LoWPAN Mesh Header MUST be set to a value greater
8 than the route cost to the destination. The hopsLft field MUST be incremented by
9 one if the device is not an active Router and also MUST be incremented by one if
10 the destination RLOC16 is not that of an active Router. Note that setting the
11 hopsLft field to a value of 14 or less reduces the size of the 6LoWPAN Mesh
12 Header.
13 The device then determines the next hop RLOC using the Route Set and sets the
14 IEEE 802.15.4 Destination using the next hop's RLOC16.
- 15 ▪ The next hop device receives the packet, looks-up the next hop in the Route Set
16 / Neighbor Table, decrements the hop count in the 6LoWPAN Mesh Header, and
17 then sends the packet to the next hop as the IEEE 802.15.4 Destination.

18 **5.10.1.2 Minimal Thread Device Forwarding**

19 MTDs operate as host-only devices and forward all packets to their Parent without use of a
20 6LoWPAN Mesh Header. As a result, the Parent performs IPv6 forwarding (rather than
21 6LoWPAN mesh forwarding) for packets from an MTD Child.

22 Specifically, an MTD does not resolve EIDs to RLOCs and relies on its FTD Parent to perform
23 the lookup and include RLOC16s in a 6LoWPAN Mesh Header. An MTD does not choose a
24 Border Router for packets with off-mesh destinations and relies on its FTD Parent to choose
25 an appropriate Border Router. An MTD also does not choose a specific DHCPv6 Agent but
26 uses the DHCPv6 Agent ALOC instead and relies on its FTD Parent to deliver the packet to
27 an appropriate DHCPv6 Agent. An MTD does not choose a specific server but uses the
28 Service ALOC for the appropriate Service Type ID and relies on its FTD Parent to deliver the
29 packet to an appropriate server.

30 **5.10.1.3 Forwarding of Packets with Mesh Address 31 Headers**

32 Section 11 of [\[RFC 4944\]](#) describes how packets with Mesh Addressing headers are
33 forwarded using information from a "link-layer routing table". In a Thread Network, this
34 table maps the destination RLOC16 from the Mesh Addressing header to the RLOC16 of the
35 next hop along the route to the destination. On an active Router, this table is populated as
36 follows:

- 37 • If there is a route to Router R, then the link-layer routing table has entries for R's
38 RLOC16 and the RLOC16s that correspond to R's potential children. The next hop for
39 these entries is the next hop along the route to R.
40 • For each Mode bit D=1 child of this Router, there is an entry mapping the child's
41 RLOC16 to itself.

42 For FEDs, the next hop for all RLOC16s is the parent. MEDs do not receive or process mesh
43 headers and do not have a link-layer routing table.

1 **5.10.2 Unicast Packet Forwarding outside the** 2 **Thread Network**

3 Thread Border Routers MUST implement IP layer packet forwarding between the Thread
4 interface and other interfaces to ensure communication with an exterior network or the
5 Internet.

6 Thread Border Routers provide information to the Leader about the set of valid prefixes that
7 they serve as default routes and any more-specific routes that they provide for a given valid
8 prefix. The Leader aggregates the Border Router information and propagates this
9 information throughout the Thread Network partition. The Thread Network Data identifies
10 Border Routers using RLOC16s.

11 Outgoing messages from the Thread Network are forwarded at the link layer using the
12 6LoWPAN Mesh Header from the originator node to an appropriate Border Router, which
13 then forwards the packet to the other interface.

14 The incoming messages to the Thread Network are based on GUA or supplemental ULA
15 addresses and the Border Router uses its internal address tables or Address Query to
16 identify the RLOC of the final destination. A Border Router that offers a prefix for SLAAC or
17 DHCPv6 address configuration MAY send Address Queries for addresses configured using
18 that prefix whether or not the prefix is on-mesh. (For more information, see Section 9.2.3,
19 Global Addresses, and Section 9.2.4, Supplemental Unique Local Addresses, in Chapter 9,
20 Border Router.) Once that step has been completed, the Border Router forwards the packet
21 to the mesh destination using the 6LoWPAN Mesh Header algorithm described in Section
22 5.10.1, **Unicast Packet Forwarding inside the Thread Network**.

23 **5.11 Multicast Packets Forwarding**

24 **5.11.1 Link-Local Scope**

25 Messages sent to link-local scope multicast addresses are transmitted differently by [Routers](#)
26 and [End Devices](#). A Thread Router MUST transmit link-local scope multicasts as IEEE
27 802.15.4 broadcasts. With the exception of MLE multicasts, a Thread End Device MUST
28 transmit link-local scope multicasts as IEEE 802.15.4 unicasts to its parent. Link-local scope
29 MLE multicast messages, which are not secured by IEEE 802.15.4 and can thus be received
30 over uninitialized radio links, MUST be transmitted using IEEE 802.15.4 broadcasts by all
31 devices.

32 In all cases, MPL MUST NOT be used for messages sent to link-local scope multicast
33 addresses.

34 By default, the only link-local multicast messages that are sent using IEEE 802.15.4 indirect
35 transmissions are those sent to the Link-Local All Thread Nodes multicast address.

36 An rx-off-when-idle Child MAY register additional link-local multicast addresses with its
37 Parent by including them in the MLE Address Registration TLV. Parents MUST forward
38 multicast packets destined to registered multicast addresses using IEEE 802.15.4 indirect
39 transmissions.

5.11.2 Realm-Local and Larger Scopes

To forward multicast packets that have realm-local or larger scope, Thread devices MUST implement the Multicast Protocol for Low Power and Lossy Networks (MPL) specification as described by [\[RFC 7731\]](#) and [\[RFC 6206\]](#).

Only MPL proactive forwarding is used in Thread Networks; reactive forwarding is disabled. Any Thread devices MAY originate a MPL Data Message. All Thread devices MUST process MPL Data Messages that they receive. Thread Routers MUST forward MPL Data Messages received from other nodes. However, all Thread devices not operating as a Router MUST NOT forward MPL Data Messages.

Table 5-6 summarizes the modifications and statements to [\[RFC 7731\]](#) for Thread Networks. (See Section 2.5, Nomenclature for Tables Referencing Other Standards, in Chapter 2, Supporting Information, for an explanation of the "Status" column.)

Table 5-6. Modifications and Statements to [\[RFC 7731\]](#)

Chapter / Section	Title and Remarks / Modifications	Status
1	Introduction	I
2	Terminology	I
3	Applicability Statement <ul style="list-style-type: none">Hosts and REEDs are part of MPL domain (can transmit MPL messages but they cannot forward MPL messages).	I
4.1	MPL Domains <ul style="list-style-type: none">All Thread devices MUST consider all realm-local scope multicast addresses as MPL Domain Addresses, allowing Thread devices to avoid IP-in-IP encapsulation when forwarding any arbitrary multicast address with realm-local scope.All Thread devices MUST also configure the realm-local ALL_MPL_FORWARDERS multicast address (FF03::FC) to forward IPv6 multicast packets that have scope larger than realm-local. When forwarding multicast packets that have larger than realm-local scope, Thread devices MUST use IP-in-IP encapsulation.Reactive forwarding is not used, so the MPL interface must not subscribe to the MPL Domain Address with a scope value of 2 (link-local).	I, M
4.2	Information Base Overview	I
4.3	Protocol Overview <ul style="list-style-type: none">Only Proactive forwarding is used.	N, S

Chapter / Section	Title and Remarks / Modifications	Status
	<ul style="list-style-type: none"> Reactive forwarding is not used (control messages are not used). 	
4.4	<p>Signaling Overview</p> <ul style="list-style-type: none"> Control messages are not used. 	N, M
5.1	<p>MPL Multicast Addresses</p> <ul style="list-style-type: none"> Control messages are not used, so the MPL interface must not subscribe to the MPL Domain Address with a scope value of 2 (link-local). 	N, M
5.2	<p>MPL Message Types</p> <ul style="list-style-type: none"> Control messages are not used. 	N, M
5.3	<p>MPL Seed Identifiers</p> <ul style="list-style-type: none"> Seed Identifier used is RLOC16 of the node that introduces the packet in the MPL domain. It is unique within each MPL domain supported by the node. 	N
5.4	<p>MPL Parameters</p> <ul style="list-style-type: none"> CONTROL_MESSAGE_IMIN, CONTROL_MESSAGE_IMAX, CONTROL_MESSAGE_K and CONTROL_MESSAGE_TIMER_EXPIRATIONS constants are not used. All supported MPL domains use the same values for the Trickle Parameters. (Trickle Parameters are the same for the entire Thread network.) 	N, M
6.1	<p>MPL Option</p> <ul style="list-style-type: none"> S field is always set to '0' if the IPv6 Source Address is an RLOC, or '1' otherwise. M bit is always set to '0' on transmission and ignored on reception. seed-id field is elided if the IPv6 Source Address is an RLOC, or a 16-bit unsigned integer (RLOC16 of the interface that introduces the packet in the MPL domain), otherwise. 	N, M
6.2	MPL Control Message	N/R
6.3	MPL Seed Info	N/R
7.1	<p>Local Interface Set</p> <ul style="list-style-type: none"> Local Interface Set is informative. 	I

Chapter / Section	Title and Remarks / Modifications	Status
7.2	Domain Set <ul style="list-style-type: none"> • Domain Set structure is informative. 	I
7.3	Seed Set <ul style="list-style-type: none"> • Seed Set structure is informative. 	I
7.4	Buffered Message Set <ul style="list-style-type: none"> • Buffered Message Set structure is informative. • Sequence number validation (what happens when the minimum sequence number is incremented) is normative. 	N, I
8	MPL Seed Sequence Numbers	N
9.1	MPL Data Message Generation	N
9.2	MPL Data Message Transmission <ul style="list-style-type: none"> • Transmission time t is randomly chosen within full Trickle interval, rather than half the Trickle interval as specified in [RFC 6206]. When an interval begins, Trickle resets c to 0 and sets t to a random point in the interval, taken from the range $[0, I]$, that is, values greater than or equal to 0 and less than I. The interval ends at I. • M flag is always set to '0' (zero) on transmission and ignored on reception. 	N, M
9.3	MPL Data Message Processing	N
10.1	MPL Control Message Generation	N/R
10.2	MPL Control Message Transmission	N/R
10.3	MPL Control Message Processing	N/R
11	IANA Considerations <ul style="list-style-type: none"> • Control messages are not used. 	N, M
11.1	MPL Option Type	N
11.2	MPL ICMPv6 Type	N/R
11.3	Well-known Multicast Address	I

Chapter / Section	Title and Remarks / Modifications	Status
12	Security Considerations	I

1 All MPL domains supported by a Thread network MUST have the same MPL parameters
2 configured per Table 5-7.

3 **Table 5-7. Configuration of MPL Parameters**

MPL Parameter	Value	Observations
PROACTIVE_PROPAGATION	TRUE	Proactive Forwarding strategy is used
DATA_MESSAGE_IMIN	64ms	
DATA_MESSAGE_IMAX	64ms	
DATA_MESSAGE_K	infinite	
DATA_MESSAGE_TIMER_EXPIRATIONS	2 for Routers 0 for Hosts and REEDs	Forwarding is disabled for Hosts and REEDs
SEED_SET_ENTRY_LIFETIME	90 seconds	Value chosen to not exceed ID_REUSE_DELAY, for correct MPL operation.

4 MPL Option from IPv6 Hop-by-Hop header MUST be configured as follows:

- 5 • S field MUST be set to '0' if the IPv6 Source Address is an RLOC; otherwise, '1'.
- 6 • M bit MUST be set to '0' (zero) on transmission and MUST be ignored on reception.
- 7 • seed-id field MUST be elided if the IPv6 Source Address is an RLOC, or set to the RLOC16 of the node introducing the MPL data message in the MPL domain otherwise.

9 The IPv6 Hop Limit field of an MPL Data Message is decreased by one by an MPL Forwarder,
10 prior to starting the forwarding process for this message. A multicast packet can be sent by
11 an application using a specific IPv6 Hop Limit (e.g., set by the IPV6_MULTICAST_HOPS
12 socket option or an equivalent API method on the Thread Device). If the sending application
13 does not specify any specific Hop Limit value for a multicast packet, a Thread Device
14 SHOULD use the value MPL_HOPLIM_DEFAULT instead. For a message with a realm-local
15 scope destination address, an application-specified Hop Limit value is directly used in Hop
16 Limit field of the MPL Data Message. For a message with a larger than realm-local scope
17 destination address, the application-specified Hop Limit value MUST be used only for the
18 inner (encapsulated) IPv6 packet, while for the outer (encapsulating) MPL Data Message the
19 Hop Limit value SHOULD be set to the value MPL_HOPLIM_DEFAULT.

20 Messages sent to realm-local and larger scope multicast addresses are transmitted
21 differently by Routers and End Devices. A Thread Router MUST transmit realm-local and

- 1 larger scope multicasts as IEEE 802.15.4 broadcasts. A Thread End Device MUST transmit
2 realm-local and larger scope multicasts as IEEE 802.15.4 unicasts to its parent.
- 3 By default, the only realm-local or larger scope multicast messages that are sent using IEEE
4 802.15.4 indirect transmissions are those sent to the Realm-Local All Thread Nodes
5 multicast address.
- 6 An rx-off-when-idle Child MAY register additional link-local multicast addresses with its
7 Parent by including them in the MLE Address Registration TLV. Parents MUST forward
8 multicast packets destined to registered multicast addresses using IEEE 802.15.4 indirect
9 transmissions.

10 **5.12 Selection of Link-Layer Destination 11 Addresses**

12 The link layer destination of a packet being sent to an rx-off-when-idle child is the RLOC16
13 of the Child, regardless of the IP destination, and the message MUST be sent indirectly. This
14 includes packets whose IP destination is a multicast address.

15 As specified in Section 5.11, **Multicast Packets Forwarding**, messages with multicast IP
16 destinations are sent as 802.15.4 broadcasts and as 802.15.4 unicasts. For those sent as
17 broadcasts the link-layer destination is the broadcast address (0xFFFF). For those sent as
18 unicasts, the link-layer destination is the RLOC16 of the intended recipient.

19 If the IP destination is a link-local unicast address, the link-layer destination is the
20 corresponding MAC Extended Address, consisting of the IP destination's 64-bit IID with the
21 universal/local bit inverted. Note that Thread devices do not configure a link-local address
22 based on their 16-bit MAC address.

23 In all other cases, the next hop for a packet is an RLOC. If the packet has a mesh header,
24 the RLOC is that of the next hop towards the mesh header destination. Otherwise, it is the
25 determined by the unicast forwarding rules in Section 5.10, **Unicast Packet Forwarding**.
26 The link-layer destination is the corresponding RLOC16.

27 **5.13 Thread Network Data**

28 In a Thread Network, the Leader device collects, collates, and distributes information about
29 Border Routers and other servers available on the network back to the Thread Network.
30 Individual servers use the Thread Management Framework (TMF) messages to inform the
31 Leader of their capabilities and to receive confirmation of which of their capabilities will be
32 included in the Thread Network Data. (For more information on the TMF, see Chapter 10,
33 Management.) The Leader also manages the 6LoWPAN context ID assignments to IPv6
34 prefixes and includes them in the Thread Network Data. The Leader collates the Thread
35 Network Data and redistributes it to other Routers via MLE messages. Routers also keep
36 track of which version of the Thread Network Data each Child has and proactively unicast
37 newer versions as they become available.

38 Devices on a Thread Network can either maintain the full set of Thread Network Data or
39 limit themselves to a stable subset. As the name implies, the stable subset of the Thread
40 Network Data changes less frequently and requires less overhead to maintain. Routers
41 maintain the full set, while Children can choose which set they maintain. Because Router ID
42 assignments may change without notice, the stable Thread Network Data does not contain
43 unicast RLOCs of any particular device. RLOCs of DHCPv6 Agents are replaced by the

- 1 DHCPv6 Agent ALOC addresses. RLOCs of servers are replaced by the Service ALOC
2 addresses.
- 3 Information about the Leader, Border Routers, 6LoWPAN contexts, and servers is distributed
4 as Thread Network Data. Border Routers and servers inform the Leader of their services.
5 The Leader collates this data and distributes it to the Thread Network partition. The goal is
6 to provide:
- 7 • Valid and on-mesh prefixes assigned to the Thread Network
8 • IPv6 address configuration options for each valid prefix
9 • Identity of Border Routers providing default routes
10 • More-specific routes for each prefix
11 • Assignment of 6LoWPAN context IDs to prefixes
12 • Commissioning data for the Thread Network
13 • Type and identity of servers on the Thread Network
- 14 Updated Thread Network Data is propagated using a simple version of the Trickle algorithm.

15 **5.13.1 Version Number Set**

16 Each set of Thread Network Data has two associated version numbers, one for the full set of
17 data and the other for the stable subset. The Leader increments one or both of these
18 version numbers whenever it makes a change to the Thread Network Data. The version
19 numbers are one-byte values and MUST be incremented modulo 256. VN_version is
20 incremented for any change and VN_stable_version is incremented after any change to
21 stable version data as follows:

- 22 • VN_version is an 8-bit unsigned integer.
23 • VN_stable_version is an 8-bit unsigned integer.
24 VN_stable_only is a Boolean value that is true if the node has only the stable Thread
25 Network Data and 'false' if it has the full Thread Network Data.

26 **5.13.2 Valid Prefix Set**

27 A network's Valid Prefix Set contains the prefixes that are available to nodes on the Thread
28 Network, along with the Border Routers that provide them. A Thread device maintains one
29 or more tuples as follows:

30 (*P_prefix*, *P_domain_id*, *P_border_router_16*, *P_stable*, *P_on_mesh*, *P_preferred*,
31 *P_slaac*, *P_dhcp*, *P_configure*, *P_default*, *P_preference*, *P_nd_dns*)

32 where:

- 33 • *P_prefix* is the IPv6 prefix that is available on the Thread Network.
34 • *P_domain_id* is an 8-bit unsigned integer identifying the provisioning domain with
35 which this tuple is associated.
36 • *P_border_router_16* is the RLOC16 of the Border Router that makes the prefix
37 available.
38 • *P_stable* is 'true' if the default route is expected to be available for at least
39 MIN_STABLE_LIFETIME; otherwise, 'false'.
40 • *P_on_mesh* is 'true' if *P_prefix* is considered to be on-mesh; if 'false' then *P_prefix* is
41 not considered to be on-mesh.

- 1 • P_preferred is 'true' if addresses autoconfigured using P_prefix are considered
2 preferred address; if 'false', then such addresses are considered deprecated.
- 3 • P_slaac is 'true' if network devices are allowed to autoconfigure addresses using
4 P_prefix; otherwise, 'false'.
- 5 • P_dhcp is true if P_border_router is a DHCPv6 Agent that manages address
6 configuration for P_prefix; otherwise, 'false'.
- 7 • P_configure is true if P_border_router is a DHCPv6 Agent that supplies other
8 configuration data, such as the identity of DNS servers; otherwise, 'false'.
- 9 • P_default is true if P_border_router offers a default route for messages whose IP
10 source uses P_prefix; otherwise, 'false'.
- 11 • P_preference is a two-bit signed integer indicating Router preference as defined in
12 [\[RFC 4191\]](#). It is valid only if P_default is 'true' as follows:

1	(01)	High
0	(00)	Medium (default)
-1	(11)	Low
-2	(10)	Reserved - MUST NOT be used

- 13 • P_nd_dns is true if P_border_router is able to supply DNS information obtained via
14 ND.

15 A valid prefix MUST NOT allow both DHCPv6 and SLAAC for address configuration: for any
16 given valid prefix either all P_DHCP values will be 'false' or all P_SLAAC values will be 'false'.

17 A valid prefix MAY use manual or other means of configuration, in which case all P_DHCP
18 and P_SLAAC values will be 'false'.

19 **5.13.3 External Route Set**

20 A network's External Route Set contains the available external routes off the Thread
21 Network, including both Border Routers and the prefixes for which they will route. The
22 Leader maintains a tuple for each route as follows:

23 (R_domain_id, R_border_router_16, R_prefix, R_stable, R_preference)

24 where:

- 25 • R_domain_id is an 8-bit unsigned integer identifying the provisioning domain with
26 which this tuple is associated.
- 27 • R_border_router_16 is the RLOC16 of the Border Router.
- 28 • R_prefix is the IPv6 prefix for the route.
- 29 • R_stable is set to 'true' if the route is expected to be available for at least
30 MIN_STABLE_LIFETIME; otherwise, set to 'false'.
- 31 • R_preference is a two-bit signed integer indicating Router preference as defined in
32 [\[RFC 4191\]](#).

33 **5.13.4 6LoWPAN Context ID Set**

34 A network's 6LoWPAN Context ID Set contains the network's 6LoWPAN context IDs and the
35 prefixes they represent. The Leader maintains a tuple for each context ID as follows:

36 (CID_id, CID_prefix, CID_compress, CID_stable)

1 where:

- 2 • CID_id is a 6LoWPAN context ID.
3 • CID_prefix is the IPv6 prefix which the CID_id encodes.
4 • CID_compress is a Boolean value. If set to 'true', this context MAY be used for
5 compressing outgoing messages. If not set, this context is only to be used for
6 uncompressing incoming messages.
7 • CID_stable is set to 'true' if the 6LoWPAN context ID is expected to be valid for at
8 least MIN_STABLE_LIFETIME; otherwise, set to 'false'.

9 **5.13.5 Server Set**

10 A network's server set contains the available servers on the network. Servers may either be
11 of a standard Thread type or specific to a particular vendor. The Leader maintains a tuple
12 for each server as follows:

13 (S_enterprise_number, S_service_data, S_server_16, S_server_data, S_stable,
14 S_id)

15 where:

- 16 • S_enterprise_number is the Private Enterprise Number assigned by IANA to the
17 vendor that defined the type of the server. Standard Thread server types use
18 THREAD_ENTERPRISE_NUMBER (44970) for this value.
19 • S_service_data is a byte string specifying the type of service along with any
20 associated data. The format of S_service_data for Thread servers will be defined in a
21 later revision of this specification.
22 • S_server_16 is the RLOC16 of the server.
23 • S_server_data is a byte string containing server-specific information. The format of
24 S_server_data for Thread server types will be defined in a later revision of this
25 specification.
26 • S_stable is set to 'true' if the server is expected to be available for at least
27 MIN_STABLE_LIFETIME; otherwise, set to 'false'.
28 • S_id is the Service Type ID, a value between 0 and 15, inclusive, assigned to this
29 S_service_data by the Leader. All servers with the same S_enterprise_number and
30 S_service_data share the same value of S_id.

31 **5.13.6 Commissioning Data**

32 A network's Commissioning Data contains information about the current state of
33 commissioning on the network. This is an opaque sequence of bytes as follows:

- 34 • COM_length is an 8-bit unsigned integer.
35 • COM_data is sequence that contains COM_length 8-bit unsigned integers.

36 For more information, see Chapter 8, Mesh Commissioning Protocol.

37 **5.13.7 Provisioning Domain**

38 A Provisioning Domain is a consistent set of configuration information about the Thread
39 Network provided by one or more Border Routers. The configuration information may

1 include valid and on-mesh prefixes, IPv6 address configuration information, and external
2 routes.
3 A Border Router that offers an external route has to specify what source prefix(es) can be
4 used when sending over that route. It does that by including Prefix TLVs that have the same
5 Provisioning Domain ID (P_domain_id). The Border Router does not necessarily need to
6 offer SLAAC or DHCPv6 for the source prefix(es), but it does need to communicate that the
7 prefix(es) within the Prefix TLVs are the valid source-address prefix(es) for the offered
8 route.
9 Provisioning Domains are needed because different Border Routers may offer the same
10 external route using different source prefixes. Also, a single Border Router may offer two
11 prefixes that have different associated external routes.
12 Because Thread Management messaging is not instantaneous, every set of Network Data
13 (set of TLVs) needs to be self-contained. The Provisioning Domain IDs are only used to
14 associate TLVs within a single set of Network Data, not to associate between sets.

15 **5.14 Stable Thread Network Data**

16 The Thread Network Data can change on multiple time scales. For example, a Border Router
17 may be a permanent part of a network, or it may be a mobile device that is only on the
18 network for a brief period. Similarly, network devices vary in their need for, and ability to
19 handle, timely updates. For example, an energy-limited, rx-off-when-idle Child that
20 communicates with its Parent once or twice a day should not be burdened with useless
21 updates. For this reason, the Thread Network Data is divided into two categories: stable and
22 temporary. Data is considered stable if it is expected to remain valid for at least
23 MIN_STABLE_LIFETIME hours; otherwise, the data is considered temporary.

24 Devices either have the complete Thread Network Data or the stable subset. All Routers and
25 REEDs MUST have the complete Thread Network Data. Rx-on-when-idle end devices receive
26 the complete Thread Network Data in the same manner as REEDs, but MAY choose to store
27 only the stable subset. Rx-off-when-idle end devices inform their Parent whether they wish
28 to receive all Thread Network Data updates or only updates to the stable data. If a device
29 has only the stable data, its Network Database has the following properties:

- 30 • VN_stable_only is true.
- 31 • R_stable, P_stable, S_stable, and CID_stable are true.
- 32 • R_border_router_16 values are not used and set to the invalid value of 0xFFFF.
- 33 • P_border_router_16 values for Border Routers that do not serve as DHCPv6 Agents
34 are not used and set to the invalid value of 0xFFFF.
- 35 • P_border_router_16 values for Border Routers that do serve as DHCPv6 Agents are
36 DHCPv6 Agent ALOC16s.
- 37 • S_server_16 values are Service ALOC16s.

38 All RLOC16s are considered temporary. In the stable subset of the Thread Network Data,
39 R_border_router_16 values are not used and External Route entries are only used to
40 determine the availability of an external route. P_border_router_16 values for Border
41 Routers that do not offer DHCPv6 services are not used and corresponding Valid Prefix
42 entries are only used for IPv6 address configuration information. P_border_router_16 values
43 for Border Routers that do offer DHCPv6 services are replaced by the DHCPv6 Agent
44 ALOC16s. Similarly, S_server_16 values for servers are replaced by Service ALOC16s.

- 1 To extract the stable subset of the Thread Network Data, either because a device that has
2 received the full data wants to store only the stable subset, or because a Router is sending
3 the stable subset to a Child that has requested it, the device performs the following actions:
4
 - Remove all 6LoWPAN Context ID Set entries for which CID_stable is ‘false’.
 - Remove all External Route Set entries for which R_stable is ‘false’.
 - Remove all Valid Prefix Set entries where P_stable is ‘false’.
 - Replace all R_border_router_16 values with 0xFFFFE.
 - Replace all P_border_router_16 values for Border Routers that do not serve as
9 DHCPv6 Agents with 0xFFFFE.
 - Replace all P_border_router_16 values for Border Routers that serve as DHCPv6
10 Agents with DHCPv6 Agent ALOC16s.
 - Replace all S_server_16 values with Service ALOC16s.

13 **5.15 Network Data and Propagation**

14 Network Data is propagated using MLE messages. Routers advertise their current Network
15 Data version in MLE Advertisement messages and nodes hearing an MLE Advertisement with
16 a newer version number request the newer data from the sender.

17 **5.15.1 Modifying Network Data**

18 Whenever the Leader modifies the Thread Network Data it MUST increment VN_version and,
19 if there was any change to stable data, MUST also increment VN_stable_version. If the
20 Leader has any neighbors or rx-on-when-idle Children, it MUST multicast a single one-hop
21 MLE Data Response message containing the full Thread Network Data in an MLE Network
22 Data TLV.

23 When a Router receives new Thread Network Data, it waits for a random jitter of up to 500
24 milliseconds and then sends a single one-hop broadcast containing the new Thread Network
25 Data. Routers also unicast the new network data to all rx-off-when-idle Children.

26 In addition, Thread Network Data version numbers are included in an MLE Leader Data TLV
27 in all MLE advertisement, request, and accept messages that are sent. If a node receives an
28 MLE message that shows that the sender has newer Thread Network Data, the node replies
29 with a request for the complete Thread Network Data. For more information, see Chapter 4,
30 Mesh Link Establishment.

31 **5.15.2 Propagation to rx-on-when-idle Devices**

32 Nodes sending MLE Advertisement and MLE handshake messages include an MLE Leader
33 Data TLV, which contains, among other values, the sender’s VN_version and
34 VN_stable_version values. When an MLE Leader Data TLV is received, the recipient MUST
35 compare the included VN_version number with its own local value. If the received
36 VN_version number is higher (Version number X is more recent than Version Number Y if:
37 $0, ((x-y) \bmod 256 \leq 127)$

5.15.3 Propagation of Operational Datasets

The Active Operational Dataset (Section 8.4.2.2.2 in Chapter 8, Mesh Commissioning Protocol) and, if valid, the Pending Operational Dataset (Section 8.4.2.2.3 in Chapter 8, Mesh Commissioning Protocol) are propagated in tandem with the Network Data. Because the Operational Datasets change much less frequently than the Network Data, the Operational Datasets are included in messages with the Network Data only when it is known that the recipient requires them. In all other cases only the Active Timestamp and, if valid, the Pending Timestamp are included. The timestamps are also included in requests for the Network Data, so that the recipient of the request can determine if either or both of the Operational Datasets need to be included in the response.

Then, if the incoming message also includes an MLE Network Data TLV and contains an MLE Active Timestamp TLV that matches the local Active Timestamp and optionally an MLE Pending Timestamp TLV that matches the local Pending Timestamp, then the local Network Database is replaced with the contents of the MLE Network Data TLV. If the message also contains an MLE Active Operational Dataset TLV and optionally an MLE Pending Operational Dataset TLV, then the Operational Datasets are also replaced by the contents of the respective TLV.

If the received VN_version number is higher, but the above criteria for using the MLE Network Data TLV are not met, then an MLE Data Request message SHOULD be sent to the original message's sender, requesting that an MLE Network Data TLV be sent. The request MUST also contain an MLE Active Timestamp TLV and, if valid, an MLE Pending Timestamp TLV with the local timestamp's respective current values. The node MAY opt to not send a request if it has already done so and is waiting for a reply.

If R_stable is true, the recipient of an MLE Leader Data TLV MUST use the received and local VN_stable_version values for comparison and MUST convert any newly received Network Data to the stable subset before storing it in the relevant data sets.

When a Router receives new Thread Network data it MUST multicast a single one-hop MLE Data Response message to the Link-Local All Nodes multicast address (FF02::1) containing the full Thread Network Data in an MLE Network Data TLV. The message must also contain an MLE Leader Data TLV. Regarding inclusion of the Active Timestamp and Active Operational Dataset TLVs:

- If the MLE Data Request contains an MLE Active Timestamp TLV that matches the Active Timestamp, then the MLE Data Response MUST contain an identical MLE Active Timestamp TLV. The MLE Data Response SHALL NOT contain an MLE Active Operational Dataset TLV.
- Otherwise, the MLE Data Response MUST contain an MLE Active Timestamp TLV with the value of the local Active Timestamp and an MLE Active Operational Dataset TLV with the complete local Active Operational Dataset, excluding the local Active Timestamp.

Regarding inclusion of the Pending Timestamp and Pending Operational Dataset TLVs, if the local Pending Operational Dataset is not valid, then the response MUST NOT contain either an MLE Pending Timestamp TLV or an MLE Pending Operational Dataset TLV. Otherwise:

- If the MLE Data Request contains an MLE Pending Timestamp TLV that matches the Pending Timestamp, then the MLE Data Response MUST contain an identical MLE Pending Timestamp TLV. The response SHALL NOT contain an MLE Pending Operational Dataset TLV.

- 1 • Otherwise, the MLE Data Response MUST contain an MLE Pending Timestamp TLV
2 with the value of the local Pending Timestamp and an MLE Pending Operational
3 Dataset TLV with the complete local Pending Operational Dataset, excluding the local
4 Pending Timestamp.

5.15.4 Propagation to rx-off-when-idle Devices

6 Each Thread Parent maintains a Child Version Number Set that contains the following tuple
7 for each rx-off-when-idle child:

8 (C_VN_full_data, C_VN_version_is_valid, C_VN_version)

9 where:

- 10 • C_VN_full_data is ‘true’ if the child has requested the full Network Data (N flag set to
11 1 in the most recent MLE Mode TLV received from the child) and ‘false’ otherwise.
12 • C_VN_version_is_valid is ‘true’ if the value of C_VN_version was received from the
13 child and ‘false’ otherwise.
14 • C_VN_version is an 8-bit unsigned integer identifying the Network Data Version most
15 recently received from the child in an MLE Leader Data TLV. If C_VN_stable_only is
16 ‘true’ this is the value of the Stable Data Version field, otherwise it is the value of the
17 Data Version field.

18 A child's network version is out of date if any one of the following holds:

- 19 • C_VN_version_is_valid is ‘false’.
20 • C_VN_version_is_valid is ‘true’, C_VN_full_data is ‘true’, and C_VN_version does not
21 equal VN_version.
22 • C_VN_version_is_valid is ‘true’, C_VN_full_data is ‘false’, and C_VN_version does not
23 equal VN_stable_version.

24 Routers use the Child Version Number Set to determine when new Thread Network Data
25 needs to be sent to a Child. For each Child whose network version is out-of-date, the Router
26 MUST send an MLE Network Data TLV and an MLE Leader Data TLV using either MLE Child
27 Update Request or MLE Data Response unicast messages containing the TLVs until it
28 receives a Leader Data TLV with the updated VN_version or VN_stable_version from the
29 Child.

5.15.5 Leader and Thread Network Data

31 Upon becoming a Leader, a device initializes the Thread Network Data to be empty. If the
32 device is a server or Border Router, it adds in its own server data, exactly as if it had sent
33 the data to itself.

34 Whenever new server data is received from a server, via a Constrained Application Protocol
35 (CoAP) POST message, the Leader first removes all existing data concerning that server
36 from the Thread Network Data and then incorporates the new data. If the new data is
37 empty, this has the effect of removing the server from the Thread Network Data. The
38 Leader MUST remove any of the server's old server data that is not in the newly-received
39 server data. The Leader SHOULD incorporate any new data into the Thread Network data.
40 New data MUST be rejected if incorporating it would make the resulting Thread Network
41 Data TLV longer than MAX_NETWORK_DATA_SIZE.

42 In incorporating new Thread Network Data from a server, the Leader MAY change the
43 Domain ID values but MUST maintain the same relationships between prefix TLVs. The

- 1 Leader MUST assign the same P_domain_id for all Border Routers associated with the same prefix.
- 3 If the routing layer reports that the cost to a server has become infinite for INFINITE_COST_TIMEOUT seconds, that server is considered unreachable and its data is removed.
- 6 If the Leader deallocates a server's Router ID, or the Router ID of a server's Parent (if the server is a Child), all data concerning that server MUST be removed from the Thread Network Data.
- 9 The Leader also manages the allocation and deallocation of 6LoWPAN contexts. Whenever the Thread Network Data changes in any way, the Leader MUST increment the VN_version and initiate the distribution of the Thread Network Data.

12 **5.15.6 Server Behavior**

- 13 A server is any node that contributes Thread Network Data to the leader. A server has its own local server data, which is a fragment of Thread Network Data containing its own information. The Thread Network Data may consist of one or more Prefix TLVs, each containing a Border Router TLV or Has Route TLV.
- 17 The local server data never contains a 6LoWPAN ID TLV, because only the Leader can assign 6LoWPAN context IDs.
- 19 There are two situations under which the local server data is sent to the Leader:
 - 20 • If the local server data changes for any reason.
 - 21 • If new Thread Network Data is received that is not consistent with the local server data. There is an inconsistency if the received Thread Network Data either lists the local node as having services that are not in the local server data or if there are services in the local server data that are not included in the Thread Network Data.

25 **Note: Children can act as servers, including offering routing and acting as DHCPv6 Agents.**

27 **5.15.6.1 SVR_DATA.ntf – Server Data Notification**

28 Whenever a server's local server data does not match that in the Network Data, either 29 because the local server data has changed or because new Network Data that does not 30 contain matching data has been received, the server sends a SVR_DATA.ntf to the Leader, 31 with the payload of the message containing the server's local server data:

- 32 CoAP Request URI
 - 33 coap://[<leader address>]:MM/a/sd
- 34 Transaction Pattern
 - 35 Ntf/Qry/Ans_CON
- 36 CoAP Payload
 - 37 [Thread Network Data TLV]
 - 38 [RLOC16 TLV]

39 The CoAP payload consists of an optional Thread Network Data TLV containing Network Data 40 TLVs that the Leader is requested to add to the Network Data, with an optional RLOC16 41 TLV.

1 A Prefix TLV is included if the sender is a Border Router. It specifies a prefix and contains
2 Border Router TLVs and/or Has Route TLVs indicating the capabilities and routes provided
3 for that prefix. The Border Router TLVs and Has Route TLVs MUST contain only the sender's
4 RLOC16. There may be more than one Prefix TLV in the same payload.
5 A Service TLV is included if the sender is providing some non-Border Router service. It
6 contains a single Server TLV with the sender's RLOC16.
7 The optional RLOC16 TLV is used to update the Leader when a server's RLOC16 changes. A
8 server that changes its RLOC16 sends the Leader an SVR_DATA.req with its Prefix TLV
9 and/or Service TLV and includes an RLOC16 TLV containing its old RLOC16. Similarly, a
10 Parent that times out a child whose RLOC16 appears in the Network Data sends the leader
11 an SVR_DATA.req containing only an RLOC16 TLV with the child's RLOC16.
12 When the Leader receives an SVR_DATA.req containing an RLOC16 TLV, it removes all
13 references to that RLOC16 from the network data. If the SVR_DATA.req contains other
14 TLVs, the Leader then continues processing the message as if the RLOC16 TLV were not
15 present.
16 The latency and cost of distributing new Thread Network Data make frequent rapid changes
17 in the Network Data undesirable. A Border Router MUST have a valid lifetime of at least
18 MIN_PREFIX_LIFETIME seconds for a prefix in order to set either the P_slaac flag or P_dhcp
19 flags. A Border Router MUST have a preferred lifetime of at least MIN_PREFIX_LIFETIME
20 seconds for a prefix in order to set the P_preferred flag.
21 A server MUST wait at least DATA_RESUBMIT_DELAY seconds before resending its data,
22 should it fail to receive an updated Network Data containing the local data sent to the
23 Leader.

24 **5.15.7 Router Behavior**

25 Routers use the route information in the Network Data along with their mesh routing cost
26 and next hop information in making forwarding decisions.
27 If the IPv6 Destination Address is on-mesh, then the mesh destination address is obtained
28 either from the EID-to-RLOC Map Cache or by sending an Address Query message.
29 If the IPv6 Destination Address is not on-mesh, then the Router uses the IPv6 Source
30 Address to determine the set of Border Routers that offer routes associated with the same
31 provisioning domain as the prefix of the IPv6 Source Address.
32 Within that set, the Router finds the longest External Route prefix match as follows:
33

- If two prefixes match, choose the one with higher preference.
- If no External Route matches, choose the Border Router that offers a Default Route.
- If more than one Border Router offers a default route, choose the one with higher
36 preference.
- Finally, if two or more Border Routers remain, choose the one with lowest mesh path
38 cost.

39 If the IPv6 Destination Address is a DHCPv6 Agent RLOC, the Router uses the Context ID
40 encoded in the ALOC to determine the set of DHCPv6 Agents with the same provisioning
41 domain as the prefix associated with the Context ID. Within that set, the Router finds the
42 "closest" DHCPv6 Agent as follows:
43

- If more than one DHCPv6 Agent offers a default route, choose the one with higher
44 preference.

- 1 • If two or more DHCPv6 Agents remain, choose the one with lowest mesh path cost.

2 **5.15.8 Host Behavior**

3 Hosts use the Valid Prefix Set in configuring addresses, either autonomously or by using
4 DHCPv6, depending on the values of P_slaac and P_dhcp for the prefix. Hosts use the
5 6LoWPAN context information when compressing and uncompressing packets. Hosts MAY
6 use the Valid Prefix Set to locate DHCPv6 Agents from which to obtain other configuration
7 information, such as DNS servers and search paths.

8 Hosts MAY use the Valid Prefix Set to locate Border Routers that are able to supply DNS
9 information obtained via the IPv6 Neighbor Discovery (ND) protocol. This information MAY
10 be obtained by sending an ND_DATA.req to one or more Border Routers for which P_nd_dns
11 is 'true'. The ND_DATA.req should contain the ND option types for the Recursive DNS Server
12 Option (25) and the DNS Search List Option (31). These options and their formats are
13 described in [\[RFC 6106\]](#).

14 **5.15.8.1 ND_DATA.req (/nd) – Neighbor Discovery Data 15 Request**

16 CoAP Request

17 `coap://[<border router address>]:MM/a/nd`

1 CoAP Payload
2 ND Option TLV
3 The CoAP payload consists of an ND Option TLV indicating the ND options that are being
4 requested.

5 **5.15.8.2 ND_DATA.rsp (/nd) – Neighbor Discovery Data 6 Response**

7 CoAP Response
8 2.04
9 CoAP Payload
10 ND Data TLV
11 The CoAP payload consists of an ND Data TLV containing the ND options that were
12 requested. Any requested options that were not available are omitted.

13 **5.15.9 6LoWPAN Contexts**

14 The Leader is responsible for managing the assignment of 6LoWPAN context IDs. The
15 Leader MUST allocate a 6LoWPAN context ID for every prefix in the Valid Prefix Set;
16 additional prefixes MUST NOT be added to the set if no 6LoWPAN context ID can be
17 assigned. The Leader MAY allocate a 6LoWPAN context ID for every other prefix in the
18 Network Data, as long as there are unallocated IDs available.

19 Before removing a 6LoWPAN ID TLV from the Network Data, the Leader MUST first
20 distribute new Network Data with that context ID's Compress flag set to '0' (zero). A Leader
21 MUST wait at least CONTEXT_ID_REUSE_DELAY seconds after first distributing a 6LoWPAN
22 ID TLV with a Compress flag of zero before reallocating that context ID to a new prefix. In
23 particular, if a prefix no longer appears in the Valid Prefix Set and the External Route Set,
24 that prefix must be retained in the 6LoWPAN Context ID Set with CID_compress set to
25 'false' for at least CONTEXT_ID_REUSE_DELAY seconds before being removed from that
26 set. This avoids the race conditions inherent in the variable rates at which new network data
27 is propagated through the network.

28 When compressing a message for transmission to a Child for which C_VN_stable_only is
29 'true', a Router MUST NOT use a 6LoWPAN context ID for which CID_stable is 'false'.

30 **5.16 Thread Network Partitions**

31 While the intent is that all nodes in a Thread Network stay connected with one another, it is
32 possible that a network may consist of two or more disconnected fragments or Partitions.
33 Thread is designed so that each Partition can operate as a separate network and so that
34 separate Partitions that become connected merge together into a single connected Partition.
35 Each Partition forms a separate mesh network with its own Leader, Thread Network Data,
36 and Router ID assignments.

37 Because of this, messages are not forwarded between Partitions. The goal is for every
38 collection of connected nodes in a Thread Network to rapidly assemble into a single
39 Partition. To make this happen, the individual nodes in a Thread Network respond to
40 changes in connectivity by forming new Partitions and merging existing ones.

1 The core partitioning algorithm is as follows:

- 2 • Each Partition has a single node designated as the Leader of that Partition.
- 3 Membership in a Partition is determined by connectivity to the Leader: a node that
- 4 loses its connectivity to the Leader is no longer in that Partition.
- 5 • Partitions are ranked by priority; for any two Partitions, one has a higher priority and
- 6 the other has a lower priority. The priority ranking is permanent and does not
- 7 change over time. Router-capable devices always join the highest priority Partition
- 8 that is available. If a node in one Partition learns of a neighboring node that is in a
- 9 higher-priority Partition, it leaves its current Partition and join its neighbor's
- 10 Partition.
- 11 • A node that loses its connection to the Leader of its Partition joins the highest
- 12 priority Partition available amongst its neighbors. If there is no Partition available,
- 13 the node starts its own, with itself as the Leader.

14 5.16.1 Losing Connectivity

15 For a Router or REED, Partition membership is determined by connectivity to the Partition's
16 Leader. A Router or REED that loses connectivity to the Leader MUST either attach to some
17 other Partition, or, if there are no other Partitions within range, form a new one. Children
18 that are not REEDs need only to maintain connectivity to their Parent; they are always in
19 the same Partition as their Parent and cannot form their own Partition.

20 Connectivity to the Leader is detected by tracking changes to the ID sequence number that
21 is periodically incremented by the Leader and distributed throughout the Leader's Partition
22 via MLE advertisements. If a Router or REED fails to receive a new ID sequence number
23 within NETWORK_ID_TIMEOUT seconds, it is considered to be disconnected from the Leader
24 and MUST attempt to reattach to a Parent in the same Partition but with a newer ID
25 sequence number. If the reattachment fails, the Router MUST attempt to attach to any
26 available partition, or to a Parent in the same partition but with a newer ID sequence
27 number. Finally, if there are no available Partitions, the Router MUST start a new Partition
28 with itself as the Leader (see Section 5.15.2, **Starting a New Thread** Network Partition).
29 After starting a new Partition, the Leader MUST ignore any advertisements from the
30 previous Partition with the ID sequence number that was last heard before the Router timed
31 out.

32 A Router attempts to reconnect to its Partition by locating a REED that is still connected to
33 the Leader and requesting that that Child become a Router. This is identical to the attaching
34 process used by Children described in Chapter 4, Mesh Link Environment except:

- 35 • The Scan Mask TLV in the initial MLE Parent Request has both the E and R flags set
36 (there is no need to do an initial request with just the R flag set; if any neighboring
37 Router was connected to the Leader, the attachment process would not be needed).
- 38 • The initial MLE Parent Request MUST be delayed by a random period between 0 and
39 1000 ms. This is to avoid inadvertent synchronization when several nodes detect the
40 loss of Leader connectivity at the same moment.
- 41 • Potential Parents whose ID sequence number as reported in the connectivity TLV is
42 no more recent than the reattaching Router MUST be ignored.

43 If a Router reconnects to its current Partition after losing connectivity and it does not meet
44 the criteria for downgrading to a REED (see Section 5.9.9, **Router ID Management**), it
45 MUST request its original Router ID from the Leader. This minimizes network disruptions
46 caused by temporary loss of connectivity to the Leader.

1 A Router MUST make two separate attempts to reconnect to its current Partition, or a higher
2 priority one, in this manner. If both attempts fail, the Router attempts to attach to any
3 other Partition within range, including a lower priority one, using the attachment process
4 described in Chapter 4, Mesh Link Environment.

5.16.2 Starting a New Thread Network Partition

6 If a Router or REED is unable to attach to any existing Thread Network Partition, it MUST
7 create its own Thread Network Partition with itself as the Leader. The device deletes all
8 information relating to its previous Thread Network Partition (see Section 5.16.4, **Resetting**
9 **Thread Network Partition Data**). The device then:

- 10 • Chooses a random Partition ID.
- 11 • Chooses a random initial VN_version and VN_stable_version.
- 12 • Chooses a random initial ID_sequence_number.
- 13 • Initializes the Thread Network Data with its own server data, if any.

14 The Leader chooses the initial VN_version, VN_stable_version, and ID_sequence_number
15 randomly to help detect situations where different Partitions have the same Partition ID.

16 The Leader sets the weight value for the Thread Network Partition to 64. All other weight
17 values are reserved to be used for Thread Network Partition prioritization methods in a
18 future revision of this specification.

19 The Router MAY keep its Router ID so that its Children will remain attached to it. The Router
20 then sends an MLE advertisement and begins to function in the new Partition.

5.16.3 Merging Thread Network Partitions

22 Thread Network Partitions are merged by having individual devices move to neighboring
23 Thread Network Partitions. Whenever a Router or REED receives an MLE Advertisement
24 from a Router or REED in a neighboring Partition with a higher-priority than its current
25 Thread Network Partition, it attempts to attach to a Parent in that higher-priority Thread
26 Network Partition, and leaves its current Thread Network Partition if successful. If a device
27 moves to another Thread Network Partition, it deletes all information related to its previous
28 Thread Network Partition (see Section 5.16.4, **Resetting Thread Network Partition**
29 **Data**).

30 To prevent a single device with poor connectivity from repeatedly disrupting a network,
31 Thread Network Partitions that have only one Router are given a lower priority than larger
32 Thread Network Partitions. A "singleton" Thread Network Partition is a Thread Network
33 Partition that consists of a single Router, the Thread Network Partition's Leader, which has
34 no REED children.

35 The rules for determining the relative priority of two Thread Network Partitions are as
36 follows:

- 37 • Rule 1: A non-singleton Thread Network Partition always has higher priority than a
38 singleton Thread Network Partition.
- 39 • Rule 2: When comparing two singleton or two non-singleton Thread Network
40 Partitions, the one with the higher 8-bit weight value has higher priority.
- 41 • Rule 3: When comparing two singleton or two non-singleton Thread Network
42 Partitions that have the same 8-bit weight value, the one with the higher Partition
43 ID, considered as unsigned 32-bit numbers, has higher priority.

1 The Leader of a Thread Network Partition can accurately determine if it is in a singleton
2 Thread Network Partition; it knows if there are other Routers or REEDs in the Thread
3 Network Partition. A REED or non-leader Router is necessarily in a non-singleton Partition. If
4 an MLE Advertisement or MLE Parent Response is received from the Leader of a Thread
5 Network Partition and the MLE Routing TLV (for MLE Advertisements) or MLE Connectivity
6 TLV (for MLE Parent Responses) indicate that there is only one Router, that Thread Network
7 Partition is assumed to be a singleton Partition, regardless of the possibility of there being
8 one or more REED children.

9 In the case that two distinct Partitions have the same weight and Partition ID, it is likely
10 that the VN_version, VN_stable_version, and/or ID_sequence_numbers will differ because
11 they are updated based on actions by devices in their respective Thread Network Partitions.
12 When receiving MLE messages carrying an MLE Leader Data TLV, the Leader compares the
13 VN_version, VN_stable_version, and ID_sequence_numbers against its own. If any of the
14 values are greater than the Leader's values, the Leader assumes another Leader exists that
15 is maintaining a Thread Network Partition with the same Partition ID and weight. When a
16 Leader detects such inconsistencies, the Leader MUST start a new Thread Network Partition
17 as described in Section 5.16.2, **Starting a New Thread Network Partition**.

18 **5.16.4 Resetting Thread Network Partition Data**

19 When a Router or REED moves from one Thread Network Partition to another, or becomes
20 the Leader of a new Thread Network Partition, it must delete all data related to its previous
21 Thread Network Partition. This includes:

- 22 • Network Data
23 • Neighbor and routing data
24 • Messages queued for MAC transmission
25 • Address cache

26 Additionally, a REED must delete all data related to its previous Parent.

27 When a Router that has Child devices moves to a new Thread Network Partition, it
28 immediately requests its original Router ID in the new Thread Network Partition. If the
29 Router succeeds in getting the same Router ID assigned to it, the Child devices will stay
30 attached. If it gets no Router ID or a different Router ID, then the Router clears its Child
31 table. Its Child devices will eventually detect that their Parent link has failed and then
32 reattach.

33 Resetting Thread Network Partition data does not automatically reset IP-layer configuration
34 information, such as IPv6 addresses. When a device loses connectivity with a Thread
35 Network Partition, it maintains its current set of IPv6 addresses assigned to the Thread
36 interface. When the same device regains connectivity with a Thread Network Partition, it
37 checks that the set of IPv6 addresses is consistent with the new Thread Network Data
38 obtained from the new Thread Network Partition. The device removes any IPv6 addresses
39 that do not match a valid prefix in the new Thread Network Partition.

40 **5.16.5 Loss of Leader**

41 If the Leader of a Thread Network Partition becomes inoperable, one or more of the
42 remaining devices will determine that they are no longer connected to the Leader and form
43 new Thread Network Partitions. These Thread Network Partitions will merge and grow until,
44 if the network is still connected, only one remains.

1 **5.16.6 Children**

2 Children are members of the same Partition as their Parent. When moving to a new Thread
3 Network Partition, Parent Routers attempt to retain their original Router ID; this avoids the
4 need for their Children to reattach.
5 REEDs keep track of their routing cost to the Leader and most recent ID sequence number
6 for inclusion in Connectivity TLVs in outgoing Parent Response messages. REEDs also
7 monitor MLE Advertisement messages sent by neighboring Routers to detect any higher-
8 priority Thread Network Partition that they may connect to. If a REED hears an MLE
9 Advertisement with a higher-priority Thread Network Partition, it MUST attempt to attach to
10 that Thread Network Partition. If the REED successfully attaches to the new Thread Network
11 Partition, it must delete all information related to its previous Thread Network Partition. If
12 the REED does not become an Active Router, it starts broadcasting REED advertisements
13 after an interval of ROUTER_SELECTION_JITTER has passed. To minimize the number of
14 Thread Network Partitions, REEDs broadcast MLE Advertisements messages every
15 REED_ADVERTISEMENT_INTERVAL seconds, plus a random jitter between zero and
16 REED_ADVERTISEMENT_MAX_JITTER seconds. No Trickle timer is used. These
17 Advertisements have no effect on routing and do not contain Route64 TLVs; they contain
18 only Source Address and Leader Data TLVs.

19 **5.17 Protocol Parameters and Constants**

20 Table 5-8 lists the parameters used for Thread Network Data in Thread Networks, their
21 definitions, and their default values.

22

Table 5-8. Protocol Parameters

Parameter Name	Definition	Default
ADDRESS_QUERY_INITIAL_RETRY_DELAY	How long after an initial failed address query a device must wait before attempting a second query of the same address.	15 seconds
ADDRESS_QUERY_MAX_RETRY_DELAY	The longest required delay between retries of an address query.	8 hours
ADDRESS_QUERY_TIMEOUT	The time allowed for an address query to complete. If no response is received within this time, the address query is assumed to have failed.	3 seconds
ADVERTISEMENT_I_MAX	The I_MAX parameter for the advertisement Trickle timer.	32 seconds

Parameter Name	Definition	Default
ADVERTISEMENT_I_MIN	The I_MIN parameter for the advertisement Trickle timer.	1 second
CONTEXT_ID_REUSE_DELAY	The minimum time between disallowing the use of a 6LoWPAN context ID for compression and reusing the ID for a new prefix. This MUST be greater than the maximum propagation latency for Thread Network Data, including transmission to rx-off-when-idle devices.	48 hours
DATA_RESUBMIT_DELAY	How long a server MUST wait after a data update has been rejected by the Leader before resending the update.	300 seconds
FAILED_ROUTER_TRANSMISSIONS	The number of consecutive MCPS.DATA-Confirms having Status NO_ACK that cause a Router-to-Router link to be considered broken.	4 (up to 16 unacknowledged individual frame transmissions and retries)
ID_REUSE_DELAY	How long a Leader must wait after a Router ID has become unassigned before it can be reused.	100 seconds
ID_SEQUENCE_PERIOD	The maximum interval between increments of ID_sequence_number by the Leader.	10 seconds
INFINITE_COST_TIMEOUT	The time a Leader should wait before deallocating a Router ID that has infinite cost.	90 seconds
LEADER_TIMEOUT	The maximum expected delay between receiving a new ID sequence value in the Leader data.	120 sec
MAX_NEIGHBOR_AGE	If the time since a neighbor's most recently heard advertisement, L_age, reaches MAX_NEIGHBOR_AGE,	100 seconds

Parameter Name	Definition	Default
	the neighbor is removed from the Link Set.	
MAX_NETWORK_DATA_SIZE	The maximum size of the Thread Network Data in bytes.	254 bytes
MAX_ROUTE_COST	The maximum cost for a route.	16
MAX_ROUTER_ID	The maximum Router ID that may be assigned. The available Router IDs are [0..MAX_ROUTER_ID].	62
MAX_ROUTERS	The maximum number of Routers that a Thread Network partition may contain.	32
MIN_DOWNGRADE_NEIGHBORS	The minimum number of neighbors with link quality 2 or better that a Router must have to downgrade to a REED.	7
MIN_PREFIX_LIFETIME	The minimum valid lifetime a DHCPv6 server MUST have for a prefix in order to advertise the prefix via the Thread Network Data.	3600 seconds
MIN_STABLE_LIFETIME	The minimum expected lifetime for Thread Network Data for it to be considered stable.	168 hours
MPL_HOPLIM_DEFAULT	The default value to use for the Hop Limit field in an MPL Data Message, in case the Hop Limit value to use could not be obtained through the API call that was used to send the packet "P".	36
NETWORK_ID_TIMEOUT	If a Router goes for NETWORK_ID_TIMEOUT seconds without receiving a new ID_sequence_number from a neighbor, it MUST consider itself	120 seconds

Parameter Name	Definition	Default
	disconnected from the Leader and stop using its current Router ID.	
PARENT_ROUTE_TO_LEADER_TIMEOUT	The number of seconds a Child waits prior to reattaching in the event its Parent advertises an infinite cost to the Leader.	20 seconds
REED_ADVERTISEMENT_INTERVAL	Minimum time between sending MLE advertisement messages by a REED.	570 seconds
REED_ADVERTISEMENT_INTERVAL	The frequency of REED's sending MLE advertisement messages, adjusted also by the jitter time.	570 sec
REED_ADVERTISEMENT_MAX_JITTER	A REED waits for REED_ADVERTISEMENT_INTERVAL plus a random value between zero and REED_ADVERTISEMENT_MAX_JITTER between sending MLE advertisement messages.	60 seconds
REED_ADVERTISEMENT_MAX_JITTER	The jitter time used on REED MLE advertisement messages.	60 sec
ROUTER_DOWNGRADE_THRESHOLD	The number of active Routers on the Thread Network partition above which an active Router may decide to become a Child.	23 Routers
ROUTER_SELECTION_JITTER	Devices changing from Child to Router or vice versa first delay a random period between 0 and this number of seconds prior to finalizing the decision.	120 seconds
ROUTER_UPGRADE_THRESHOLD	The number of active Routers on the Thread Network partition below which a REED may	16 Routers

Parameter Name	Definition	Default
	decide to become a Router.	

5.18 Network Data Encoding

Network data is encoded using a TLV format for transmission between devices. Data is grouped within the encoding by prefix. TLVs are byte-aligned and transmitted serially with no intervening padding. All values in TLVs are in network byte order.

Figure 5-2 defines the Thread Network Data message format.

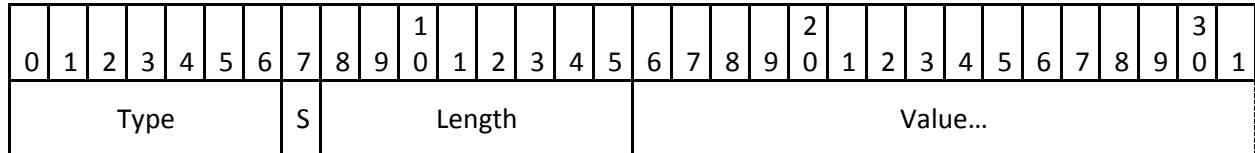


Figure 5-2. Thread Network Data Message Format

Type

A seven-bit unsigned integer that indicates the type of the value

S (Stable)

'1' if this data is expected to be valid for at least MIN_STABLE_LIFETIME; otherwise, '0'.

Length

An eight-bit unsigned integer that indicates the length of the Value field in bytes.

Value

Length bytes of value, formatted as defined for the Type.

TLVs may be used standalone or contain sub-TLVs as noted in Table 5-9. The lengths of these sub-TLVs, including their own Type and Length fields, are included in the Length field of the outer TLV. The outer TLV may contain an initial block of data specific to that type of TLV, followed by any sub-TLVs. Of the TLVs in this section, only the Prefix TLV and the Service TLV may contain sub-TLVs.

Table 5-9. TLV Usage

Name	Standalone	Prefix sub-TLV	Service sub-TLV
Has Route TLV		X	
Prefix TLV	X		
Border Router TLV		X	
6LoWPAN ID TLV		X	
Service TLV	X		
Server TLV			X
Commissioning Data TLV	X		

1 **5.18.1 Has Route TLV**

2 The Has Route TLV (Network Data TLV Type 0) contains the R_border_router_16 values for
3 which the enclosing Prefix TLV contains R_prefix. The Stable flag has the value of R_stable;
4 if there are both stable and temporary entries for this prefix, then there MUST BE two Has
5 Route TLVs, one with Stable as 0 and the other with Stable as 1.

6 Figure 5-3 defines the Has Route TLV format.

7

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3
R_border_router_16 0												Prf	Reserved												
R_border_router_16 1												Prf	Reserved												
...																									

Figure 5-3. Has Route TLV Format

R_border_router_16

Value of R_border_router_16

Prf

Associated value of R_preference

Reserved

MUST be set to '0' (zero) on transmission and ignored upon reception.

There MUST NOT be two or more Has Route TLVs with the same value of Stable within any set of sub-TLVs.

5.18.2 Prefix TLV

The Prefix TLV (Network Data TLV Type 1) contains an IPv6 prefix and any associated information. The Stable flag is set if any of the included sub-TLVs have their Stable flag set. Figure 5-4 defines the Prefix TLV format.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1																	
Domain ID								Prefix Length								Prefix...																																			
...																																																			
Sub-TLVs...																																																			

Figure 5-4. Prefix TLV Format

Domain ID

Identifier for the provisioning domain to which this TLVs information belongs.

Prefix Length

Length of the prefix in bits

Prefix

Prefix itself, ceiling(Length/8) bytes in length

1 Sub-TLVs

2 Sub-TLVs containing information about the prefix. The allowed Sub-TLV types are
3 6LoWPAN ID, Has Route, and Border Router.

4 There may be any number of Prefix TLVs in the Network Data; there MUST NOT be two or
5 more Prefix TLVs containing the same Prefix value within a single encoding of Network Data.

6 **5.18.3 Border Router TLV**

7 The Border Router TLV (Network Data TLV Type 2) contains the P_border_router_16 and
8 associated values from the Valid Prefix Set for which the enclosing Prefix TLV has the prefix
9 P_prefix. The Stable flag is set if P_stable is 'true' for all of the P_border_router values
10 included and 'false' if it is 'false' for all. If there are both stable and temporary
11 P_border_router_16 entries for one prefix, then there MUST BE two Border Router TLVs,
12 one with Stable as 0 and the other with Stable as 1. Border Router TLVs MUST be sub-TLVs
13 of a Prefix TLV.

14 Figure 5-5 defines the Border Router TLV format.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
P_border_router_16 0																Prf	P	S	D	C	R	O	N	Reserved								
P_border_router_16 1																Prf	P	S	D	C	R	O	N	Reserved								

15 **Figure 5-5. Border Router TLV Format**

16 P_border_router_16

17 Value of P_border_router_16

18 Prf

19 Value of P_preference

20 P

21 Value of P_preferred

22 S

23 Value of P_slaac

24 D

25 Value of P_dhcp

26 C

27 Value of P_configure

28 R

29 Value of P_default

30 O

31 Value of P_on_mesh

- 1 N
- 2 Value of P_nd_dns
- 3 Reserved
- 4 MUST be set to '0' (zero) on transmission and ignored upon reception.

5.18.4 6LoWPAN ID TLV

6 The 6LoWPAN ID TLV (Network Data TLV Type 3) is used to encode the entries of the
7 6LoWPAN Context ID Set. 6LoWPAN ID TLVs MUST be sub-TLVs of a Prefix TLV and encode
8 contexts for which CID_prefix is either the prefix of the outer Prefix TLV or a prefix of that
9 prefix.

10 Figure 5-6 defines the 6LoWPAN ID TLV format.

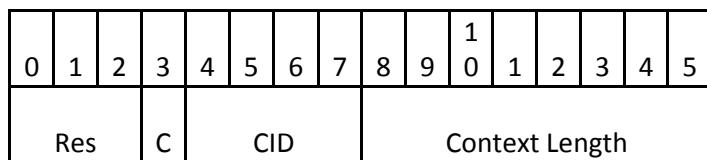


Figure 5-6. 6LoWPAN ID TLV Format

- 12 Res
- 13 Reserved bits; MUST be set to '0' (zero) on transmission and ignored upon reception.
- 14 C
- 15 Value of CID_compress
- 16 CID
- 17 Value of CID_id
- 18 Context Length
- 19 Length of the CID_prefix in bits. This MUST NOT be longer than the prefix length of the
20 outer Prefix TLV.
- 21 There may be any number of 6LoWPAN ID TLVs within the sub-TLVs of any Prefix TLV; there
22 MUST NOT be two or more 6LoWPAN ID TLVs with the same Context ID.

5.18.5 Commissioning Data TLV

24 The Commissioning Data TLV (Network Data TLV Type 4) is used to encode the
25 Commissioning Data. The length of the Commissioning Data TLV is COM_length and its
26 contents are COM_data. There may be at most one Commissioning Data TLV in the Network
27 Data.

5.18.6 Service TLV

29 The Service TLV (Network Data TLV Type 5) is used to encode entries in the Server Set,
30 with individual servers encoded as sub-TLVs using the Server TLV type. The Stable flag is
31 set if any of the included sub-TLVs have their Stable flag set. Figure 5-7 defines the Service
32 TLV format.

1

Figure 5-7. Service TLV Format

3 T

4 `1' if S_enterprise number is THREAD_ENTERPRISE_NUMBER; otherwise, `0'.

5 Res

6 Reserved bits. MUST be set to '0' (zero) on transmission and ignored upon reception.

7 S_id

8 Value of S_id

9 S_enterprise_number

10 Value of S_enterprise_number; present only if S_enterprise_number is not
11 THREAD_ENTERPRISE_NUMBER.

12 S_service_data_Length

13 Length of the S_service_data in bytes

14 S_service_data

15 Value of the S_service_data

16 There may be any number of Service TLVs in the Network Data, but there MUST NOT be two
17 or more Service TLVs with the same S_enterprise_number and S_service_data. All servers
18 with the same values for S_enterprise_number and S_service_data MUST be grouped into a
19 single Service TLV.

20 5.18.7 Server TLV

21 The Server TLV (Network Data TLV Type 6) is used to encode entries in the Server Set.
22 Figure 5-8 defines the Server TLV format.

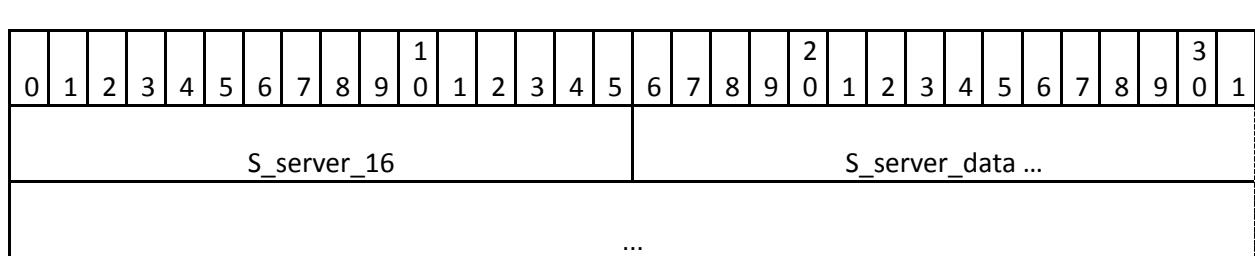


Figure 5-8. Server TLV Format

- 3 S_server_16
- 4 Value of S_server_16
- 5 S_server_data
- 6 Value of S_server_data

5.19 Network Layer TLVs

The following Network Layer TLVs are used in Address management messages within the Thread network. Values are encoded using a TLV format, where the type and length are one byte each and the length field contains the length of the value in bytes. TLVs are stored serially with no padding between them. They are byte-aligned but are not aligned in any other way such as on 2- or 4-byte boundaries. All values in TLVs are in network byte order.

Figure 5-9 defines the base TLV format.

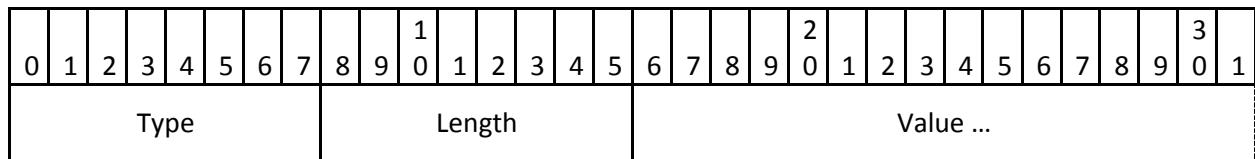


Figure 5-9. Base TLV Format

- 16 Type
 - 17 An eight-bit unsigned integer giving the type of the value, where all possible values are defined as Network Layer TLV Types in this section.
- 19 Length
 - 20 An eight-bit unsigned integer giving the length of the Value field from 0 to 254 octets.
 - 21 The value of 255 is reserved to indicate a 16-bit length following. The lengths of the Type and Length fields are not counted in the Length field.
- 23 Value
 - 24 Octets of value, formatted as defined for the Type.
 - 25 Note that Network Layer TLV Types 1, 5, and 8 are not used in this specification and are reserved.

5.19.1 Target EID TLV

2 The Target EID TLV is a Network Layer TLV Type 0 and contains an EID for which an
3 RLOC16 is requested. The Length is 16 octets. Figure 5-10 defines the Target EID TLV
4 format.

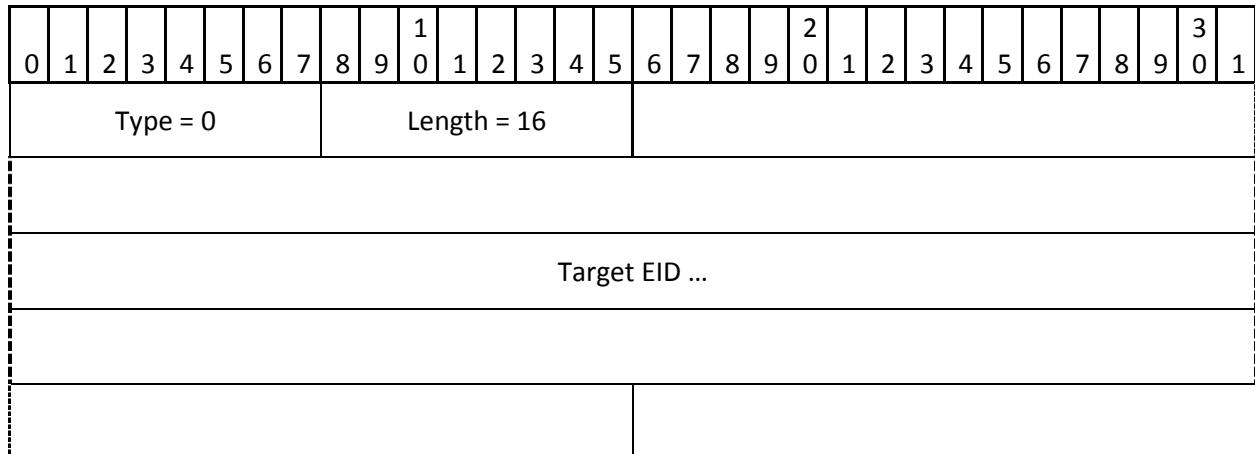


Figure 5-10. Target EID TLV Format

5.19.2 MAC Extended Address TLV

7 The MAC Extended Address TLV is Network Layer TLV Type 1 and contains the sender's MAC
8 Extended Address. This TLV is used by receiving devices to detect duplicate EIDs in the
9 network. The Length is 8 octets. The format is as follows:

10 Type = 1

11 Length = 8

12 Value = MAC Extended Address

13 The MAC Extended Address is the random MAC produced after commissioning has taken
14 place.

15 5.19.3 RLOC16 TLV

16 The RLOC16 TLV is Network Layer TLV Type 2 and contains the RLOC16 for a device. The
17 Length is 2 octets. Figure 5-11 defines the RLOC16 TLV format.

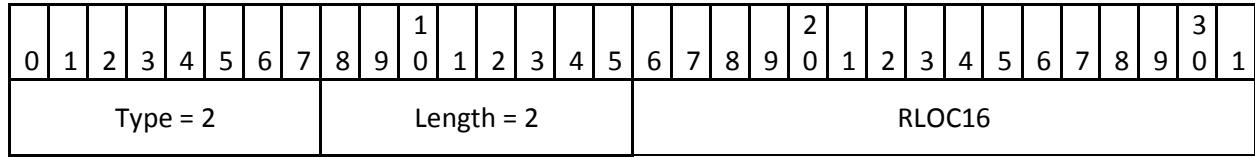
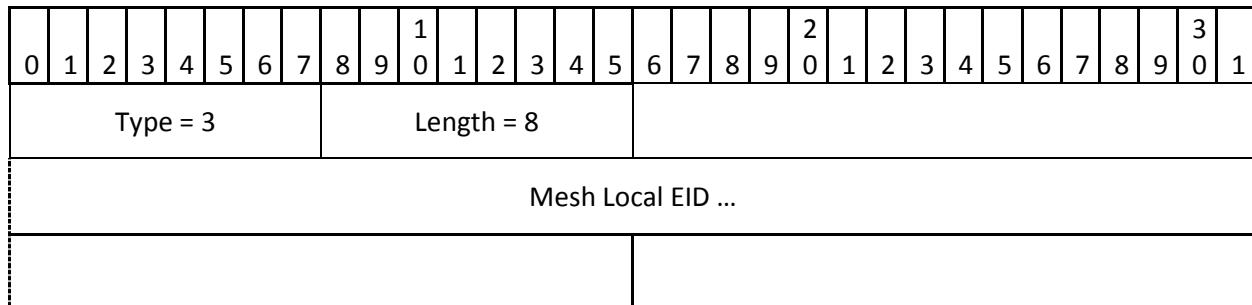


Figure 5-11. RLOC16 TLV Format

1 **5.19.4 ML-EID TLV**

- 2 The ML-EID TLV is Network Layer TLV Type 3 and contains the sender's ML-EID. This TLV is
3 used by receiving devices to detect duplicate EIDs in the network. The Length is 8 octets.
4 Figure 5-12 defines the ML-EID TLV format.



5 **Figure 5-12. ML-EID TLV Format**

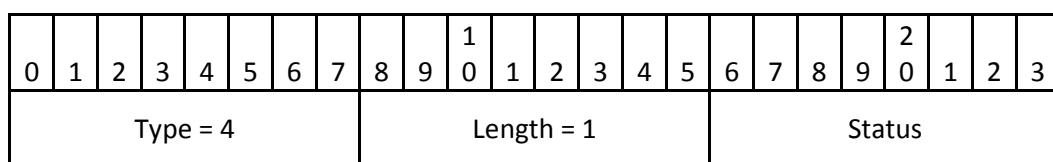
6 **5.19.5 Status TLV**

- 7 The Status TLV is Network Layer TLV Type 4 and contains 1 status byte that is an unsigned
8 integer with the Status values as follows:

0	Success
1	No Address Available
2	TOO_FEW_ROUTERS
3	HAVE_CHILD_ID_REQUEST
4	PARENT_PARTITION_CHANGE

9 The Length is 1 octet.

10 Figure 5-13 defines the Status TLV format.



11 **Figure 5-13. Status TLV Format**

12 **5.19.6 Time Since Last Transaction TLV**

- 13 The Time Since Last Transaction TLV is Network Layer TLV Type 6 and contains a 4-octet
14 unsigned value that represents the number of seconds since the device assigned to an RLOC
15 has been heard from, by means of a keep-alive message. The Length is 4 octets. Figure
16 5-14 defines the Time Since Last Transaction TLV format.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 6										Length = 4										Time Since Last Transaction...														

Figure 5-14. Time Since Last Transaction TLV Format

5.19.7 Router Mask TLV

The Router Mask TLV is Network Layer TLV Type 7. This TLV contains the same ID sequence and Assigned Router ID Mask fields as shown in the MLE Route64 TLV described in Section 5.20.1, [MLE Route64 TLV Format](#). The Length of this TLV is 9 octets. Figure 5-15 defines the Router Mask TLV format.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1																			
Type = 7										Length = 9										ID Sequence																																	

Figure 5-15. Router Mask TLV Format

5.19.8 ND Option TLV

The ND Option TLV is Network Layer TLV Type 8. This TLV contains one or more ND option types, each as a one-octet unsigned integer. The length is the number of options requested.

Figure 5-16 defines the ND Option TLV format.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 8										Length										Option Type 0 ...														

Figure 5-16. ND Option TLV Format

5.19.9 ND Data TLV

The ND Data TLV is Network Layer TLV Type 9. This TLV contains ND options formatted as described in [\[RFC 4861\]](#) Section 4.6 and related documents.

Figure 5-17 defines the ND Data TLV format.

19

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 9										Length										Option 0 ...														

Figure 5-17. ND Data TLV Format

5.19.10 Thread Network Data TLV

The Thread Network Data TLV is Network Layer TLV Type 10. This TLV contains the Thread Network Data (Section 5.13, [Thread Network Data](#)) encoded as described in Section 5.18, [Network Data Encoding](#).

Figure 5-18 defines the Thread Network Data TLV format.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 10										Length										Thread Network Data ...														

Figure 5-18. Thread Network Data TLV Format

5.20 MLE Routing TLV

The MLE Routing TLV is used for distributing router ID and routing information. This is done using the MLE Route64 TLV which is an MLE TLV in Chapter 4, Mesh Link Establishment, but it is defined in the following sub-sections.

5.20.1 MLE Route64 TLV Format

The Route64 TLV (MLE TLV Type 9) contains an ID Sequence number and a bit mask of assigned router IDs, followed by the sender's link quality and route data, encoded as one byte per assigned router ID. The length of the TLV is one plus ceiling ($\text{MAX_ROUTER_ID}/8$) plus the number of assigned router IDs. Figure 5-19 defines the MLE Route64 TLV format.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
ID Sequence										Assigned Router ID Mask...																...								
																Link Quality and Route Data...																		

Figure 5-19. MLE Route64 TLV Format

ID Sequence

The sequence number associated with the set of ID assignments in the Router ID Mask. This number is incremented whenever the set of assigned router ID changes. Nodes

1 receiving a Route64 TLV with an ID Sequence higher than ID_sequence_number MUST
2 update ID_sequence_number and ID_set to match those in the Route64 TLV.

3 Assigned Router ID Mask

4 A bit mask that encodes ID_set, the set of Router IDs that have been assigned. Bit N is
5 '1' if router ID N has been assigned. For example, if Router ID 0 and Router ID 10 are
6 assigned, the bit mask would be in hexadecimal bytes:

7 80 20 00 00 00 00 00 00

8 Link Quality and Route Data

9 One byte of data per '1' bit in the Assigned Router ID Mask.

10 The link quality and route data is encoded as one byte per Router as follows:

0	1	2	3	4	5	6	7
Out	In	Route					

11 **Figure 5-20. Link Quality and Route Data Encoding**

12 Out

13 The value of L_outgoing_quality for this Router in the Link Set, or 0 if this router does
14 not appear in the Link Set.

15 In

16 The value of L_incoming_quality for this Router in the Link Set, or 0 if this router does
17 not appear in the Link Set.

18 Route

19 Routing cost, as defined in Section 5.9.5, **Routing Cost and Next Hop**, if it is less than
20 MAX_ROUTE_COST, otherwise 0. Infinite routing cost (represented as value 0) indicates
21 that this Router is not reachable via the sender.

22 The byte corresponding to the sender's own Router ID MUST be set to '0x01'. This value
23 MUST NOT be used with any other Router ID; it encodes a route cost of 1 with a link
24 quality of 0, which is not a valid combination.

1 **CHAPTER 6 TRANSPORT LAYER**

2 **Contents**

3 6.1 UDP	6-2
4 6.2 TCP	6-3

6.1 UDP

A Thread-compliant device MUST implement the UDP (User Datagram Protocol) protocol as described in [\[RFC 768\]](#) (see Table 6-1) and [\[RFC 1122\]](#) (see Table 6-2). Thread does not elide checksums when using [\[RFC 6282\]](#).

Table 6-1. Modifications and Statements to [\[RFC 768\]](#)

Chapter	Title and Remarks / Modifications	Status
All	User Datagram Protocol	N

Table 6-2. Modifications and Statements to Chapter 4.1 of [\[RFC 1122\]](#)

Chapter / Section	Title and Remarks / Modifications	Status
4.1.3.2	Pass received IP options to application layer	N
4.1.3.2	Application layer can specify IP options in Send	O
4.1.3.2	UDP passes IP options down to IP layer	O
4.1.3.3	Pass ICMP messages up to application layer	N
4.1.3.4	Able to generate/check checksum	N
4.1.3.4	Silently discard bad checksum	N
4.1.3.4	Sender Option to not generate checksum	O
4.1.3.4	Receiver Option to require checksum	O
4.1.3.5	Pass spec-dest addr to application	N
4.1.3.5	Application layer can specify Local IP addr	N
4.1.3.5	Application layer specify wild Local IP addr	N
4.1.3.5	Application layer notified of Local IP address used	O
4.1.4	Able to specify TTL, TOS, IP options when sending datagrams	O

Chapter / Section	Title and Remarks / Modifications	Status
4.1.4	Pass received TOS up to application layer	N

1 6.2 TCP

- 2 The implementation of TCP (Transport Control Protocol) is OPTIONAL; a Thread-compliant device MAY implement TCP as described in [\[RFC 793\]](#) and [\[RFC 1122\]](#).

CHAPTER 7 SECURITY

Contents

3	7.1 Security Material Generation.....	7-3
4	7.1.1 Security Constants	7-3
5	7.1.2 Security MIB.....	7-3
6	7.1.3 Sequence Counter Maintenance	7-4
7	7.1.4 Key Generation	7-4
8	7.1.4.1 Test Vector 1	7-5
9	7.1.4.2 Test Vector 2	7-5
10	7.1.4.3 Test Vector 3	7-5
11	7.1.4.4 Example Test Program	7-6
12	7.1.5 Key Index	7-6
13	7.1.6 Key Rotation.....	7-6
14	7.1.7 Key Switching.....	7-7
15	7.2 MAC Security.....	7-8
16	7.2.1 MAC Frame Security Processing	7-8
17	7.2.1.1 Key ID Mode 0	7-8
18	7.2.1.2 Key ID Mode 1	7-8
19	7.2.1.3 Key ID Mode 2	7-9
20	7.2.2 MAC Security PIB Settings.....	7-9
21	7.2.2.1 MAC Default Key Source.....	7-10
22	7.2.2.2 MAC Key Table	7-10
23	7.2.2.3 MAC Device Table	7-13
24	7.2.2.4 MAC Security Level Table.....	7-14
25	7.2.3 MAC Auxiliary Security Header Format.....	7-15
26	7.2.3.1 Security Control Field.....	7-15
27	7.2.3.2 Frame Counter Field.....	7-15
28	7.2.3.3 Key Identifier Field	7-15
29	7.3 MLE Security	7-16
30	7.3.1 MLE Message Security Processing	7-16
31	7.3.1.1 Key ID Mode 1	7-16
32	7.3.1.2 Key ID Mode 2	7-18
33	7.3.2 MLE Security MIB Settings.....	7-19
34	7.3.2.1 MLE Shared Frame Counter.....	7-20
35	7.3.2.2 MLE Frame Counter	7-20
36	7.3.2.3 MLE Key Table.....	7-20
37	7.3.2.4 MLE Device Table	7-22
38	7.3.3 MLE Auxiliary Security Header Format.....	7-23
39	7.3.3.1 Security Control Field.....	7-23

1	7.3.3.2	Frame Counter Field.....	7-23
2	7.3.3.3	Key Identifier Field	7-24
3	7.3.4	MLE Frame Counter Processing	7-24
4	7.3.4.1	Frame Counter Values.....	7-24
5	7.3.4.2	MLE Message Processing.....	7-24
6	7.3.4.3	MLE Message Processing Example.....	7-25
7	7.4	DTLS	7-26
8	7.4.1	Conventions	7-26
9	7.4.1.1	7.4.1.1 Elliptic Curve Points	7-26
10	7.4.1.2	Integers	7-27
11	7.4.1.3	Octet Strings.....	7-27
12	7.4.2	Integer and Octet String Conversions	7-27
13	7.4.2.1	Integer to Octet String Conversion.....	7-27
14	7.4.2.2	Octet String to Integer Conversion.....	7-27
15	7.4.3	Handshake	7-28
16	7.4.4	Failure Processing	7-28
17	7.4.5	Retransmission Processing	7-28
18	7.4.5.1	Retransmission example.....	7-29
19	7.4.6	Random Number Generation.....	7-29
20	7.4.7	Elliptic Curve J-PAKE Extensions.....	7-29
21	7.4.7.1	Existing Definitions	7-29
22	7.4.7.2	Additional Definitions	7-30
23	7.4.7.3	ClientHello and ServerHello Extensions.....	7-31
24	7.4.7.4	Shared Secret Conversion.....	7-32
25	7.4.7.5	ServerKeyExchange and ClientKeyExchange Enumeration	7-33
26	7.4.7.6	ServerKeyExchange	7-33
27	7.4.7.7	ClientKeyExchange	7-35
28	7.4.7.8	Server Secret Key Generation	7-36
29	7.4.7.9	Client Secret Key Generation	7-37
30	7.4.7.10	AEAD Record Protection.....	7-37
31	7.4.8	Test Program for Hash Generation.....	7-38

1 **7.1 Security Material Generation**

2 **7.1.1 Security Constants**

3 Table 7-1 summarizes the Thread Security Constants.

4 **Table 7-1. Security Constants**

Constant	Value	Comment
<i>THR_DTLS_MAX_RETRANSMIT_COUNTER</i>	2	The maximum number of times DTLS retransmission occurs.
<i>THR_DTLS_INIT_RETRANSMIT_TIMEOUT</i>	8	The initial value for the DTLS retransmit timer in seconds.
<i>THR_SINGLE_USE_KEY_PERSISTENCE_TIME</i>	20	The time in seconds a single use key persists for. It MUST NOT persist beyond this time.

5 **7.1.2 Security MIB**

6 Table 7-2 summarizes the Thread Security MIB (Management Information Base).

7 **Table 7-2. Security MIB**

MIB attribute	Type	Default	Comment
<i>thrMasterKey</i>	Key	-	Key used to derive security material for MAC and MLE protection.
<i>thrKeyRotation</i>	uint32	672	Key rotation period in hours. The minimum key rotation period SHALL be 1 hour and a key rotation period of 0 SHALL NOT be allowed.
<i>thrKeySwitchGuardTime</i>	uint32	624	Key switching guard time in hours to prevent inadvertent key switching.
<i>thrKeySwitchGuardTimer</i>	uint32	0	Key switching guard timer to prevent inadvertent key switching. It SHALL be decremented every hour until it reaches 0. Initial default value of 0 allows key switching to occur.

MIB attribute	Type	Default	Comment
<i>thrKeySequenceCounter</i>	uint32	0	Monotonically increasing counter used for key rotation.

7.1.3 Sequence Counter Maintenance

A node maintains its own sequence counter, *thrKeySequenceCounter*, for the purposes of key rotation as shown in Figure 7-1. *thrKeySequenceCounter* is kept in synchronization with neighboring nodes through the use of the Key Index fields in a Media Access Control (MAC) frame's auxiliary security header and the use of Key Source and Key Index fields in a Mesh Link Establishment (MLE) message's auxiliary security header.

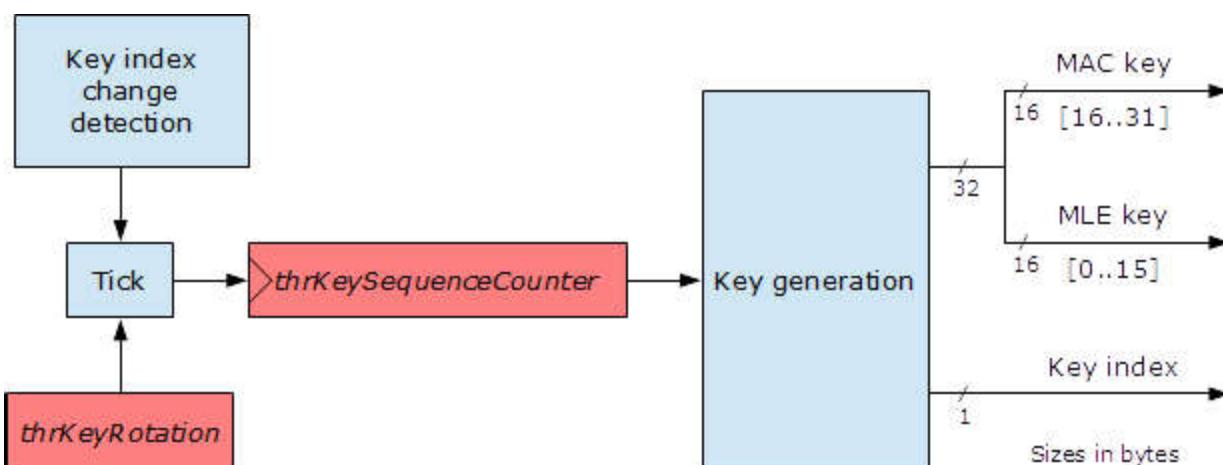


Figure 7-1. Sequence Counter Maintenance

7.1.4 Key Generation

10 Each Thread node receives the Master Key when joining and assigns it to the *thrMasterKey*
11 attribute. *thrMasterKey* is used in conjunction with a sequence counter to derive two
12 separate keys for use by the MAC sublayer and MLE. As protection may potentially be
13 applied at both MAC and MLE layers in a packet, a different key MUST be used at each
14 layer. The use of Hashed Message Authentication Mode with the SHA-256 algorithm (HMAC-
15 SHA256) as the keyed hash function produces an output of 32 bytes [\[RFC 6234\]](#). Therefore,
16 this is sufficient for the two separate keys required for the MAC sublayer and MLE.

17 The 32-byte combined key is generated as follows:

18 HMAC-SHA256(thrMasterKey, sequence counter || "Thread")

19 where `||` is the concatenation operator and “Thread” is the byte sequence:

20 54 68 72 65 61 64

21 The MAC key is the upper 16 bytes of the resulting key and the MLE key is the lower 16
22 bytes of the resulting key as shown in Figure 7-2.

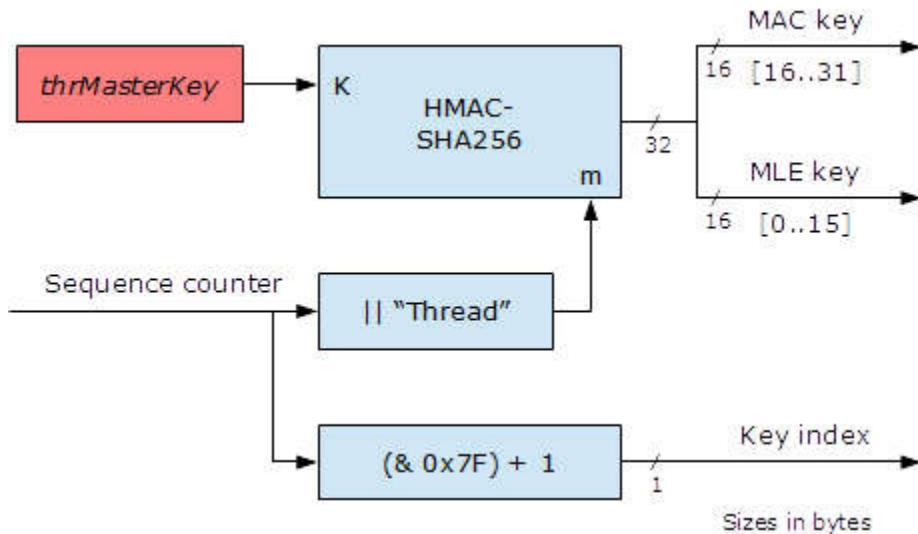


Figure 7-2. Key Generation

7.1.4.1 Test Vector 1

7.1.4.1.1 Inputs

thrMasterKey = 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff

Sequence counter = 00 00 00 00

Concatenation: 00 00 00 00 54 68 72 65 61 64

7.1.4.1.2 Outputs

MLE key = 54 45 f4 15 8f d7 59 12 17 58 09 f8 b5 7a 66 a4

MAC key = de 89 c5 3a f3 82 b4 21 e0 fd e5 a9 ba e3 be f0

Key index = 1

7.1.4.2 Test Vector 2

7.1.4.2.1 Inputs

thrMasterKey = 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff

Sequence counter = 00 00 00 01

Concatenation: 00 00 00 01 54 68 72 65 61 64

7.1.4.2.2 Outputs

MLE key = 8f 4c d1 a2 7d 95 c0 7d 12 db 89 74 bd 61 5c 13

MAC key = 9b e0 d1 af 7b d8 73 50 de ab cd d0 7f eb b9 d5

Key index = 2

7.1.4.3 Test Vector 3

7.1.4.3.1 Inputs

thrMasterKey = 00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff

Sequence counter = 00 00 00 02

Concatenation: 00 00 00 02 54 68 72 65 61 64

1 **7.1.4.3.2 Outputs**

2 MLE key = 01 6e 2a b8 ec 88 87 96 87 a7 2e 0a 35 7e cf 2a
3 MAC key = 56 41 09 e9 d2 aa d7 f7 23 ec 3b 96 11 0e ef a3
4 Key index = 3

5 **7.1.4.4 Example Test Program**

6 The following is a test program written in Python which generates the key values above

```
7 import hmac
8 import hashlib
9 import binascii
10
11 key_thread = '00112233445566778899aabbccddeeff'
12
13 print "\nThread test vectors"
14 print "-----"
15
16 print "\nKey: %s" % key_thread
17 k = binascii.unhexlify(key_thread)
18
19 for counter in range(3):
20     counter_str = '%08x' % counter
21     print "\nSequence counter: " + counter_str
22     print "Data: %s" % counter_str + binascii.hexlify('Thread')
23     s = binascii.unhexlify(counter_str) + 'Thread'
24     d = binascii.hexlify(hmac.new(k, s, digestmod=hashlib.sha256).digest())
25     print "\nMLE key: %s" % d[:32]
26     print "MAC key: %s" % d[32:]
27     print "Key index: %d" % ((counter & 0x7f) + 1)
```

28 **7.1.5 Key Index**

29 The outgoing key index is 1 plus the value of the bottom 7 bits of *thrKeySequenceCounter*,
30 preventing the disallowed use of key index value 0. This allows a device to identify relatively
31 easily which key in a sequence of key rotations is currently being used. Typically, three keys
32 are stored:

- 33 1. Key indexed by the next key index (calculated pending key change)
- 34 2. Key indexed by the current key index (storage required)
- 35 3. Key indexed by the previous key index (storage required)

36 This storage of multiple keys allows migration to a new key while still retaining the old key.

37 **7.1.6 Key Rotation**

38 *thrKeySequenceCounter* rotation and key switching SHALL happen simultaneously. The keys
39 will rotate as shown in Figure 7-3.

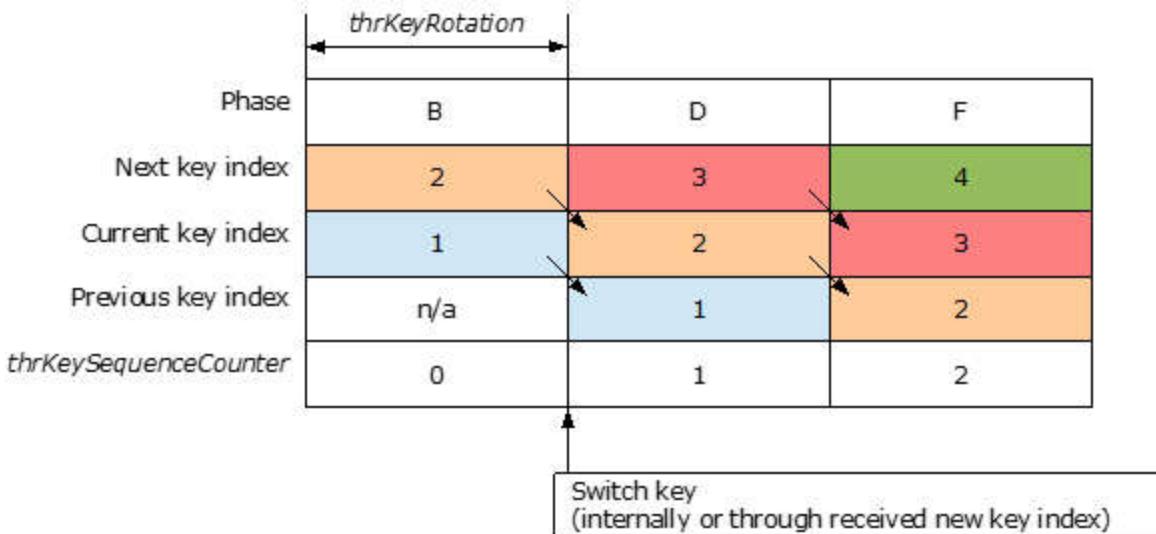


Figure 7-3. Key Rotation

3 Next key generation takes place at the same time as key switching. It is NOT
4 RECOMMENDED to offset next key generation from key switching. The key indexed by the
5 previous key index MAY be retired prior to the next key switch.

7.1.7 Key Switching

7 The definition of key switching is moving to the next key index. This happens when one of
8 the following use cases occurs:

- 9 1. *thrKeyRotation* expires.
- 10 2. *thrKeySwitchGuardTimer* is 0 and the node receives and successfully processes a MAC
11 frame or MLE message whose incoming key index matches the next key index.

12 **Note: If *thrKeySwitchGuardTimer* is not 0, the node SHALL process the incoming
13 frame or message but SHALL NOT update *thrKeySequenceCounter* and rotate the
14 key.**

15 In case (1), the node SHALL increment *thrKeySequenceCounter* by 1 automatically. In case
16 (2), the node SHALL update *thrKeySequenceCounter* to match the incoming key index, thus
17 causing the node's own key index to update. In both cases, the node SHALL rotate the key
18 as described in Section 7.1.6, **Key Rotation**.

19 When key switching occurs, the node SHALL reset its outgoing frame counter for MAC and
20 MLE to 0. Incoming frame counters for each node are stored per key, so effectively a new
21 DeviceDescriptor is created, replacing the old DeviceDescriptor. The node SHALL set
22 *thrKeySwitchGuardTimer* to *thrKeySwitchGuardTime*; *thrKeySwitchGuardTimer* SHALL
23 subsequently be decremented every hour until it reaches 0. The node MUST prepare
24 security material for the anticipated next key index. This SHOULD occur at the point of key
25 switching.

26 On node reset, *thrKeySwitchGuardTimer* and *thrKeyRotation* SHALL be initialized to their
27 default values (see Table 7-2).

7.2 MAC Security

This section describes MAC frame security processing, MAC security PIB settings, and the MAC auxiliary security header format used in MAC frames.

Note: In this section, *tKSC* is used as an abbreviation for *thrKeySequenceCounter* and "Key ID Mode" is used as an abbreviation for "Key Identifier Mode".

7.2.1 MAC Frame Security Processing

Secured MAC data frames SHALL use only:

- Key ID Mode 0
- Key ID Mode 1
- Key ID Mode 2

Secured MAC command frames SHALL use only:

- Key ID Mode 1

7.2.1.1 Key ID Mode 0

When a node receives a valid secured MAC data frame with Key ID Mode field set to '0', this indicates the use of a single-use pairwise key for Joiner Entrust request and response. The incoming frame security procedure looks up the key based on the source address of the MAC data frame. As the key is single use, the outgoing and incoming frame counters are both set to 0 and no frame counter synchronization is required.

This mode is used in conjunction with a single-use pairwise key only, which is set during the joining process on receipt of a KEK TLV contained in an RLY_TX.ntf message. The single-use pairwise key is used to secure and verify the outgoing JOIN_ENT.ntf and the corresponding dummy response. A single-use pairwise key SHALL be stored for no longer than THR_SINGLE_USE_KEY_PERSISTENCE_TIME.

The following messages MAY use Key ID Mode 0:

- JOIN_ENT.ntf
- JOIN_ENT.ntf dummy response

7.2.1.1.1 JOIN_ENT.req Processing

If the single-use key and associated security material process the MAC data frame successfully, the single-use key and associated security material are discarded after sending the corresponding dummy response message.

7.2.1.1.2 JOIN_ENT.rsp Processing

If the single-use key and associated security material process the MAC data frame successfully, the single-use key and associated security material are discarded.

7.2.1.2 Key ID Mode 1

When a node receives a valid secured MAC frame with Key ID Mode field set to '1', there are two potential outcomes with regard to looking up the key index:

1. The incoming key index in the frame matches a stored key index (key index match).
2. The incoming key index in the frame does not match a stored key index (key index mismatch).

1 **7.2.1.2.1 Key Index Match**

2 If the incoming key index matches, the key can be directly retrieved. If the incoming key
3 index is higher than the current key index and *thrKeySwitchGuardTimer* is 0, the recipient
4 MUST switch to the new key index.

5 **7.2.1.2.2 Key Index Mismatch**

6 If the incoming key index does not match, the MAC frame SHALL be silently discarded in
7 accordance with the rules in Section 7.5.8.2.3 of [\[IEEE802154\]](#).

8 **7.2.1.3 Key ID Mode 2**

9 Thread nodes MAY use the MAC Key ID Mode 2, a Key Source set to 0xffffffff and the Key
10 Index set to 0xff for messages sent to the broadcast PAN ID.

11 For messages sent with these parameters, the use of MAC Security is intended only to
12 provide cursory filtering for non-Thread Networks based on mismatching incoming security
13 processing.

14 The Key used for this mode is pre-defined and openly specified and the use of MAC Security
15 is used for filtering only. As such, it does not provide any useful MAC layer encryption or
16 protection characteristics.

17 When a Thread node receives a valid secured MAC data frame with Key ID Mode field set to
18 '2', the Key Source subfield set to 0xffffffff, and the Key Index set to 0xff, this indicates the
19 use of this specific static key material.

20 Given that this use lacks any true security characteristics or replay protection, Thread nodes
21 MUST enforce specific processing of such messages as if they are unsecured at the MAC
22 layer. This enforcement MUST provide protection against Denial of Service (DoS) attacks as
23 when receiving completely unsecured messages.

24 The incoming frame security procedure MUST look up the Key Descriptor based on matching
25 on all the values: Key Mode = 2, Key Source = 0xffffffff, and Key Index = 0xff.

26 The Device Descriptor Handle in the Key Device Descriptor of the MAC Key ID Mode 2
27 Descriptor references a specific Device Descriptor used for this Mode.

28 The Frame Counter content of the Key ID Mode 2 Device Descriptor MUST be reset to
29 0x00000000 after each successful processing of an incoming Key Mode = 2,
30 Key Source = 0xffffffff, and Key Index = 0xff message.

31 For both outgoing and incoming processing, the nonce value for the Key ID Mode 2 frames
32 input to the CCM* algorithm MUST be set to the *ExtAddress* value of the Key ID Mode 2
33 Device Descriptor instead of the value of *aExtendedAddress* of the originating node.

34 **7.2.2 MAC Security PIB Settings**

35 Thread security material SHALL be used to create a KeyDescriptor entry in the MAC Key
36 Table. The MAC Key Table SHALL be updated when key switching occurs (see Section 7.1.7,
37 **Key Switching**). The MAC Key Table SHALL be maintained in a FIFO order where the latest
38 update displaces the oldest MAC Key Table entry.

39 Each Thread node SHALL maintain a MAC Key Table entry corresponding to the current
40 active MAC key.

1 Note: The following tables describing attributes are conceptual only. A particular
2 implementation MAY represent the tables in whatever form is appropriate and MAY optimize
3 in any way, but the node behavior MUST match that of the conceptual model.

4 **7.2.2.1 MAC Default Key Source**

5 A participating Thread node SHALL have the set defined in Table 7-3.

6 **Table 7-3. MAC Default Key Source**

PIB Attribute	Value	Comment
<i>macDefaultKeySource</i>	0xff00000000000000	Arbitrary value indicating one global key used for MAC protection. Note there is no need to store any identifier of the originator of <i>thrMasterKey</i> .

7 Key ID Mode 1 SHALL be used in conjunction with *macDefaultKeySource*. Where a global
8 MAC key is used in conjunction with a key index, this implies that the lookup data reduces
9 to the key index only.

10 **7.2.2.2 MAC Key Table**

11 A Thread node SHALL have the set defined in Table 7-4.

12 **Table 7-4. MAC Key Table**

PIB Attribute	Value	Comment
<i>macKeyTable</i>	KeyDescriptor entries	One entry each for: <ul style="list-style-type: none">• Next MAC Key• Current MAC Key• Previous MAC Key One entry for the Key ID Mode 2 Key One entry for each KEK for joining device
<i>macKeyTableEntries</i>	$4 + n$	n is the maximum number of simultaneous joining devices

13 **7.2.2.2.1 Key Descriptor**

14 A Thread node SHALL have the KeyDescriptor entry set defined in Table 7-5 for each MAC
15 Key.

1

Table 7-5. Key Descriptor

KeyDescriptor Attribute	Value	Comment
KeyIdLookupList	One KeyIdLookupList entry	One entry for this MAC Key
KeyIdLookupListEntries	1	One entry for this MAC Key
KeyDeviceList	KeyDeviceList entries	Entries in the MAC device table
KeyDeviceListEntries	(variable)	Multiple entries for this MAC Key
KeyUsageList	Two KeyUsageList entries	One key usage for MAC data frames One key usage for MAC data request command frames
KeyUsageListEntries	2	Two entries for this MAC Key
Key	(variable)	The MAC key value For the MAC Key ID Mode 2, Key Index =0xff, and Key Source = 0xffffffff, Key Descriptor, the value of the key is always 78 58 16 86 fd b4 58 0f b0 92 54 6a ec bd 15 66

2 **7.2.2.2.2 Key ID Lookup for Key ID Mode 0**

3 The single KeyIdLookupList for a KEK for a joining device SHALL have the set defined in
4 Table 7-6.

5 **Table 7-6. Key ID Lookup for Key ID Mode 0**

KeyIdLookupDescriptor Attribute	Value	Comment
LookupData	Counterpart MAC address 0x00	There is no key index in this case because it is a pairwise key.
LookupAxisSize	9	Size is 9 bytes.

6 **7.2.2.2.3 Key ID Lookup for Key ID Mode 1**

7 The single KeyIdLookupList SHALL have the set defined in Table 7-7.

1

Table 7-7. Key ID Lookup for Key ID Mode 1

KeyIdLookupDescriptor Attribute	Value	Comment
LookupData	<i>macDefaultKeySource</i> KeyIndex	KeyIndex is the key index associated with this MAC Key, which is the bottom 7 bits of <i>tKSC</i> plus 1. Only KeyIndex needs to be stored as <i>macDefaultKeySource</i> is a constant
LookupDataSize	9	Size 9 bytes

2 **7.2.2.2.4 Key ID Lookup for Key ID Mode 2**

3 The single KeyIdLookupList SHALL have the set defined in Table 7-7.

4 **Table 7-8. Key ID Lookup for Key ID Mode 2**

KeyIdLookupDescriptor Attribute	Value	Comment
LookupData	0xffffffff 0xff	These lookup values allow specific processing of Key ID Mode 2 messages.
LookupDataSize	5	Size 5 bytes

5 **7.2.2.2.5 Key Device List**

6 Each KeyDeviceList entry SHALL have the set defined in Table 7-8.

7 **Table 7-8. Key Device List**

KeyDeviceDescriptor Attribute	Value	Comment
DeviceDescriptorHandle	Implementation specific	Indexes the appropriate DeviceDescriptor.
UniqueDevice	0	The key is not unique per Thread node.
Blacklisted	(variable)	Initially set to 'FALSE'.

1 **7.2.2.2.6 Key Usage List**

2 The two KeyUsageList entries SHALL have the sets defined in Table 7-9 and Table 7-10.

3 **Table 7-9. Key Usage List for MAC Data Frame**

KeyUsageDescriptor Attribute	Value	Comment
FrameType	0x02	MAC data frame. Note this is a constant and does not need to be stored

4 **Table 7-10. Key Usage List for MAC Data Request Command Frame**

KeyUsageDescriptor Attribute	Value	Comment
FrameType	0x03	MAC command frame. Note this is a constant and does not need to be stored
CommandFrameIdentifier	0x04	Data request command frame. Note this is a constant and does not need to be stored

6 **7.2.2.3 MAC Device Table**

7 A Thread node SHALL have the set defined in Table 7-11.

8 **Table 7-11. MAC Device Table**

PIB Attribute	Value	Comment
<i>macDeviceTable</i>	DeviceDescriptor entries	One entry for each neighbor the Thread node is in communication with One entry for the Key ID Mode 2 Device Descriptor
<i>macDeviceTableEntries</i>	(variable)	Multiple entries

9 **7.2.2.3.1 Device Descriptor**

10 A Thread node SHALL have the DeviceDescriptor entry set defined in Table 7-12 for each
11 MAC Key.

1

Table 7-12. Device Descriptor Entry Set

DeviceDescriptor Attribute	Value	Comment
PANId	2 bytes	The PAN ID of the neighbor node. Note that this data can be implied and no storage is needed because the neighbor node will have the same PAN ID as this node. Set to 0xffff for the Key ID Mode 2 Device Descriptor.
ShortAddress	2 bytes	The short address allocated to the neighbor node. Set to Key ID Mode 2 for the Discovery Device Descriptor.
ExtAddress	8 bytes	The IEEE address of the neighbor node Set to 0x3506feb823d48712 for the Key ID Mode 2 Device Descriptor.
FrameCounter	4 bytes	The incoming frame counter corresponding to the most recently received frame from the neighbor node. Used for replay protection by checking that a subsequent incoming frame counter is greater than this value. For the Key ID Mode 2 Device Descriptor, the FrameCounter is initially set to 0x00000000 and then reset to 0x00000000 after each successful Key ID Mode = 2; KeySource = 0xffffffff, Key Index =0xff message processing.
Exempt	FALSE	Irrelevant because there is no security policy in place at the MAC layer. Note: This is a constant and does not need to be stored.

2 A Thread node MUST maintain a separate DeviceDescriptor list consisting of all its neighbors
3 for each of its KeyDescriptors, thus ensuring a separate incoming frame counter per
4 neighbor per Key. This implies that each Key is valid to be used with any of the neighbor
5 nodes.

6 **7.2.2.4 MAC Security Level Table**

7 There is no security policy at the MAC layer, therefore all Thread nodes SHALL have the set
8 defined in Table 7-13.

1

Table 7-13. MAC Security Level Table

PIB Attribute	Value	Comment
<i>macSecurityLevelTable</i>	Empty	No security policy at MAC layer

2 Security policy enforcement takes place at the next higher layer.

7.2.3 MAC Auxiliary Security Header Format

4 The MAC frame Auxiliary Security Header (see Section 7.6.2 of [\[IEEE802154\]](#)) is used when
5 a MAC frame is secured to provide additional data required for security.

7.2.3.1 Security Control Field

7 The Security Control field SHALL have the values defined in Table 7-14.

Table 7-14. Security Control Field

Field	Value	Comment
Security Level	0x05	ENC-MIC-32 is the default value for Thread MAC security.
Key Identifier Mode	0x00 0x01 0x02	Mode 0: Key is implicitly determined from the source address field. Mode 1: Key is determined from the 1-byte Key Index field of the Key Identifier field of the auxiliary security header in conjunction with <i>macDefaultKeySource</i> . Mode 2: Key is determined from the 4-byte Key Source and the 1-byte Key Index field of the Key Identifier field of the auxiliary security header.

7.2.3.2 Frame Counter Field

10 The Frame Counter field SHALL assume the value of the *macFrameCounter* PIB attribute.

7.2.3.3 Key Identifier Field

12 The Key Identifier field comprises the Key Index subfield and optional Key Source subfield.
13 The Key Source subfield SHALL only be present when Key ID Mode is 0x02.

7.2.3.3.1 Key Index Subfield

15 For Key ID Mode 1, the Key Index subfield SHALL be the key index associated with the
16 active MAC Key.

17 For Key ID Mode 2, the Key Index subfield SHALL be set to 0xff.

1 **7.2.3.3.2 Key Source Subfield**

2 The Key Source subfield used for Key ID Mode 2 SHALL be set to 0xffffffff.

3 **7.3 MLE Security**

4 This section describes MLE message security processing, MLE security MIB settings and the
5 MLE auxiliary security header format used in MLE messages.

6 **Note: In this section, *tKSC* is used as an abbreviation for *thrKeySequenceCounter*
7 and “Key ID Mode” is used as an abbreviation for “Key Identifier Mode”.**

8 **7.3.1 MLE Message Security Processing**

9 Secured MLE messages SHALL use only Key ID Mode 2. All MLE messages except Discovery
10 Request and Discovery Response SHALL be secured. MLE Discovery Request and Discovery
11 Response messages SHALL NOT be secured.

12 **7.3.1.1 Key ID Mode 1**

13 **The use of MLE Key ID Mode 1 is deprecated. This section should be ignored.**

14 When a node receives a valid secured MLE message with Key ID Mode field set to ‘1’, there
15 are two potential outcomes with regard to looking up the key index:

- 16 1. The incoming key index in the frame matches a stored key index (key index match); see
17 Section 7.3.1.1.1, **Key Index Match**.
- 18 2. The incoming key index in the frame does not match a stored key index (key index
19 mismatch).

20 **7.3.1.1.1 Key Index Match**

21 If the incoming key index matches, the key can be directly retrieved. If the incoming key
22 index is higher than the current key index and *thrKeySwitchGuardTimer* is 0, the recipient
23 MUST switch to the new key index.

24 **7.3.1.1.2 Key Index Mismatch**

25 If the incoming key index does not match, there are two possible outcomes:

- 26 1. The incoming key index represents the originator’s effective *tKSC* within 128 of the
27 recipient’s *tKSC*.
- 28 2. The incoming key index represents the originator’s effective *tKSC* not within 128 of the
29 recipient’s *tKSC*.

30 **7.3.1.1.2.1 Key Index within 128**

31 If the key index represents the originator’s effective *tKSC* within 128 of the recipient’s *tKSC*,
32 it should be possible to reconstruct the key and associated keying material and process the
33 MLE message.

34 The incoming key index may be lower or higher in value than a stored key index. Due to
35 key rotation and wrapping past the 128 value, it is possible that:

- 36 • A lower incoming key index represents a higher effective *tKSC* than the recipient’s
37 *tKSC*

- 1 • A higher incoming key index represents a lower effective *tKSC* than the recipient's
2 *tKSC*

3 Table 7-15 provides examples on how the originator's effective *tKSC* could be reconstructed
4 depending on whether the incoming key index represents a lower or higher counter.

5 **Table 7-15. Example of Key Index Mismatch**

Incoming key index	Recipient's current key index	Originator's effective <i>tKSC</i>	Recipient's <i>tKSC</i>	Assumption
43	32	1578	1567	Incoming key index represents higher effective <i>tKSC</i>
12	68	1675	1603	Incoming key index represents higher effective <i>tKSC</i>
43	32	1450	1567	Incoming key index represents lower effective <i>tKSC</i>
12	68	1547	1603	Incoming key index represents lower effective <i>tKSC</i>

6 The initial assumption is that the incoming key index represents a higher effective *tKSC* on
7 the originator. The new higher effective *tKSC* value is calculated and passed to the key
8 generation procedure outlined in Section 7.1.3, **Sequence Counter Maintenance** to
9 produce a transient key and associated security material. The incoming and outgoing frame
10 counters in the associated security material SHALL be set to 0.

11 If the security processing of the MLE message using the transient key and associated
12 security material is successful, then the MLE message is processed further according to the
13 rules described in Section 7.3.4, **MLE Frame Counter Processing**. The transient key and
14 associated security material MAY be stored and subsequently used as the current key and
15 associated security material.

16 If the security processing of the MLE message using the transient key is unsuccessful, then
17 the secondary assumption is that the incoming key index represents a lower effective *tKSC*
18 on the originator. The new lower effective *tKSC* is calculated and passed to the key
19 generation procedure outlined in Section 7.1.3, **Sequence Counter Maintenance** to
20 produce a transient key and associated security material. The incoming and outgoing frame
21 counters in the associated security material SHALL be set to 0.

22 If the security processing of the MLE message using the transient key is successful, then the
23 MLE message is partially processed further according to the rules described in Section 7.3.4,
24 **MLE Frame Counter Processing**. The transient key and associated security material
25 SHALL be subsequently discarded, as they represent a key that has already been used by
26 the recipient, therefore cannot be used again by the recipient.

27 **7.3.1.1.2.2 Key Index not within 128**

28 If the transient key still fails to process the frame successfully, it can be assumed that either
29 the originator's effective *tKSC* is not within 128 of the recipient's *tKSC* or *thrMasterKey* has
30 been updated. In either case, the node MUST attempt re-attachment using the procedure

1 described in Section 4.7.7.3, Router Synchronization after Reset, in Chapter 4, Mesh Link
2 Establishment. This will force the use of MLE messages using Key ID Mode 2.
3 During re-attachment, the node will receive the full effective *tKSC* from the Parent node and
4 thus be able to resynchronize if *thrMasterKey* has not been updated. If *thrMasterKey* has
5 been updated, re-attachment will fail, and the node MUST re-commission itself onto the
6 Thread Network.

7 7.3.1.2 Key ID Mode 2

8 When a node receives a valid secured MLE message with Key ID Mode field set to '2', the
9 incoming key index SHALL be used to determine whether there is a stored key index. There
10 are two potential outcomes:

- 11 1. The incoming key index in the frame matches a stored key index (key index match); see
12 Section 7.3.1.1.1, **Key Index Match**.
- 13 2. The incoming key index in the frame does not match a stored key index (key index
14 mismatch).

15 7.3.1.2.1 Key Index Mismatch

16 If the incoming key index does not match, the originator's *tKSC* is carried in the key source
17 field. This counter value is extracted and is passed to the key generation procedure
18 described in Section 7.1.3, **Sequence Counter Maintenance** to create a transient key and
19 associated security material. The incoming and outgoing frame counters in the associated
20 security material SHALL be set to 0. The MLE message type is then taken into consideration
21 with regard to subsequent processing, *tKSC* updating and transient key and associated
22 security material storage. The MLE messages are broadly categorized into three groups:

- 23 • Tentative
- 24 • Authoritative
- 25 • Peer

26 7.3.1.2.1.1 Tentative MLE Messages

27 An MLE message classed as tentative is used in the case where the originator of the MLE
28 message may not be sure that its *tKSC* is in synchronization with the rest of the Thread
29 network and is thus attempting to synchronize with the Thread network. The originator
30 therefore uses Key ID Mode 2 to ensure the recipient can process the MLE message,
31 assuming the same value of *thrMasterKey*. The recipient, however, will not store the
32 transient key or associated security material produced to process the MLE message.

33 Tentative MLE messages are:

- 34 • MLE Link Request
- 35 • MLE Link Reject
- 36 • MLE Parent Request
- 37 • MLE Child Update Request

38 If the security processing of the MLE message using the transient key and associated
39 security material is successful, then the MLE message is partially processed further in
40 accordance with the rules for MLE message processing detailed in Section 7.3.4, **MLE**
41 **Frame Counter Processing**. The transient key and associated security material are
42 unconditionally discarded after the processing of the MLE message.

7.3.1.2.1.2 Authoritative MLE Messages

An MLE message classed as authoritative is used in the case where the originator of the MLE message is confident that its *tKSC* is in synchronization with the rest of the Thread network and is thus providing synchronization to the recipient. The originator therefore uses Key ID Mode 2 to ensure the recipient can process the MLE message, assuming the same value of *thrMasterKey*. The recipient will typically store the resulting transient key and associated security material produced to process the MLE message.

Authoritative MLE messages are:

- MLE Link Accept
- MLE Parent Response
- MLE Child ID Request
- MLE Child Update Response

If the security processing of the MLE message using the transient key and associated security material is successful, then the MLE message is processed further in accordance with the rules for MLE message processing detailed in Section 7.3.4, **MLE Frame Counter Processing**. If the subsequent processing of the MLE message is successful and the incoming Key Source field is greater than the recipient's *tKSC*, the recipient's *tKSC* SHALL be updated to the incoming Key Source field and the transient key and associated security material SHALL be stored and subsequently used as the current key and associated security material.

7.3.1.2.1.3 Peer MLE Messages

An MLE message classed as peer is used in the case where the originator of the MLE message is confident that its *tKSC* is in synchronization with the rest of the Thread network and where it is expected that the recipient is also confident that its *tKSC* is in synchronization with the rest of the Thread network. In this case, it is expected that the incoming key index will match, i.e. be the current or next stored key index, however the originator uses Key ID Mode 2 to ensure the recipient can process the MLE message in the case where there may be a key index mismatch between the two Thread devices.

Peer MLE messages are:

- MLE Link Advertisement

An MLE Advertisement is processed as an authoritative MLE message as detailed in Section 7.3.1.2.1.2, **Authoritative MLE Messages**.

If there is a key mismatch, the recipient should be aware of its current state with respect to the originator to determine if it should process the incoming MLE Advertisement. If the recipient was previously synchronized to the originator and expecting only a key switch, it SHOULD NOT process the MLE Advertisement and SHOULD NOT store the transient key and associated security material.

7.3.2 MLE Security MIB Settings

The MLE protocol defines its own mechanism to secure its payload and thus has its own corresponding MIB.

Thread security material SHALL be used to create a MLEKeyDescriptor entry in the MLE Key Table. The MLE Key Table SHALL be updated when key switching occurs (see Section 7.1.7, **Key Switching**). The MLE Key Table SHALL be maintained in a FIFO order where the latest

1 update displaces the oldest MLE Key Table entry. Transient MLE Key Table entries MAY be
2 used for key synchronization purposes.

3 **Note: The following tables describing attributes are conceptual only. A particular**
4 **implementation MAY represent the tables in whatever form is appropriate and**
5 **MAY optimize in any way, but the node behavior MUST match that of the**
6 **conceptual model.**

7 7.3.2.1 MLE Shared Frame Counter

8 A participating Thread node SHALL have the attribute defined in Table 7-16.

9 **Table 7-16. MLE Shared Frame Counter**

PIB Attribute	Value	Comment
<i>thrMLESharedFrameCounter</i>	Boolean	The MLE layer shares a frame counter with the MAC layer.

10 7.3.2.2 MLE Frame Counter

11 A participating Thread node SHALL have the attribute defined in Table 7-17.

12 **Table 7-17. MLE Frame Counter**

PIB Attribute	Value	Comment
<i>thrMLEFrameCounter</i>	-	Outgoing frame counter. Only present if <i>thrMLESharedFrameCounter</i> is FALSE.

13 7.3.2.3 MLE Key Table

14 A Thread node SHALL have the set defined in Table 7-18.

15 **Table 7-18. MLE Key Table**

PIB Attribute	Value	Comment
<i>thrMLEKeyTable</i>	MLEKeyDescriptor entries	One entry each for: <ul style="list-style-type: none">• Next MLE Key• Current MLE Key• Previous MLE Key One entry for each transient key
<i>thrMLEKeyTableEntries</i>	$3 + n$	n is the maximum number of simultaneous transient keys

16 7.3.2.3.1 MLE Key Descriptor

17 A Thread node SHALL have the MLEKeyDescriptor entry set defined in Table 7-19 for each
18 MAC Key.

1

Table 7-19. MLE Key Descriptor

MLEKeyDescriptor Attribute	Value	Comment
KeyIdLookupList	One MLEKeyIdLookupDescriptor entry	One entry for this MLE Key
KeyIdLookupListEntries	1	One entry for this MLE key
KeyDeviceList	MLEKeyDeviceDescriptor entries	Entries in the MLE device table
KeyDeviceListEntries	(variable)	Number of entries in MLE device table
Key	(variable)	The MLE key value

2 **7.3.2.3.2 Key ID Lookup for Key ID Mode 1**

3 The single MLEKeyIdLookupDescriptor entry for Key ID Mode 1 SHALL have the set defined
4 in Table 7-20.

5 **Table 7-20. Key ID Lookup for Key ID Mode 1**

MLEKeyIdLookupDescriptor Attribute	Value	Comment
LookupData	KeyIndex	KeyIndex is the key index associated with this MLE Key, which is the bottom 7 bits of $tKSC$ plus 1. Note that the concept of default key source is not used in MLEKeyIdLookupList.
LookupDataSize	1	Size 1 byte

6 **7.3.2.3.3 Key ID Lookup for Key ID Mode 2**

7 The single MLEKeyIdLookupList entry for Key ID Mode 2 SHALL have the set defined in
8 Table 7-21.

1

Table 7-21. Key ID Lookup for Key ID Mode 2

MLEKeyIdLookupDescriptor Attribute	Value	Comment
LookupData	Effective <i>tKSC</i> KeyIndex	KeyIndex is the key index associated with this MLE Key. This is typically a transient key and is thus calculated from the sequence counter. The effective <i>tKSC</i> is the value used to create the transient key.
LookupDataSize	5	Size 5 bytes

2

7.3.2.3.4 MLE Key Device List

3 Each MLEKeyDeviceDescriptor entry SHALL have the set defined in Table 7-22.

4

Table 7-22. Key Device List

MLEKeyDeviceDescriptor Attribute	Value	Comment
DeviceDescriptorHandle	Implementation specific	Indexes the appropriate MLEDeviceDescriptor.

5

7.3.2.4 MLE Device Table

6 A Thread node SHALL have the set defined in Table 7-23.

7

Table 7-23. MLE Device Table

PIB Attribute	Value	Comment
<i>thrMLEDeviceTable</i>	MLEDeviceDescriptor entries	One entry for each neighbor the Thread node is in communication with
<i>thrMLEDeviceTableEntries</i>	(variable)	Multiple entries

8

7.3.2.4.1 MLE Device Descriptor

9 A Thread node SHALL have the MLEDeviceDescriptor entry set defined in Table 7-24 for each MLE Key.

1

Table 7-24. MLE Device Descriptor

MLEDeviceDescriptor Attribute	Value	Comment
ExtAddress	8 bytes	The IEEE address of the neighboring node.
SharedFrameCounter	Boolean	Set to FALSE if the neighbor has provided separate Link Layer Frame Counter and MLE Frame Counter values, set to TRUE if only a Link Layer Frame Counter value was provided.
FrameCounter	4 bytes	The incoming frame counter corresponding to the most recently received frame from the neighboring node. Used for replay protection by checking that a subsequent incoming frame counter is greater than this value. Only present if SharedFrameCounter is FALSE.

2

7.3.3 MLE Auxiliary Security Header Format

3 The MLE message Auxiliary Security Header is used when a MLE frame is secured to provide
4 additional data required for security. For more information, see Section 4.3, Security
5 Formats, in Chapter 4, Mesh Link Establishment.

6 **7.3.3.1 Security Control Field**
7 The Security Control field SHALL have the values defined in Table 7-25.
8

Table 7-25. Security Control Field

Field	Value	Comment
Security Level	0x05	ENC-MIC-32 is the default value for Thread MLE security.
Key Identifier Mode	0x01 0x02	Mode 1: Key is determined from the 1-byte Key Index field of the Key Identifier field of the auxiliary security header in conjunction with <i>macDefaultKeySource</i> . Mode 2: Key is determined from the 4-byte Key Source and the 1-byte Key Index field of the Key Identifier field of the auxiliary security header.

9

7.3.3.2 Frame Counter Field

10 The Frame Counter field SHALL assume the value of the *thrMLEFrameCounter* MIB attribute.

1 **7.3.3.3 Key Identifier Field**

2 The Key Identifier field comprises the Key Index subfield and optional Key Source subfield.
3 The Key Source subfield SHALL only be present when Key Identifier Mode is 0x02.

4 **7.3.3.3.1 Key Index Subfield**

5 The Key Index Subfield SHALL be the key index associated with the active MLE Key.

6 **7.3.3.3.2 Key Source Subfield**

7 The Key Source subfield SHALL be the effective *tKSC* of the originator, in network byte
8 order.

9 **7.3.4 MLE Frame Counter Processing**

10 A primary purpose of MLE is the synchronization of frame counters between neighboring
11 devices.

12 **7.3.4.1 Frame Counter Values**

13 The two frame counter values used in the security processing of an incoming MLE message:

- 14 • auxFrameCounter
- 15 • storedFrameCounter

16 **7.3.4.1.1 auxFrameCounter**

17 auxFrameCounter is the frame counter from the incoming MLE message's Auxiliary Security
18 Header.

19 **7.3.4.1.2 storedFrameCounter**

20 storedFrameCounter is the frame counter stored for freshness checking of the incoming MLE
21 message. If the MLE Device Table contains a stored FrameCounter Descriptor for the sender
22 is present, its value is determined by the SharedFrameCounter attribute.

23 If SharedFrameCounter is TRUE, storedFrameCounter is the value of the FrameCounter
24 attribute of the MAC Layer Device Descriptor for the sender.

25 If SharedFrameCounter is FALSE, storedFrameCounter is the value of the FrameCounter
26 attribute of the MLE Device Descriptor for the sender.

27 **7.3.4.2 MLE Message Processing**

28 On receipt of the MLE message, look up the MLE Device Descriptor for the sender. There are
29 two possible outcomes:

- 30 • MLE Device Descriptor present.
- 31 • MLE Device Descriptor absent.

32 **7.3.4.2.1 MLE Device Descriptor Present**

33 There are two further possible outcomes based on freshness checking:

- 34 • auxFrameCounter is invalid.
- 35 • auxFrameCounter is valid.

1 **7.3.4.2.1.1 auxFrameCounter Is Invalid**

2 auxFrameCounter is invalid if storedFrameCounter is greater than or equal to
3 auxFrameCounter. If auxFrameCounter is invalid:

4 1. Discard the MLE message

5 **7.3.4.2.1.2 auxFrameCounter Is Valid**

6 auxFrameCounter is valid if storedFrameCounter is less than auxFrameCounter. If
7 auxFrameCounter is valid:

8 1. Check the integrity of the MLE message.

9 2. If the MLE message is successfully authenticated:

10 A. Fully process the MLE message according to its payload

11 **7.3.4.2.2 MLE Device Descriptor Absent**

12 1. Check the integrity of the MLE message.

13 2. If the MLE message is successfully authenticated and the MLE message contains a valid
14 Response TLV

15 A. If the MLE message contains a valid Link Layer Frame Counter TLV:

16 I. Look up the MAC layer Device Descriptor for the sender.

17 a. If absent, create the MAC layer Device Descriptor for the sender

18 II. Set the FrameCounter attribute of the MAC layer Device Descriptor for the sender
19 to the value in the Link Layer Frame Counter TLV.

20 III. Create the MLE Device Descriptor for the sender.

21 IV. If the MLE message contains both a Link Layer Frame Counter TLV and a MLE
22 Frame Counter TLV:

23 a. Set the SharedFrameCounter attribute of the MLE Device Descriptor for the
24 sender to FALSE.

25 b. Set the FrameCounter attribute of the MLE Device Descriptor for the sender to
26 the value in the MLE Frame Counter TLV.

27 V. If the MLE message contains only a Link Layer Frame Counter TLV:

28 a. Set the SharedFrameCounter attribute of the MLE Device Descriptor for the
29 sender to TRUE.

30 VI. auxFrameCounter MUST NOT be stored in either the MAC layer Device Descriptor
31 or the MLE Device Descriptor for the sender.

32 VII. Fully process the MLE message according to its payload.

33 3. If the MLE message is successfully authenticated and the MLE message does not contain
34 a valid Response TLV:

35 A. Partially process the MLE message to determine if a response should be sent.

36 **7.3.4.3 MLE Message Processing Example**

37 Figure 7-4 shows an example of an exchange of MLE messages and resulting processing. All
38 MLE messages are secured at the MLE layer and unsecured at the MAC layer.

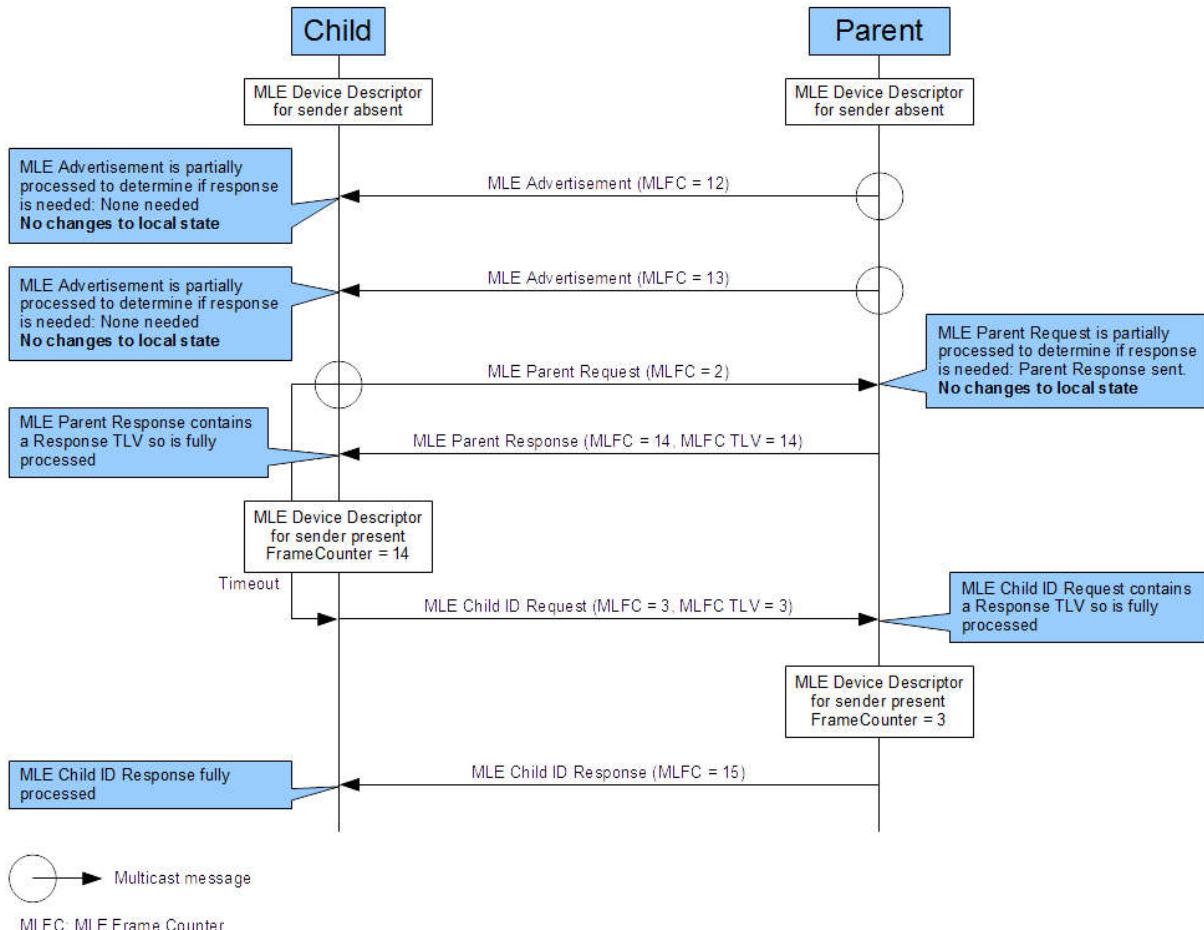


Figure 7-4. MLE Message Processing Example

7.4 DTLS

Note: TLS-ECJPAKE will be developed as an Internet Draft.

The cipher suite will provisionally be called TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256. It has the cipher suite value of { 0xC0, 0xFF }.

7.4.1 Conventions

7.4.1.1 Elliptic Curve Points

Elliptic curve points are represented in uppercase or subscripted uppercase, for example:

- X
- X_C

Elliptic curves are stored and used in uncompressed form. Conversion to and from elliptic curve points to octet strings is as specified in Sections 2.3.3 and 2.3.4 of [SEC1].

Point multiplication is shown as an elliptic curve point multiplied by a scalar integer using the '*' operator:

- 1 • G^*x
- 2 Point addition or subtraction is shown as the addition or subtraction of elliptic curve points
- 3 or scalar multiplied elliptic curve points using the '+' and '-' operators respectively:
- 4 • $X_1 + X_3 + X_4$
- 5 • $X^*h + G^*r$
- 6 • $X_s - X_4 * X_2 * s$

7.4.1.2 Integers

8 Integers are represented in lowercase or subscripted lowercase, for example:

- 9 • X
- 10 • X_c

11 Where expressed, integers are shown in hexadecimal and/or decimal form. Hexadecimal
12 numbers have an '0x' prefix:

- 13 • $0x12ab34cd$
- 14 • 3132110061

15 Integer multiplication is shown as two integers multiplied together using the '*' operator:

- 16 • $x * s$

17 Integer addition or subtraction is shown as the addition or subtraction of integers or
18 multiplied integers using the '+' and '-' operators respectively:

- 19 • $v - x * h$

7.4.1.3 Octet Strings

21 Octet strings are expressed in a hexadecimal form, with no '0x' prefix and with a space
22 separator, first octet leftmost, for example:

- 23 • $12\ ab\ 34\ cd$

7.4.2 Integer and Octet String Conversions

25 These conversions are used throughout Section 7.4.7, [Elliptic Curve J-PAKE Extensions](#).

7.4.2.1 Integer to Octet String Conversion

27 Integer to octet string conversion SHALL be performed as stated in Section 2.3.7 of [SEC1].
28 It is represented as follows:

$$M = \text{str}(m\text{len}, x)$$

30 where x , $m\text{len}$, and M are the parameters as stated in Section 2.3.7 of [SEC1].

7.4.2.2 Octet String to Integer Conversion

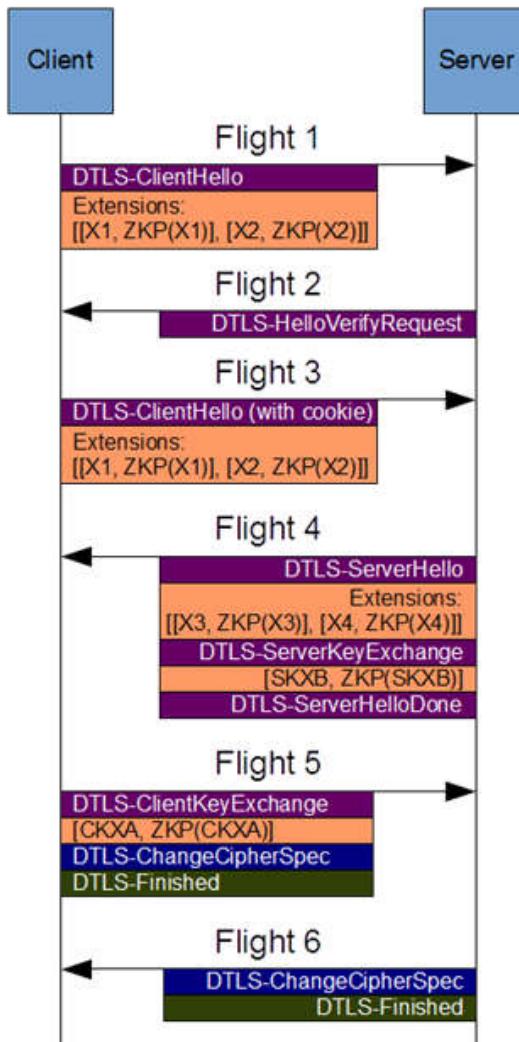
32 Octet string to integer conversion SHALL be as stated in section 2.3.8 of [SEC1]. It is
33 represented as follows:

$$x = \text{int}(m\text{len}, M)$$

35 where x , $m\text{len}$, and M are the parameters as stated in Section 2.3.8 of [SEC1].

1 7.4.3 Handshake

2 Figure 7-5 shows the basic DTLS message exchange used during the device handshake.



3 4 **Figure 7-5. Basic DTLS Message Exchange**

5 7.4.4 Failure Processing

6 If there are failures for any reason on client or server side, for example, Schnorr ZKP
7 verification or missing extensions, the handshake SHALL abort immediately and send a TLS
8 Error Alert message to the peer, using code 40 (handshake_failure) (see Section 7.2 of [\[RFC
9 5246\]](#)).

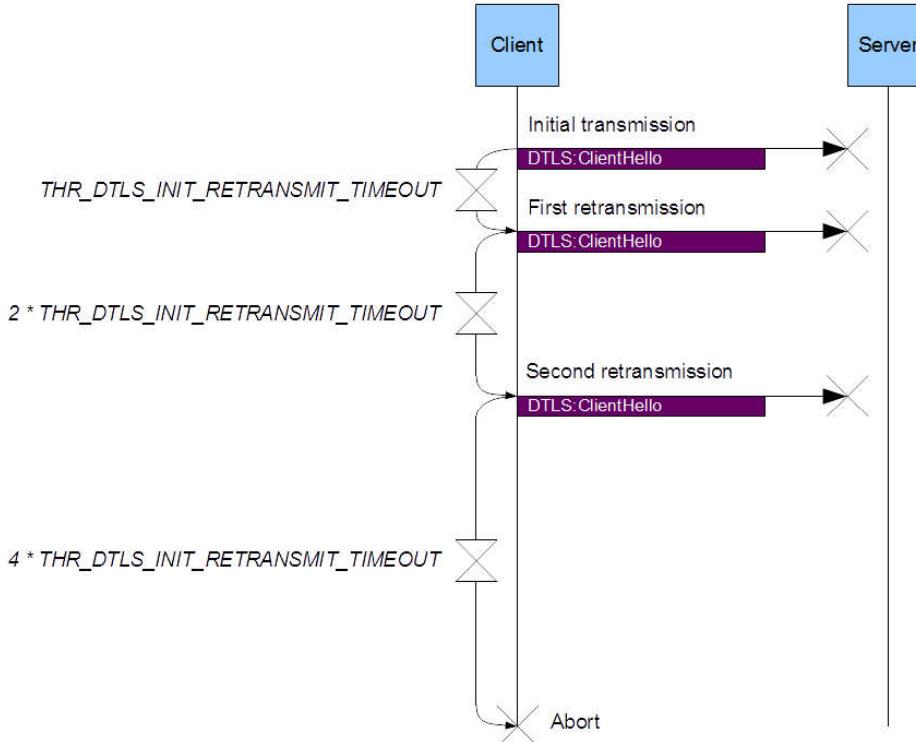
10 7.4.5 Retransmission Processing

11 Retransmissions are handled as described in [\[RFC 6347\]](#) Section 4.2.4. The initial retransmit
12 timer value SHALL be *THR_DTLS_INIT_RETRANSMIT_TIMEOUT* as defined in Table 7-1.

1 There SHALL be a maximum of *THR_DTLS_MAX_RETRANSMIT_COUNTER* retransmissions as
2 defined in Table 7-1. The timer value SHALL double on every retransmission.

3 **7.4.5.1 Retransmission example**

4 In the case there is no responding handshake message flight, transmissions will occur as
5 shown in Figure 7-6.



6
7 **Figure 7-6. Retransmission Example**

8 **7.4.6 Random Number Generation**

9 Where a random number is required as specified in this chapter, a Thread device MUST
10 implement a True Random Number Generator (TRNG) where the entropy source can be
11 proven to be from a verifiably random source, e.g., from thermal noise, avalanche noise, or
12 atmospheric noise.

13 **7.4.7 Elliptic Curve J-PAKE Extensions**

14 The structure definition used is defined in [\[RFC 5246\]](#).

15 **7.4.7.1 Existing Definitions**

16 TLS-ECJPAKE requires the following from [\[RFC 4492\]](#):

```
17 struct {  
18     opaque point <1..2^8-1>;  
19 } ECPoint;
```

20 The curve used SHALL be the NIST-P256 curve.

7.4.7.2 Additional Definitions

TLS-ECJPAKE requires additional definitions for:

- Public key and ZKP pair
- Schnorr ZKP (Zero Knowledge Proof)

7.4.7.2.1 Public Key and ZKP Pair

The TLS structure is as follows:

```
7 struct {  
8     ECPPoint X;  
9     ECSchnorrZKP zkp;  
10 } ECJPAKEKeyKP;
```

7.4.7.2.2 Schnorr Zero Knowledge Proof

The TLS structure is as follows:

```
13 struct {  
14     ECPPoint V;  
15     opaque r<1..2^8-1>;  
16 } ECSchnorrZKP;
```

See Section 7.4.7.3.2, [Schnorr ZKP Generation](#) for further details.

7.4.7.2.2.1 Schnorr ZKP Hash Calculation

The hash calculation is defined in Table 7-26.

Table 7-26. Schnorr ZKP Hash Calculation

X1, X2, X3 and X4 ZKPs	$h = \text{SHA-256}(G, V, X, ID) \bmod n$
X _s ZKP	$h = \text{SHA-256}(GB, V, X_s, ID_s) \bmod n$
X _c ZKP	$h = \text{SHA-256}(GA, V, X_c, ID_c) \bmod n$

Each item in the hash calculation is prepended with its length in octets represented an octet (length 4), formed by applying integer to octet string conversion as defined in Section 7.4.2.1, [Integer to Octet String Conversion](#).

For example, the length of an uncompressed octet string representation of a public key is 65 (decimal) therefore the octet string (length 4) representation of 65 in hexadecimal is:

- 00 00 00 41

Each public key (elliptic curve point) is first converted to an octet string according to section 2.3.3 of [SEC1].

0x00, 0x00, 0x00, 0x41

The order of the hash is as follows:

1. G (or GA, GB): Generator
2. V: ZKP ephemeral public key
3. X (or X_s, X_c): Public key to be verified
4. ID (or ID_c, ID_s): User ID ("client" or "server", see below)

- 1 The hash is therefore performed on the concatenation as follows:
- 2 • $H = \text{SHA-256}(\text{lenG} || G || \text{lenV} || V || \text{lenX} || X || \text{lenID} || ID)$
- 3 An integer representation of the hash is produced:
- 4 • $h = \text{int}(H) \bmod n$ (see Section 7.4.2.2, **Octet String to Integer Conversion**)
- 5 The hash calculation also requires identities of the originator. These identities are as
- 6 follows:

Originator	Name	Identity	Identity in hex	Length
Client	ID_c	"client"	63 6c 69 65 6e 74	6
Server	ID_s	"server"	73 65 72 76 65 72	6

7 See Section 7.4.8, **Test Program for Hash Generation** for an example test program.

7.4.7.3 ClientHello and ServerHello Extensions

9 The following from [\[RFC 4492\]](#) SHALL be present in ClientHello and ServerHello:

- 10 • Supported Elliptic Curves Extension (NamedCurve, EllipticCurveList)
- 11 • Supported Point Formats Extension (ECPointFormat, ECPointFormatList)

12 **Note: ECPointFormat SHALL be "uncompressed" (0).**

13 Additionally, the following extension SHALL be present in ClientHello and ServerHello:

```
14 enum { ecjpake_key_kp_pair(TBC) } ExtensionType;
15
16 struct {
17     ECJPAKEKeyKP ecjpake_key_kp_pair_list[2];
18 } ECJPAKEKeyKPPairList;
```

19 The value being used for `ecjpake_key_kp_pair` is 256.

20 This list is precisely two elements long and forms the (HX1, HX2) extension pair for
21 ClientHello and (HX3, HX4) extension pair for ServerHello.

7.4.7.3.1 Public Key Generation

23 In each case, the value for the public key part x of the `ECJPAKEKeyKP` structure is created as
24 follows:

25 The inputs are:

- 26 • Base point: G
- 27 • Order of the base point: n

28 The public key of the key pair is calculated as follows:

- 29 1. A random integer in the range 1 to $n-1$ is assigned to private key x
- 30 2. A public key associated with x is generated and assigned to X :
- 31 • $X = G*x$

7.4.7.3.2 Schnorr ZKP Generation

This section describes the general Schnorr ZKP generation. The values for the ZKP part `zkp.v` and `zkp.r` of the `ECJPAKEKeyKP` structure are created as follows. See Section 7.4.7.2.2.1, **Schnorr ZKP Hash Calculation** for further details.

The inputs are:

- Base point: G
- Order of the base point: n
- Identity of originator (see Section 7.4.7.2, **Additional Definitions**): ID , ID_c or ID_s depending on context)

Key pair to provide a ZKP of: (X, x) (public key: X , private key: x), where X is X_1, X_2, X_3 , or X_4 and x is x_1, x_2, x_3 , or x_4 , depending on context. The ZKP is generated as follows:

1. A random integer in the range 1 to $n-1$ is assigned to ephemeral private key v
2. An ephemeral public key associated with v is generated and assigned to V :
 - $V = G * v$
3. An integer representation of a hash is generated: h
 - $h = \text{int}(\text{SHA-256}(G, V, X, ID)) \bmod n$
4. A signature is created: r
 - $r = v - x * h \bmod n$

7.4.7.3.3 Schnorr ZKP Verification

This section describes the general Schnorr ZKP verification. See Section 7.4.7.2.2.1, **Schnorr ZKP Hash Calculation** for further details.

The inputs are:

- Base point: G
- Order of the base point: n
- Identity of originator (see Section 7.4.7.2, **Additional Definitions**): ID (ID_c or ID_s depending on context)
- Public key to be verified: X (X_1, X_2, X_3 , or X_4 depending on context)
- ZKP ephemeral public key: V
- ZKP signature: r

The ZKP is verified as follows:

1. An integer representation of a hash is generated: h
 - $h = \text{int}(\text{SHA-256}(G, V, X, ID)) \bmod n$
2. Compute V' :
 - $V' = X * h + G * r$
3. If V' is equal to V then the ZKP verifies, otherwise it does not verify

7.4.7.4 Shared Secret Conversion

The shared secret is in a variable length octet string format and is converted to an integer according to Section 7.4.2.1, **Integer to Octet String Conversion**, where m/len is the length of the shared secret in UTF-8 characters (not including any null termination characters if they exist) and M is the shared secret string comprised of UTF-8 characters.

1 **7.4.7.4.1 Example**

2 Shared secret:
3 “d45yj8e”
4 Equivalent octet string M using UTF-8 conversion (no null termination):
5 64 34 35 79 6a 38 65
6 Length $m\text{len}$:
7 7
8 Integer:
9 0x643435796a3865
10 28204901945981028 (decimal)

11 **7.4.7.5 ServerKeyExchange and ClientKeyExchange 12 Enumeration**

13 ServerKeyExchange and ClientKeyExchange require the following additional enumeration:
14 enum { ecjpake } KeyExchangeAlgorithm;

15 **7.4.7.6 ServerKeyExchange**

16 ServerKeyExchange for ecjpake SHALL BE formatted as follows:

```
17   struct {
18     ECPParameters curve_params;
19     ECJPAKEKeyKP ecjpake_key_kp;
20 } ServerECJPAKEParams;  
21  
22 select (KeyExchangeAlgorithm) {
23   case ecjpake:
24     ServerECJPAKEParams params;
25 } ServerKeyExchange;
```

26 This forms the SKXB extension for ServerKeyExchange.

27 **7.4.7.6.1 Public Key Generation**

28 This section describes generation of the public key part x of the ECJPAKEKeyKP structure.

29 The inputs are:

- 30 • Public keys X_1 , X_2 and X_3
- 31 • Private key x_4
- 32 • Shared secret: s (integer format; see Section 7.4.7.4, **Shared Secret Conversion**)

33 The public key of the key pair is calculated as follows:

- 34 1. A new generator is produced:
 - 35 • $GB = X_1 + X_2 + X_3$
- 36 2. A private key is generated:
 - 37 • $x_s = x_4 * s \bmod n$
- 38 3. A public key associated with x_s (i.e. $x_4 * s$) is generated and assigned to X_s :
 - 39 • $X_s = GB * x_s$

1 4. X_s is assigned to public key part x of the `ECJPAKEKeyKP`

2 **7.4.7.6.2 Schnorr ZKP Generation**

3 This section describes generation of the values for the Schnorr ZKP part `zkp.V` and `zkp.r` of
4 the `ECJPAKEKeyKP` structure. See Section 7.4.7.2.2.1, **Schnorr ZKP Hash Calculation** for
5 further details.

6 The inputs are:

- 7 • Base point: GB
- 8 • Order of the base point: n
- 9 • Identity of originator (see Section 7.4.7.2, **Additional Definitions**): Identity of
10 originator: ID_s
- 11 • Key pair to provide a ZKP of: (X_s, x_s) (public key: X_s , private key: x_s)

12 The Schnorr ZKP is generated as follows:

- 13 1. A random integer in the range 1 to $n-1$ is assigned to ephemeral private key v .
- 14 2. An ephemeral public key associated with v is generated and assigned to V :
 - 15 • $V = GB*v$
- 16 3. An integer representation of a hash is generated: h
 - 17 • $h = \text{int}(\text{SHA-256}(GB, V, X_s, ID_s)) \bmod n$
- 18 4. A signature is created: r
 - 19 • $r = v - x_s * h \bmod n$

20 **7.4.7.6.3 Schnorr ZKP Verification**

21 This section describes verification of the Schnorr ZKP. See Section 7.4.7.2.2.1, **Schnorr**
22 **ZKP Hash Calculation** for further details.

23 The inputs are:

- 24 • New generator: GB
- 25 • Order of the base point: n
- 26 • Identity of originator (see Section 7.4.7.2, **Additional Definitions**): Identity of
27 originator: ID_s
- 28 • Public key to be verified: X_s
- 29 • ZKP ephemeral public key: V
- 30 • ZKP signature: r

31 The Schnorr ZKP is verified as follows:

- 32 1. An integer representation of a hash is generated: h
 - 33 • $h = \text{SHA-256}(GB, V, X_s, ID_s) \bmod n$
- 34 2. Compute V' :
 - 35 • $V' = X_s * h + GB * r$
- 36 3. If V' is equal to V then the ZKP verifies, otherwise it does not verify.

7.4.7.7 ClientKeyExchange

ClientKeyExchange for ecjpake SHALL BE formatted as follows:

```
1 struct {  
2     ECJPAKEKeyKP ecjpake_key_kp;  
3 } ClientECJPAKEParams;  
4  
5 select (KeyExchangeAlgorithm) {  
6     case ecjpake:  
7         ClientECJPAKEParams params;  
8 } ClientKeyExchange;
```

This forms the CKXA extension for ClientKeyExchange.

7.4.7.7.1 Public Key Generation

This section describes generation of the public key part x of the ECJPAKEKeyKP structure.

The inputs are:

- Public keys: X_1 , X_3 and X_4
- Private key: x_2
- Shared secret: s (integer format; see Section 7.4.7.4, [Shared Secret Conversion](#))

The public key of the key pair is calculated as follows:

1. A new generator is produced:
 - $GA = X_1 + X_3 + X_4$
2. A private key is generated:
 - $x_c = x_2 * s \text{ mod } n$
3. A public key associated with x_c (i.e. $x_2 * s$) is generated and assigned to X_c :
 - $X_c = GA * x_c$
4. X_c is assigned to public key part x of the ECJPAKEKeyKP

7.4.7.7.2 Schnorr ZKP Generation

This section describes generation of the values for the Schnorr ZKP part $zkp.v$ and $zkp.r$ of the ECJPAKEKeyKP structure.

The inputs are:

- New generator: GA
- Order of the base point: n
- Identity of originator: ID_c (see Section 7.4.7.2.2.1, [Schnorr ZKP Hash Calculation](#))
- Key pair to provide a ZKP of: (X_c, x_c) (public key: X_c , private key: x_c)

The Schnorr ZKP is generated as follows:

1. A random integer in the range 1 to $n-1$ is assigned to ephemeral private key v .
2. An ephemeral public key associated with v is generated and assigned to V :
 - $V = GA * v$
3. An integer representation of a hash is generated: h
 - $h = \text{int}(\text{SHA-256}(GA, V, X_c, ID_c)) \text{ mod } n$

- 1 4. A signature is created: r
 - 2 • $r = v - x_c * h \bmod n$

7.4.7.7.3 Schnorr ZKP Verification

4 This section describes verification of the Schnorr ZKP.

5 The inputs are:

- 6 • New generator: GA
- 7 • Order of the base point: n
- 8 • Identity of originator: ID_c (see Section 7.4.7.2.2.1, **Schnorr ZKP Hash Calculation**)
- 9 • Public key to be verified: X_c
- 10 • ZKP ephemeral public key: V
- 11 • ZKP signature: r

12 The Schnorr ZKP is verified as follows:

- 13 1. An integer representation of a hash is generated: h
 - 14 • $h = \text{int}(\text{SHA-256}(GA, V, X_c, ID_c)) \bmod n$
- 15 2. Compute V' :
 - 16 • $V' = X_c * h + GA * r$
- 17 3. If V' is equal to V then the ZKP verifies, otherwise it does not verify

7.4.7.8 Server Secret Key Generation

19 This section describes Server secret key generation, including generation of the KEK (Key
20 Encryption Key) as used in commissioning, used for transporting network parameters to the
21 Client.

22 The inputs are:

- 23 • Public key of the client: X_c
- 24 • Public key: X_2
- 25 • Private key: x_4
- 26 • Shared secret: s (integer format; see Section 7.4.7.4, **Shared Secret Conversion**)

27 The pre-master secret is calculated as follows:

- 28 1. Compute PMSK:
 - 29 • $PMSK = (X_c - X_2 * x_4 * s) * x_4$
- 30 2. Compute PMS:
 - 31 • $PMS = \text{SHA-256}(\text{str}(32, X \text{ coordinate of PMSK}))$
- 32 3. The master secret and key expansion is generated according to Section 8.1 and Section
33 6.3 of [\[RFC 5246\]](#) respectively.
- 34 4. Compute KEK:
 - 35 • $KEK = \text{SHA-256}(\text{key expansion})[0..15]$

7.4.7.9 Client Secret Key Generation

This section describes Client secret key generation, including generation of the Key Encryption Key (KEK) as used in commissioning, used for transporting network parameters to the Client.

The inputs are:

- Public key of the server: X_s
- Public key: $X4$
- Private key: $x2$
- Shared secret: s (integer format; see Section 7.4.7.4, **Shared Secret Conversion**)

The pre-master secret (PMS) is calculated as follows:

1. Compute PMSK:

- $PMSK = (X_s - X4 * x2 * s) * x2$

2. Compute PMS:

- $PMS = \text{SHA-256}(\text{str}(32, X \text{ coordinate of PMSK})$

3. The master secret and key expansion is generated according to Section 8.1 and Section 6.3 of [\[RFC 5246\]](#) respectively.

4. Compute KEK. The KEK is 16 bytes:

- $KEK = \text{SHA-256}(\text{key expansion})[0..15]$

7.4.7.10 AEAD Record Protection

The cipher suite uses the AEAD_AES_128_CCM_8 AEAD record protection as described in Section 6.1 of [\[RFC 6655\]](#).

7.4.7.10.1 Nonce Formation

The nonce_explicit field in the DTLS record SHALL be comprised of:
Epoch (16-bit) || Sequence number (48-bit) in network byte order.

Example:

Epoch = 0x0001

Sequence number = 0x00000000000001

nonce_explicit = 00 01 00 00 00 00 00 01

7.4.8 Test Program for Hash Generation

The following is an example test program written in Python for hash generation.

```

1 import hashlib
2 import struct
3
4
5 def displaydataline(s, offs, npl):
6     '''Display npl values from string s'''
7     print '%04X: ' % offs + '%02X' * npl % struct.unpack('%dB' % npl, s)
8
9 def printhex(s, npl):
10    '''Print string s as hex with up to npl values per line'''
11    offs = 0
12    bufsize = len(s)
13    while bufsize > 0:
14        if bufsize < npl:
15            npl = bufsize;
16            displaydataline(s[offs:], offs, npl)
17            bufsize = 0
18        else:
19            displaydataline(s[offs:offs + npl], offs, npl)
20            offs = offs + npl
21            bufsize = bufsize - npl
22
23 def convert(intuple):
24    '''Convert tuple into string'''
25    return struct.pack(*(('%dB' % len(intuple)),) + intuple)
26
27 test_lenG = (0x00, 0x00, 0x00, 0x41)
28 test_G = \
29 (0x04, 0x6b, 0x17, 0xd1, 0xf2, 0xe1, 0x2c, 0x42, 0x47, 0xf8, 0xbc, 0xe6, 0xe5, 0x63, 0xa4, 0x40, \
30 0xf2, 0x77, 0x03, 0x7d, 0x81, 0x2d, 0xeb, 0x33, 0xa0, 0xf4, 0x1, 0x39, 0x45, 0xd8, 0x98, 0xc2, \
31 0x96, 0x4f, 0xe3, 0x42, 0xe2, 0xfe, 0x1a, 0x7f, 0x9b, 0x8e, 0xe7, 0xeb, 0x4a, 0x7c, 0x0f, 0x9e, \
32 0x16, 0x2b, 0xce, 0x33, 0x57, 0x6b, 0x31, 0x5e, 0xce, 0xcb, 0xb6, 0x40, 0x68, 0x37, 0xbf, 0x51, \
33 0xf5)
34
35 test_lenV = (0x00, 0x00, 0x00, 0x41)
36 test_V = \
37 (0x04, 0xfa, 0x9a, 0x24, 0x9d, 0x73, 0x6e, 0x30, 0x28, 0xd1, 0x2d, 0xf1, 0xdc, 0xfa, 0x22, 0xd1, \
38 0xed, 0x62, 0x82, 0xbf, 0xab, 0x27, 0x7c, 0x52, 0x56, 0xf3, 0xfd, 0x38, 0x07, 0xa5, 0xae, \
39 0xe0, 0x72, 0xfb, 0x4d, 0x9c, 0x2b, 0xd6, 0xa4, 0x70, 0xf7, 0xb4, 0xd0, 0xbd, 0xfb, 0x4a, 0x94, \
40 0x96, 0xcf, 0xcd, 0xd3, 0x53, 0xf9, 0x90, 0x3c, 0xa, 0x69, 0xa4, 0x4b, 0x18, 0xc6, 0xd2, 0x9b, \
41 0xb8)
42
43 test_lenX = (0x00, 0x00, 0x00, 0x41)
44 test_X = \
45 (0x04, 0x52, 0xa4, 0xda, 0x90, 0xa5, 0x15, 0x7f, 0xc0, 0xe5, 0x1f, 0x79, 0x4b, 0xe3, 0xbb, 0x3f, \
46 0x1d, 0xf8, 0xdf, 0xb1, 0xe3, 0x18, 0xa8, 0x10, 0xf2, 0x05, 0x2e, 0x64, 0xa8, 0xe8, 0x35, 0x64, \
47 0xe8, 0xe2, 0x8c, 0x17, 0x15, 0xab, 0xf7, 0x8d, 0x1f, 0x8b, 0x18, 0x99, 0x6d, 0x6a, 0xb7, 0xbd, \
48 0xcc, 0xbe, 0x52, 0x08, 0x1a, 0x3a, 0xe7, 0x65, 0x4b, 0xdf, 0x66, 0x62, 0xf5, 0x74, 0xe0, 0xfd, \
49 0x80)
50
51 test_lenUID = (0x00, 0x00, 0x00, 0x06)
52 test_UID = (0x63, 0x6c, 0x69, 0x65, 0x6e, 0x74) # "client"
53
54 # Instantiate SHA-256 hash object
55 m = hashlib.sha256()
56
57 # Update the hash with data:
58 # lenG || G || lenV || V || lenX || X || lenUID || UID
59
60 print "lenG:"
61 s = convert(test_lenG)
62 printhex(s, 16)
63 m.update(s)
64
65 print "\nG:"
66 s = convert(test_G)
67 printhex(s, 16)
68 m.update(s)
69
70 print "\nlenV:"
71 s = convert(test_lenV)
72 printhex(s, 16)
73 m.update(s)
74
75 print "\nV:"
76 s = convert(test_V)
77

```

```
1   printhex(s, 16)
2   m.update(s)
3
4   print "\nlenX:"
5   s = convert(test_lenX)
6   printhex(s, 16)
7   m.update(s)
8
9
10  print "\nX:"
11  s = convert(test_X)
12  printhex(s, 16)
13  m.update(s)
14
15  print "\nlenUID:"
16  s = convert(test_lenUID)
17  printhex(s, 16)
18  m.update(s)
19
20  print "\nUID:"
21  s = convert(test_UID)
22  printhex(s, 16)
23  m.update(s)
24
25  # Print the hash
26  print "\nSHA-256 hash:"
27  printhex(m.digest(), 16)
```

28 The above program should produce the following output:

```
29
30 lenG:
31 0000: 00 00 00 41
32
33 G:
34 0000: 04 6B 17 D1 F2 E1 2C 42 47 F8 BC E6 E5 63 A4 40
35 0010: F2 77 03 7D 81 2D EB 33 A0 F4 A1 39 45 D8 98 C2
36 0020: 96 4F E3 42 E2 FE 1A 7F 9B 8E E7 EB 4A 7C 0F 9E
37 0030: 16 2B CE 33 57 6B 31 5E CE CB B6 40 68 37 BF 51
38 0040: F5
39
40 lenV:
41 0000: 00 00 00 41
42
43 V:
44 0000: 04 FA 9A 24 9D 73 6E 30 28 D1 2D F1 DC FA 22 D1
45 0010: ED 62 82 BF AB 27 7C 52 56 F3 FD 38 07 A5 AE
46 0020: E0 72 FB 4D 9C 2B D6 A4 70 F7 B4 D0 BD FB 4A 94
47 0030: 96 CF CD D3 53 F9 90 3C 0A 69 A4 4B 18 C6 D2 9B
48 0040: B8
49
50 lenX:
51 0000: 00 00 00 41
52
53 X:
54 0000: 04 52 A4 DA 90 A5 15 7F C0 E5 1F 79 4B E3 BB 3F
55 0010: 1D F8 DF B1 E3 18 A8 10 F2 05 2E 64 A8 E8 35 64
56 0020: E8 E2 8C 17 15 AB F7 8D 1F 8B 18 99 6D 6A B7 BD
57 0030: CC BE 52 08 1A 3A E7 65 4B DF 66 62 F5 74 E0 FD
58 0040: 80
59
60 lenUID:
61 0000: 00 00 00 06
62
63 UID:
64 0000: 63 6C 69 65 6E 74
65
66 SHA-256 hash:
67 0000: EC F3 24 46 16 CE A5 34 58 46 D2 45 BA 27 63 36
68 0010: 50 C4 70 3D 56 0C 7A 7C 51 69 FE A7 A3 F7 79 10
```

CHAPTER 8 MESH COMMISSIONING PROTOCOL

Contents

8.1	Introduction.....	8-5
8.2	Terminology.....	8-5
8.3	Overview.....	8-7
8.4	Functional Specification	8-8
8.4.1	Commissioner Protocol.....	8-8
8.4.1.1	Commissioner Discovery of Thread Network.....	8-8
8.4.1.2	Commissioner Authentication	8-14
8.4.1.3	Commissioner Registration	8-15
8.4.2	Thread Management Protocol.....	8-15
8.4.2.1	Commissioner Petitioning	8-15
8.4.2.2	Commissioner Management	8-16
8.4.3	Dissemination of Datasets	8-19
8.4.3.1	Management of Thread Network Data Versions.....	8-19
8.4.3.2	Transmitting Thread Network Data	8-19
8.4.3.3	Receiving New Datasets	8-19
8.4.3.4	Delay Timer Management	8-20
8.4.3.5	Migrating Thread Network Partitions	8-20
8.4.4	Joiner Protocol	8-21
8.4.4.1	Discovery of Thread Network	8-21
8.4.4.2	IEEE 802.15.4 Beacon Frame Format	8-24
8.4.4.3	Bloom Filters.....	8-26
8.4.4.4	Joiner Provisional Join	8-27
8.4.4.5	Joiner Authentication	8-27
8.4.5	Joiner Finalization	8-29
8.4.5.1	Joiner Entrust.....	8-30
8.4.5.2	Joiner Provisioning	8-31
8.4.5.3	Joiner Session Close.....	8-31
8.4.6	Out-of-band Commissioning	8-31
8.4.6.1	First Border Agent Commissioning	8-31
8.4.6.2	Secure Out-of-band Commissioning	8-32
8.4.6.3	Out-of-band Network Commands.....	8-32
8.4.7	Diagrams	8-34
8.4.7.1	Topology	8-34
8.4.7.2	Flow Charts.....	8-38
8.4.7.3	Data Flows.....	8-41
8.4.7.4	Port Usage Diagrams	8-41

1	8.5 Message Format.....	8-44
2	8.6 Commissioner Scope TMF Messages.....	8-44
3	8.6.1 Petitioning Commands	8-44
4	8.6.1.1 COMM_PET.req – Commissioner Petition Request	8-44
5	8.6.1.2 COMM_PET.rsp – Commissioner Petition Response.....	8-45
6	8.6.1.3 COMM_KA.req – Commissioner Keep Alive Request.....	8-45
7	8.6.1.4 COMM_KA.rsp – Commissioner Keep Alive Response.....	8-46
8	8.6.2 Proxy Commands	8-46
9	8.6.2.1 UDP_RX.ntf – Proxy Receive UDP Notification.....	8-46
10	8.6.2.2 UDP_TX.ntf – Proxy Transmit UDP Notification	8-47
11	8.7 Commissioner and Thread Network Scope Commands	8-47
12	8.7.1 Relay Commands	8-48
13	8.7.1.1 RLY_RX.ntf – DTLS Relay Receive Notification	8-48
14	8.7.1.2 RLY_TX.ntf – DTLS Relay Transmit Notification.....	8-49
15	8.7.2 Network Management Commands	8-49
16	8.7.2.1 MGMT_GET.req – Get Management Data Request	8-50
17	8.7.2.2 MGMT_GET.rsp – Get Management Data Response	8-50
18	8.7.2.3 MGMT_SET.req – Set Management Data Request.....	8-50
19	8.7.2.4 MGMT_SET.rsp – Set Management Data Response.....	8-51
20	8.7.3 Updating the Commissioner Dataset	8-51
21	8.7.3.1 Updates from a Commissioner.....	8-51
22	8.7.3.2 MGMT_COMMISsIONER_GET.req – Get Commissioner Dataset	
23	Request.....	8-51
24	8.7.3.3 MGMT_COMMISsIONER_GET.rsp – Get Commissioner Dataset	
25	Response.....	8-52
26	8.7.3.4 MGMT_COMMISsIONER_SET.req – Set Commissioner Dataset	
27	Request.....	8-52
28	8.7.3.5 MGMT_COMMISsIONER_SET.rsp – Set Commissioner Dataset	
29	Response.....	8-53
30	8.7.3.6 TLV Encoding	8-53
31	8.7.3.7 Leader Behavior	8-54
32	8.7.4 Updating the Active Operational Dataset	8-54
33	8.7.4.1 Updates from a Commissioner.....	8-55
34	8.7.4.2 Updates from a Thread Device	8-55
35	8.7.4.3 MGMT_ACTIVE_GET.req – Get Active Operational Dataset	
36	Request.....	8-56
37	8.7.4.4 MGMT_ACTIVE_GET.rsp – Get Active Operational Dataset	
38	Response.....	8-57
39	8.7.4.5 MGMT_ACTIVE_SET.req – Set Active Operational Dataset	
40	Request.....	8-57
41	8.7.4.6 MGMT_ACTIVE_SET.rsp – Set Active Operational Dataset	
42	Response.....	8-58
43	8.7.4.7 TLV Encoding	8-58
44	8.7.4.8 Leader Behavior	8-59
45	8.7.5 Updating the Pending Operational Dataset.....	8-60
46	8.7.5.1 Updates from a Commissioner.....	8-60
47	8.7.5.2 Updates from a Thread Device	8-61

1	8.7.5.3	MGMT_PENDING_GET.req – Get Pending Operational Dataset Request.....	8-61
2	8.7.5.4	MGMT_PENDING_GET.rsp – Get Pending Operational Dataset Response.....	8-62
3	8.7.5.5	MGMT_PENDING_SET.req – Set Pending Operational Dataset Request.....	8-62
4	8.7.5.6	MGMT_PENDING_SET.rsp – Set Pending Operational Dataset Response.....	8-63
5	8.7.5.7	TLV Encoding	8-63
6	8.7.5.8	Leader Behavior	8-64
7	8.7.6	Concurrent Updates.....	8-65
8	8.7.6.1	MGMT_DATASET_CHANGED.ntf –Dataset Changed Notification	8-65
9	8.7.7	Orphaned End Devices	8-65
10	8.7.8	Merging Channel and PAN ID Partitions	8-66
11	8.7.8.1	MGMT_ANNOUNCE_BEGIN.ntf –Announce Begin Notification	8-66
12	8.7.9	Avoiding PAN ID Conflicts	8-67
13	8.7.9.1	MGMT_PANID_QUERY.qry –PAN ID Query Notification....	8-67
14	8.7.9.2	MGMT_PANID_CONFLICT.ans – PAN ID Conflict Notification	8-67
15	8.7.10	Collecting Energy Scan Information	8-69
16	8.7.10.1	MGMT_ED_SCAN.qry – Energy Detect Scan Notification...	8-69
17	8.7.10.2	MGMT_ED_REPORT.ans –Energy Detect Report Notification	8-70
18	8.8 Thread Network Scope Commands.....	8-71	
19	8.8.1	Leader Commands	8-71
20	8.8.1.1	LEAD_PET.req – Leader Petition Request	8-71
21	8.8.1.2	LEAD_PET.rsp – Leader Petition Response	8-71
22	8.8.1.3	LEAD_KA.req – Leader Petition Keep Alive Request.....	8-72
23	8.8.1.4	LEAD_KA.rsp – Leader Petition Keep Alive Response	8-72
24	8.8.2	Thread Network Data Update.....	8-73
25	8.8.2.1	Commissioning Data TLV	8-73
26	8.8.2.2	Thread Network Data Change Callback	8-74
27	8.8.2.3	Thread Network Data Change Command	8-74
28	8.8.2.4	Thread Network Data Update after Attach.....	8-75
29	8.9 Joiner Scope Commands	8-75	
30	8.9.1	Joiner Entrust Commands.....	8-75
31	8.9.1.1	JOIN_ENT.ntf – Joiner Entrust Notification	8-75
32	8.9.2	Joiner Session Commands	8-76
33	8.9.2.1	JOIN_FIN.req – Joiner Finalization Request	8-76
34	8.9.2.2	JOIN_FIN.rsp – Joiner Finalization Response	8-77
35	8.9.3	Joiner Provisioning Commands.....	8-77
36	8.9.3.1	JOIN_APP.req – Joiner Application Request	8-77
37	8.9.3.2	JOIN_APP.rsp – Joiner Application Response	8-78
38	8.10 MeshCoP TLV Formats.....	8-79	

1	8.10.1	Network Management TLVs	8-79
2	8.10.1.1	Channel TLV (0)	8-80
3	8.10.1.2	PAN ID TLV (1).....	8-80
4	8.10.1.3	Extended PAN ID TLV (2).....	8-81
5	8.10.1.4	Network Name TLV (3).....	8-81
6	8.10.1.5	PSKc TLV (4).....	8-81
7	8.10.1.6	Network Master Key TLV (5)	8-83
8	8.10.1.7	Network Key Sequence Counter TLV (6)	8-83
9	8.10.1.8	Network Mesh-Local Prefix TLV (7)	8-83
10	8.10.1.9	Steering Data TLV (8)	8-83
11	8.10.1.10	Border Agent Locator TLV (9)	8-84
12	8.10.1.11	Commissioner ID TLV (10)	8-85
13	8.10.1.12	Commissioner Session ID TLV (11).....	8-85
14	8.10.1.13	Active Timestamp TLV (14)	8-85
15	8.10.1.14	Commissioner UDP Port TLV (15)	8-86
16	8.10.1.15	Security Policy TLV (12)	8-86
17	8.10.1.16	Pending Timestamp TLV (51).....	8-87
18	8.10.1.17	Delay Timer TLV (52)	8-88
19	8.10.1.18	Channel Mask TLV (53)	8-88
20	8.10.2	MeshCoP Protocol Command TLVs	8-89
21	8.10.2.1	Get TLV (13).....	8-89
22	8.10.2.2	State TLV (16)	8-89
23	8.10.2.3	Joiner DTLS Encapsulation TLV (17).....	8-89
24	8.10.2.4	Joiner UDP Port TLV (18)	8-90
25	8.10.2.5	Joiner IID TLV (19)	8-90
26	8.10.2.6	Joiner Router Locator TLV (20)	8-90
27	8.10.2.7	Joiner Router KEK TLV (21).....	8-91
28	8.10.2.8	Count TLV (54).....	8-91
29	8.10.2.9	Period TLV (55)	8-92
30	8.10.2.10	Scan Duration TLV (56).....	8-92
31	8.10.2.11	Energy List TLV (57)	8-92
32	8.10.3	TMF Provisioning and Discovery TLVs	8-93
33	8.10.3.1	Provisioning URL TLV (32)	8-93
34	8.10.3.2	Vendor Name TLV (33).....	8-93
35	8.10.3.3	Vendor Model TLV (34).....	8-94
36	8.10.3.4	Vendor SW Version TLV (35).....	8-94
37	8.10.3.5	Vendor Data TLV (36)	8-94
38	8.10.3.6	Vendor Stack Version TLV (37).....	8-95
39	8.10.3.7	UDP Encapsulation TLV (48).....	8-95
40	8.10.3.8	IPv6 Address TLV (49)	8-96
41	8.10.3.9	Discovery Request TLV (128)	8-96
42	8.10.3.10	Discovery Response TLV (129).....	8-97
43	8.11 Parameters and Constants	8-97	
44	8.12 IANA Considerations	8-99	

8.1 Introduction

This chapter defines MeshCoP (Mesh Commissioning Protocol), a protocol for securely authenticating, commissioning and joining new, untrusted radio devices to a mesh network. Thread Networks comprise an autonomous self-configuring mesh of devices with IEEE 802.15.4 [IEEE802154] interfaces and a link-level security layer that requires each device in the mesh to possess the current, shared secret master key. Adding a new device to a Thread Network is the process of a human administrator authenticating it to the network as eligible for joining, followed by the commissioning of the device with the master key for the network over a secured channel. Only after completing this process, is the new device qualified to attach and participate on the secured Thread Network.

8.2 Terminology

There are several roles played by participants in the MeshCoP protocol (see Table 8-1 for brief descriptions of each role). Not all roles require a Thread interface to perform, and every role may be combined with other roles in a single device except the Joiner, which is a discrete role.

Table 8-1. Participant Roles in the MeshCoP

Participant Role	Description
Border Agent	Any device capable of relaying MeshCoP messages between a Thread Network and a Commissioner. The Commissioner may be reachable via either a non-Thread Network (external Commissioner) or a Thread Network (Native Commissioner). The Border Agent requires a Thread interface to operate and may be combined in any device with other Thread roles except the Joiner.
Commissioner	The currently elected authentication server for new Thread Devices and the authorizer for providing the Network Credentials they require to join the network. A device capable of being elected as a Commissioner is called a Commissioner Candidate. Devices without Thread interfaces may perform this role, but those that have them may combine this role with all other roles except the Joiner. This device may be, for example, a cell phone or a server in the cloud, and typically provides the interface by which a human administrator manages joining a new device to the Thread Network.
Commissioner Candidate	A device that is capable of becoming the Commissioner and either intends or is currently petitioning the Leader to become the Commissioner but has not yet been formally assigned the role of Commissioner.

Participant Role	Description
Joiner	The device to be added by a human administrator to a commissioned Thread Network. This role requires a Thread interface to operate and cannot be combined with another role in one device. Most importantly, the Joiner does not have Network Credentials.
Joiner Router	An existing Thread router or REED (Router-Eligible End Device) on the secure Thread Network that is one radio hop away from the Joiner. The Joiner Router requires a Thread interface to operate, and may be combined in any device with other roles except the Joiner role.
Leader	The single distinguished device in any Thread Network Partition that currently acts as a central arbiter of network configuration state. The Leader requires a Thread interface to perform and may be combined in any device with other roles except the Joiner.
On-mesh Commissioner	A combined role for certain collapsed cases where the Commissioner has a 15.4 interface, and possesses the Thread Network Credentials. An On-mesh Commissioner is always both a Commissioner and its own Border Agent. An On-mesh Commissioner may also be a Joiner Router.
Native Border Agent	A specific term for a device that is serving the role of Border Agent for a Native Commissioner. Such a device needs only a IEEE 802.15.4 radio [IEEE802154] to bridge between the unsecured 15.4 external neighbors and the secured Thread Network. As such, any Joiner Router MAY also serve as a Native Border Agent role.
Native Commissioner	A Commissioner or Commissioner Candidate that has the same interface used by the mesh. In the case of Thread, this would be an IEEE 802.15.4 radio [IEEE802154]. Unlike an On-mesh Commissioner, a Native Commissioner does not possess the Thread Network Credentials.

1 Table 8-2 describes the different kinds of credentials used in the MeshCoP protocol.

2 **Table 8-2. Credentials in the MeshCoP Protocol**

Credential	Description
Commissioning Credential	A human-scaled passphrase for use in authenticating that a device may petition to become the Commissioner of the network. This passphrase is encoded in utf-8 format and has a length of six (6) bytes minimum and 255 bytes maximum. This credential can be thought of as the network administrator password for a Thread Network.

Credential	Description
	The first device in a network, typically the initial Leader, MUST be out-of-band commissioned to inject the correct user generated Commissioning Credential into the Thread Network, or provide a known default Commissioning Credential to be changed later. This credential is used to derive an enhanced key using key stretching called the PSKc (Pre-Shared Key for the Commissioner) which is used to establish the Commissioner Session.
Commissioning Key (PSKc)	The PSKc (Pre-Shared Key for the Commissioner) is a derived key based on the Commissioning Credential which is used to establish the Commissioner Session. The PSKc is used as a passphrase input to a PAKE cipher suite, such as J-PAKE [draft-hao-jpake-03] , to establish a secure session between the Commissioner and Border Agent. All devices in a Thread Network will store the PSKc.
Joining Device Credential	A human-scaled passphrase for use in authenticating that a new Joiner device is the correct one. This passphrase is used in conjunction with a PAKE cipher suite to establish a secure session between the Commissioner and the Joiner. The Commissioner and the Joiner typically share the Joining Device Credential using some out-of-band mechanism such as scanning a barcode or entering a serial number from the Joiner device label. This credential when specifically encoded in binary is referred to as the PSKd (Pre-Shared Key for the Device). A Joining Device Credential is encoded as uppercase alphanumeric characters (base32-thread: 0-9,A-Y excluding I,O,Q, and Z for readability) with a minimum length of 6 such characters and a maximum length of 32 such characters.
Network Credentials	All the security and operational parameters required for a device to be part of a Thread Network as contained in the Joiner Entrust message, including the Network Master Key, Mesh Local Prefix, etc. All devices on a Thread Network store the Network Credentials, including the Leader, Routers, and End Devices.

1 Table 8-3 summarizes other useful MeshCoP terms.

2 **Table 8-3. Other Useful MeshCoP Terms**

Term	Description
Steering Data	An array of bytes present in a Discovery Response message or IEEE 802.15.4 beacon used to indicate the supported in-band facilities or means by which a Thread Device may be commissioned.

3 **8.3 Overview**

4 This chapter describes a comprehensive protocol for use in mesh networks to take a new device and securely join it to a Thread Network. The main goal of the MeshCoP is to

- 1 authenticate that the correct device is joined, and then to securely transfer the Network
2 Credentials to that device over an unsecured link.

3 **8.4 Functional Specification**

4 **8.4.1 Commissioner Protocol**

5 **8.4.1.1 Commissioner Discovery of Thread Network**

6 The commissioning process begins when a Commissioner Candidate, typically a mobile
7 phone with Wi-Fi, discovers the Thread Network through one of its Border Agents. Border
8 Agents advertise their availability to Commissioners using whatever service location protocol
9 is appropriate, for example, mDNS (multicast Domain Name System) [\[RFC 6762\]](#) or
10 Discovery Messages for Native Commissioners.

11 The discovery mechanism must provide the Commissioner Candidate both a communication
12 path and the Network Name string for the network, because the Network Name is used later
13 as a cryptographic salt for establishing the Commissioning Session.

14 **8.4.1.1.1 Commissioner Discovery - Native 802.15.4**

15 For Native Commissioners, the discovery process is identical to the one used by Joiners,
16 using Discovery Messages. (See Section 8.4.4.1, **Discovery of Thread Network** for more
17 information.) Native Commissioners use the Commissioner Port contained in the Discovery
18 Response.

19 **8.4.1.1.2 Commissioner Discovery - Ethernet / Wi-Fi**

20 Devices implementing commissioning Border Agents with external Ethernet, Wi-Fi, or some
21 other type of interface to a non-constrained IPv4 and/or IPv6 network with DNS query and
22 DNS response capabilities SHALL advertise on this external interface using DNS-SD (DNS
23 Service Discovery, [\[RFC 6763\]](#)) protocol to indicate availability of commissioning services to
24 Commissioner Candidates.

25 A DNS-SD Service Instance Name advertised by a Border Agent SHALL use the following
26 <Service> section:

27 _meshcop._udp

28 Unless otherwise configured by an administrator, the <Domain> section of a Service
29 Instance Name SHALL be set to the "local." domain and Border Agents SHALL implement
30 link-local Multicast DNS for the external interface as specified in [\[RFC 6762\]](#).

31 When the Border Agent advertises Commissioning services for a single Thread interface, the
32 <Instance> section of the DNS-SD Service Instance Name SHALL be a user-friendly name
33 identifying the device model or product. When selecting the name for the <Instance>
34 section, devices SHOULD adhere to the guidelines in Appendices B and C of [\[RFC 6763\]](#). An
35 example of a recommended format for the <Instance> section is "VendorName
36 ProductName". A complete Service Instance Name has the following recommended format:

37 VendorName ProductName._meshcop._udp_.local.

38 When the Border Agent advertises commissioning services for 2 or more Thread interfaces
39 for the same IP address on the external interface, a standalone DNS-SD Service Instance
40 Name SHALL be advertised for each Thread interface, each specifying a distinct
41 Commissioner Port in the SRV Answer Record. Any vendor-specific naming used to

1 differentiate multiple Thread interfaces in the same Border Agent MUST be applied to the
2 beginning of the <Instance> section, typically before VendorName. An example of Service
3 Instance Names advertised by the same physical device for 2 Thread interfaces are:

4 ThreadIf0 VendorName DeviceName._meshcop._udp_.local.

5 ThreadIf1 VendorName DeviceName._meshcop._udp_.local.

6 When a local network contains multiple Border Agents advertising the same Service
7 Instance Name, conflict resolution SHALL be performed as described in Section 9 of
8 [\[RFC 6762\]](#).

9 Commissioner Candidates looking to discover Border Agents will attempt a DNS query for
10 PTR records of the _meshcop._udp.<Domain> section of the commissioning Instance
11 Names as described in [\[RFC 6763\]](#).

12 The DNS query response from the Border Agent for the commissioning DNS-SD record
13 SHALL include one set of PTR+SRV+TXT Answer Records for each Service Instance Name
14 reflecting the characteristics of the service location and the Thread interface. The query
15 response SHALL also include matching A and/or AAAA Answer Records used for respective
16 IPv4 and IPv6 address resolution of the Border Agent external interface.

17 Format of the PTR Answer Record:

18 Name: _meshcop._udp.<Domain> (default: ._meshcop._udp.local.)

19 Type: PTR (12)

20 Class: IN (0x0001)

21 Cache flush: False

22 Time to live: <configurable, default 4500>

23 Data length: <length of data following>

24 Domain Name: <Instance>._meshcop._udp.<Domain>

25 Format of the SRV Answer Record:

26 Service: <Instance>

27 Protocol: _meshcop

28 Name: _udp.<Domain> (default: _udp.local.)

29 Type: SRV (33)

30 Class: IN (0x0001)

31 Cache flush: True

32 Time to live: <configurable, default 4500>

33 Data length:

34 Priority: 0

35 Weight: 0

36 Port: <Thread Mesh Commissioner Port (:MC) for the interface>

37 Target: <IP address A/AAAA Record Name>

38 Format of the TXT Answer Record:

39 Name: <Instance>._meshcop._udp.<Domain>

40 Type: TXT (16)

41 Class: IN (0x0001)

42 Cache Flush: True

- 1 Time to live: <configurable, default 4500>
- 2 Data length: <Length of key-value pair data following>
- 3 <List of length and key value metadata pairs as described in Table 8-4>
- 4 TXT Length: <Length of individual TXT key-value pair following>
- 5 TXT: <key-value pair in the format key=value>
- 6 Table 8-4 describes the format of the Key-Value pairs in the TXT Answer record.

7 **Table 8-4. DNS-SD TXT Answer Record Key-Value Pair Format**

TXT Key	TXT Value Format	Description and Constraints	Example
rv	UTF-8 string 1-3 bytes length	<p>Version of TXT record format specified as a UTF-8 encoded decimal.</p> <p>TXT record inclusion is required.</p> <p>All values other than '1' are reserved for subsequent versions of the record format and MUST NOT be used.</p>	<p>TXT Length: 4</p> <p>TXT: rv=1</p>
tv	UTF-8 string 1-8 bytes length	<p>Version of Thread Specification implemented by the Thread interface specified as a UTF-8 encoded decimal.</p> <p>TXT record inclusion is required.</p> <p>All values other than '1' are reserved for subsequent versions of the protocol specification and MUST NOT be used.</p>	<p>TXT Length: 8</p> <p>TXT: tv=1.1.0</p>
sb	32-bit binary bitmap	<p>State Bitmap as referenced in Table 8-5.</p> <p>TXT record inclusion is required.</p> <p>All reserved bitmap values MUST be set to 0.</p>	<p>TXT Length: 7</p> <p>TXT: sb=<00:00:00:31> (value is binary encoded)</p>
nn	UTF-8 string 1-16 bytes length	<p>TXT record inclusion is required when a connection to the Border Agent is allowed by the Connection Mode field of the State Bitmap. In such cases, the value of this field SHALL be used as the Network Name in the PSKc computation used for Commissioner Authentication as described in section 8.4.1.2.1, <u>Derivation of PSKc</u>.</p> <p>If the Border Agent Connection Mode is set to '0' (connection not allowed), the field MAY be elided.</p>	<p>TXT Length: 15</p> <p>TXT: nn=Network Name</p>

TXT Key	TXT Value Format	Description and Constraints	Example
		<p>When the State Bitmap Connection Mode is set to '1' (connection with shared PSKc) and the Thread Interface Status is set to '2' (interface active) the value of this TXT field MUST be set to the shared Thread Network Name.</p> <p>When the Thread Interface Status is set to '0' (interface not initialized), the value of the TXT field SHOULD be a string reflecting the Border Agent product or model name.</p>	
xp	8-byte binary data	<p>TXT record inclusion is required when a connection to the Border Agent is allowed by the Connection Mode field of the State Bitmap. In such cases, the value of this field SHALL be used as the Extended PAN ID in the PSKc computation used for Commissioner Authentication as described in section 8.4.1.2.1, <u>Derivation of PSKc</u>.</p> <p>If the Connection Mode is set to '0' (connection not allowed), the field MAY be elided.</p> <p>When the Connection Mode is set to '1' (connection with shared PSKc) and the Thread Interface Status is set to '2' (interface active), the value of this TXT field MUST be set to the shared Thread Network Extended PAN ID.</p> <p>When the Thread Interface Status is set to '0' (interface not initialized), the value of the TXT field SHOULD consist in the SHA-256 hash value of the factory IEEE EUI-64 of the Thread interface as specified in Chapter 3, PHY/MAC/6LoWPAN.</p>	<p>TXT Length: 11 TXT: xp=<00:00:00:00:00:00:00:01> (value is binary encoded)</p>
vn	UTF-8 string 2-24 bytes length	Vendor Name TXT record inclusion is optional.	<p>TXT Length: 9 TXT: vn=Vendor</p>

TXT Key	TXT Value Format	Description and Constraints	Example
mn	UTF-8 string 1-24 bytes length	Model Name TXT record inclusion is optional. If included, the value SHOULD refer to the Border Agent product model and not be specific to the Thread interface.	TXT Length: 12 TXT: mn=Device1.2
at	64-bit binary format, matching the Value format of the Active Timestamp TLV	Active Operational Dataset Timestamp of the current active Thread Network Partition of the Thread interface. TXT record inclusion is optional. Field SHALL not be included unless Thread Interface Status field is set to '2' (interface active).	TXT Length: 11 TXT: sb=<00:00:00:00:00:00:00:00> (value is binary encoded)
pt	32-bit binary format	Partition ID of the Thread interface. TXT record inclusion is optional. Field SHALL not be included unless Thread Interface Status is set to '2' (interface active).	TXT Length: 7 TXT: pt=<00:00:00:00> (value is binary encoded)
vd	Vendor-specific format 1-64 bytes length	Vendor-specific data which may guide application specific discovery. TXT record inclusion is optional. The Vendor Data Field MUST NOT expose privileged shared Thread Network information such as security material or network credentials.	TXT Length: 4-67 TXT: vv=<Vendor-specific>
vo	24-bit binary data	Device Vendor OUI as assigned by IEEE. TXT record inclusion is required if Vendor Data field is present. Otherwise, the value SHALL be elided.	TXT Length: 6 TXT: vv=<00:00:00>

- 1 Table 8-5 details the Border Agent State Bitmap advertised as the value of the "sb" TXT key.

1

Table 8-5. Border Agent State Bitmap

Bitmap Field Name	Bit Index	Bitmap Field Value Enumeration
Connection Mode	0-2	0: DTLS connection to Border Agent is not allowed 1: DTLS connection to Border Agent allowed with a user chosen network Commissioner Credential and shared PSKc for the active Thread Network Partition 2: DTLS connection to Border Agent allowed using the Border Agent Device Passphrase (PSKd) as the Commissioner Credential 3: DTLS connection to Border Agent allowed using a vendor defined Commissioner Credential 4-7: Reserved
Thread Interface Status	3-4	0: Thread interface is not active and is not initialized with a set of valid operational network parameters 1: Thread interface is initialized with a set of valid operational parameters, but is not actively participating in a Thread Network Partition 2: Thread interface is initialized with a set of valid operational parameters and is actively part of a Thread Network Partition 3: Reserved
Availability	5-6	0: Infrequent availability - Thread interface may become inactive when the Border Agent device is not in use 1: High availability – The Border Agent device and its Thread interface are part of stable, always-on network infrastructure (such as a wireless Access Point, Home Router, or Cable Modem) 2-3: Reserved
Reserved	7-31	Reserved

2

3 Format of the A or AAAA Answer Records pointed to by the SRV Target field:

4 Name: <IP address A/AAAA Record Name>

5 Type: <A or AAAA> (1 or 28)

6 Class: IN (0x0001)

7 Cache flush: True

8 Time to live: <configurable, default 120>

9 Data length: <4 for A record; 16 for AAAA record>

10 Address: <binary representation of IPv4 or IPv6 address>

11 **8.4.1.1.3 Commissioner Discovery - BLE / other**

12 A mechanism for discovery of a Thread Network by a Commissioner with a BLE (Bluetooth Low Energy) interface will be defined at a future date.

1 Future versions of Thread may also define the standard discovery methods for other
2 interfaces as appropriate.

3 **8.4.1.2 Commissioner Authentication**

4 The Commissioner Candidate, after having discovered the Thread Network of interest, would
5 then securely connect to the Thread Network using the Commissioning Credential.

6 The Commissioner Authentication step establishes a secured client/server socket connection
7 between the Commissioner Candidate and a Border Agent via DTLS (Datagram Transport
8 Layer Security, [\[RFC 6347\]](#) or optionally TLS (Transport Layer Security, [\[RFC 5246\]](#)). This
9 secured session is hereby known as the **Commissioning Session**.

10 The Commissioning Session SHOULD use the assigned UDP (User Datagram Protocol) port
11 number advertised during the discovery phase. This port is known as the Commissioner Port
12 ("MC").

13 The credential used to establish the Commissioning Session is known as the PSKc and is
14 generated as follows.

15 **8.4.1.2.1 Derivation of PSKc**

16 Because the PSKc is stored by all devices in the Thread Network, a key-stretching algorithm
17 is used to protect the actual user-generated passphrase, or Commissioning Credential, that
18 is used to derive it. A Commissioner will locally generate the PSKc, and only authenticate
19 with the Thread Network using that PSKc. This provides some protection for a user that may
20 reuse the same password across various services. The key stretching uses the PBKDF2 [\[RFC](#)
21 [2898\]](#), AES-CMAC-PRF-128 [\[RFC 4615\]](#).

22 PSKc = PBKDF2 (
23 PRF=AES-CMAC-PRF-128,
24 P=<Commissioning Credential>,
25 S="Thread" || <Extended PAN ID> || <Network Name>,
26 c=16384, dkLen=16)

27 Where the **Network Name** is all characters up to the maximum size or null terminator, and
28 || is the concatenation operator.

29 **8.4.1.2.2 Test Vector for Derivation of PSKc**

30 The following example test vector is provided to confirm the PSKc derivation algorithm is
31 implemented properly.

32 Inputs:

33 Commissioning Credential = "12SECRETPASSWORD34" [utf-8]
34 Extended PAN ID = 00 01 02 03 04 05 06 07 [hex]
35 Network Name = "Test Network" [utf-8]

36 Output:

37 Passphrase buffer: 31325345 43524554 50415353 574f5244 3334
38 Salt buffer: 54687265 61640001 02030405 06075465
39 7374204e 6574776f 726b
40 PSKc: c3f59368 445a1b61 06be420a 706d4cc9

8.4.1.3 Commissioner Registration

After the Commissioner Authentication is successful, the Commissioner Candidate registers its identity with its Border Agent by sending the COMM_PET.req message via the secure Commissioning Session.

The COMM_PET.req message contains a human-readable string that identifies the Commissioner. The COMM_KA.req message is sent periodically during the subsequent Commissioner Petitioning process as a keep-alive.

Failed attempts by a Commissioner to establish its authority with a Thread Network shall be rate-limited to COMM_PET_RETRY_DELAY second between attempts up to COMM_PET_RETRY_COUNT attempts and COMM_PET_ATTEMPT_DELAY seconds between such groups of attempts.

8.4.2 Thread Management Protocol

8.4.2.1 Commissioner Petitioning

The Border Agent unicasts to the Leader of the Thread Network Partition a request to petition its Commissioner Candidate to be the one elected Commissioner. This LEAD_PET.req message, which MAY include the Commissioner ID string, is sent securely over the mesh network using 802.15.4 security.

The Leader responds by either accepting or rejecting the Border Agent as a viable relaying agent for the Commissioner, and sends a LEAD_PET.rsp command to the appropriate Border Agent to inform it of the decision. If the LEAD_PET.rsp indicates an 'Accept' state, the Commissioner Candidate can now become the Commissioner.

Upon acceptance, the Leader updates its internal state to track the active Commissioner, updates the Commissioner Dataset to reflect the active Commissioning Session, and propagates it within the Thread Network as described in Section 8.8.2, **Thread Network Data Update**.

Potential Joiner Routers store the Commissioner ID and associated Border Agent information to allow communication with the Commissioner at a later stage (see Figure 8-1). Upon receiving the LEAD_PET.rsp from the Leader, the Border Agent relays the corresponding state in a COMM_PET.rsp message to the Commissioner Candidate. If the corresponding state value is 'Accept', the Commissioner Candidate becomes the Commissioner (see Figure 8-1).

The Commissioner will send periodically COMM_KA.req keep-alive messages to keep the Commissioner Session open. The Commissioner can also use the COMM_KA.req message to gracefully resign as Commissioner by setting the state of the message to 'Reject'. The Border Agent will relay the COMM_KA.req message as LEAD_KA.req message to the Leader.

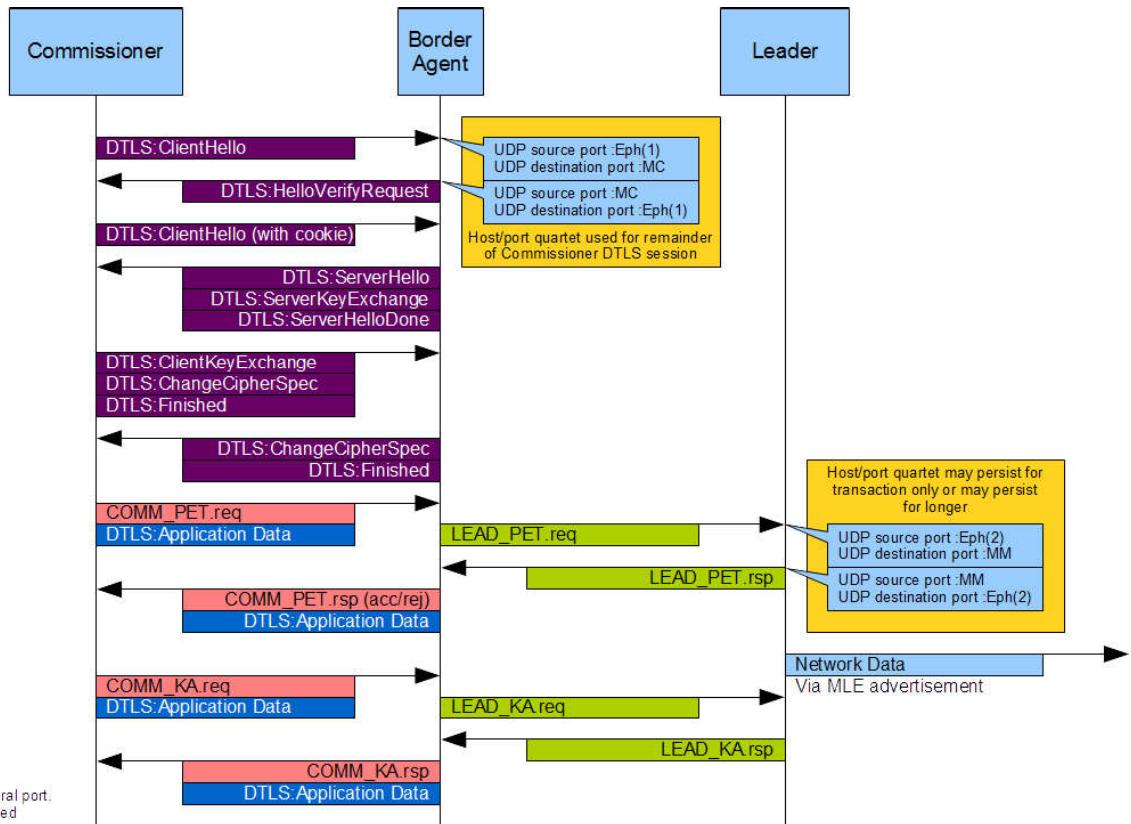


Figure 8-1. Commissioner Petitioning Data Flow

8.4.2.2 Commissioner Management

- Once confirmed, a Commissioner may manage the network through Commissioner and Operational Datasets.
- The Commissioner may send MGMT_GET, MGMT_ACTIVE_GET, and MGMT_PENDING_GET commands directly to the Border Agent over the Commissioning Session to get its view of the current network state, including important parameters such as the current Mesh Local Prefix for the network.
- The MGMT_PENDING_SET and MGMT_ACTIVE_SET, and other management commands may be sent to the Leader via the Border Agent over the secure Commissioning Session using the UDP proxy (see Figure 8-2). As these commands affect global network-wide state, they MUST be sent to the Leader ALOC to properly take effect and be applied to the entire Thread Network. A Commissioner MAY send a MGMT_GET.req or other management command to a Thread Device that is neither the Leader nor the Border Agent using the proxy transport by encapsulating it within a UDP_TX message.

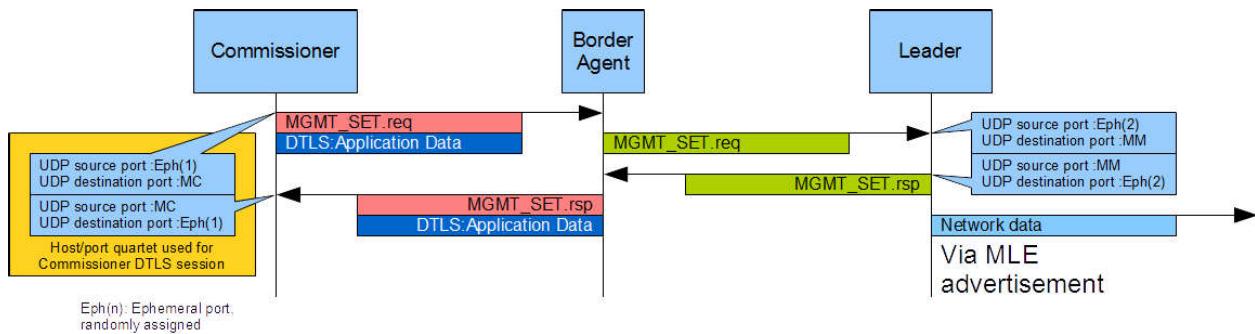


Figure 8-2. Commissioner Management Flow

Thread Devices maintain three Datasets:

- Commissioner Dataset: includes information about the Commissioner and are only valid within a given Thread Network Partition.
- Active Operational Dataset: includes parameters that are currently in use across an entire Thread Network.
- Pending Operational Dataset: includes parameters that are scheduled to be used across an entire Thread Network.

8.4.2.2.1 Commissioner Dataset

The Commissioner Dataset records information about the Commissioner and is only valid within a given Thread Network Partition and consists of:

- Border Agent Locator: the RLOC16 of the Border Agent associated with the active Commissioner. Presence of this field indicates that a Commissioner is active on the network; otherwise, joining is disabled.
- Commissioner Session ID: the session ID for the active Commissioner. The Leader maintains the session ID and increments the session ID anytime a unique Commissioner is elected or resigns.
- Steering Data: indicates which Joiners should attempt join the network. A Bloom Filter with zero value indicates that joining is disabled, which is the default for a new Commissioner.
- Joiner UDP Port: the preferred Joiner Port (:MJ) that Joiner Routers SHOULD use if available as specified by the active Commissioner.

8.4.2.2.2 Active Operational Dataset

The Active Operational Dataset records parameter values that are currently used by Thread Devices to participate within a given Thread Network. The set of Active Operational Dataset parameters are:

- Active Timestamp: a time value assigned to the Active Operational Dataset and used for comparing the priority between different Active Operational Datasets.
- Channel: the PHY-layer channel that the Thread Network uses for data transmissions.
- Channel Mask: the set of PHY-layer channels that a device should use when performing network discovery using Discovery Messages or MLE Announce frames as well as when performing MAC ED or active scans.

- 1 • Extended PAN ID: the Extended PAN ID that is used to identify the Thread Network.
- 2 • Mesh-Local Prefix: the Mesh-Local Prefix used for communication by Thread Devices
- 3 in the same Thread Network.
- 4 • Network Master Key: the key used to derive security material for MAC and MLE
- 5 message protection (i.e. *thrMasterKey* as defined in Section 7.1.2).
- 6 • Network Name: the human-readable string that identifies the Thread Network.
- 7 • PAN ID: the MAC-layer PAN ID that the Thread Network uses for data transmissions.
- 8 • PSKc: the PSKc is derived from the Commissioning Credential passphrase that a
- 9 Commissioner needs to connect to and authenticate with the Thread Network.
- 10 • Security Policy: specifies network administrator preferences for which security-
- 11 related operations are allowed or disallowed.

12 Thread Devices MUST persistently store the Active Operational Dataset.

13 Note that there may be use cases where more than one parameter needs to be changed
14 simultaneously. For example, if the Network Master Key is changed due to recycling a
15 Thread Device (either through disposal or transfer to third party), then the PSKc SHOULD
16 also be changed.

17 **8.4.2.2.3 Pending Operational Dataset**

18 The Pending Operational Dataset records parameter values that are scheduled to be used by
19 the Thread Network in the future. Some Active Operational Dataset values affect the ability
20 for neighboring devices to communicate, including:

- 21 • Channel
- 22 • Mesh-Local Prefix
- 23 • Network Master Key
- 24 • PAN ID

25 The Pending Operational Dataset is used to communicate changes to these values before
26 they take effect.

27 The Pending Operational Dataset contains all the parameters from the Active Operational
28 Dataset, with the addition of:

- 29 • Delay Timer: the time remaining until Thread Devices overwrite the values in the
30 Active Operational Dataset with the corresponding values in the Pending Operational
31 Dataset. The Delay Timer MUST be no greater than MAX_DELAY_TIMER.
- 32 • Pending Timestamp: a time value assigned to the Pending Operational Dataset and
33 used for comparing the priority between different Pending Operational Datasets.

34 The Delay Timer specifies how much time remains before devices switch to using the new
35 parameter values. When the Delay Timer expires, Thread Devices apply the Pending
36 Operational Dataset values to the Active Operational Dataset and remove the Pending
37 Operational Dataset.

38 A Thread Device MUST NOT use the Active Timestamp to determine the priority of a Pending
39 Operational Dataset. Only the Pending Timestamp may be used for that purpose.

40 Thread Devices MUST persistently store the Pending Operational Dataset.

1 **8.4.3 Dissemination of Datasets**

- 2 The Leader disseminates its Commissioner Dataset through a Thread Network Partition by
3 including it in an MLE Network Data Commissioning Data TLV.
- 4 The Active Operational Dataset is disseminated by being included in messages containing
5 MLE Network Data TLVs. The Active Operational Dataset, excluding the Active Timestamp, is
6 encoded inside an MLE Active Operational Dataset TLV using the appropriate MeshCOP sub-
7 TLVs. The Active Timestamp is encoded inside an MLE Active Timestamp TLV. If valid, the
8 Pending Operational Dataset is also disseminated alongside the Active Operational Dataset.
9 The Pending Operational Dataset, excluding the Pending Timestamp, is encoded inside an
10 MLE Pending Operational Dataset TLV using the appropriate MeshCOP sub-TLVs. The
11 Pending Timestamp is encoded inside an MLE Pending Timestamp TLV.
- 12 To support forward-compatibility, a Thread Device MUST store and forward sub-TLVs that
13 are not understood by the device received in the Commissioning Data TLV, Active
14 Operational Dataset TLV and Pending Operational Dataset TLV.

15 **8.4.3.1 Management of Thread Network Data Versions**

16 When the Leader changes any value in the Commissioner, Active Operational or Pending
17 Operational Datasets, the Leader MUST increment the Full Network Data Version,
18 VN_version.

19 When the Leader changes the Active Operational Dataset Active Timestamp or Pending
20 Operational Dataset Pending Timestamp, the Leader MUST increment the Stable Network
21 Data Version, VN_stable_version.

22 **8.4.3.2 Transmitting Thread Network Data**

23 When transmitting full Thread Network Data, the Network Data MUST include a
24 Commissioning Data TLV with the stable flag set to zero that contains the Commissioner
25 Dataset parameters encoded in appropriate sub-TLVs.

26 The Active Operational Dataset and Pending Operational Dataset are included in MLE
27 messages as specified in Section 4.7.1, Attaching to a Parent, in Chapter 4, Message Link
28 Establishment and Section 5.15.3, Propagation of Operational Datasets, in Chapter 5,
29 Network Layer.

30 **8.4.3.3 Receiving New Datasets**

31 When a new MLE Network Data TLV that contains a Commissioning Data TLV is received,
32 the Commissioner Dataset is replaced with the values encoded in the sub-TLVs. A Thread
33 Device MUST update its Commissioner Dataset with the received values.

34 When an MLE Active Operational Dataset TLV is received and the associated MLE Active
35 Timestamp TLV value is newer than the local Active Timestamp, the Active Commissioner
36 Dataset MUST BE updated using the values encoded in the included sub-TLVs and the
37 values committed to non-volatile memory.

38 When an MLE Pending Operational Dataset TLV is received and the associated MLE Pending
39 Timestamp TLV value is newer than the local Pending Timestamp, the Pending
40 Commissioner Dataset MUST BE updated using the values encoded in the included sub-TLVs
41 and the values committed to non-volatile memory.

8.4.3.4 Delay Timer Management

All Pending Operational Dataset parameter values are communicated unmodified from the Leader, with the exception of the Delay Timer. When receiving a new Pending Operational Dataset, a Thread Device initializes its local Delay Timer with the received value and begins counting down.

When forwarding the Pending Operational Dataset to another device, the Delay Timer TLV is included with the current value of the Delay Timer as maintained by the device. The objective is for the Delay Timer to expire at the same time on all devices.

When a Thread Device's Delay Timer reaches zero, the Thread Device compares Active Operational Dataset Active Timestamp with the Pending Operational Dataset Active Timestamp. If the Pending Operational Dataset Active Timestamp is newer or the Pending Operational Dataset contains a new Network Master Key, the Thread Device MUST:

1. Replace the Active Operational Dataset with the Pending Operational Dataset, removing the Delay Timer and Pending Timestamp at the same time.
2. Commit the new Active Operational Dataset to non-volatile memory.
3. Completely remove the Pending Operational Dataset.

The Pending Operational Dataset Active Timestamp may be older than the Active Operational Dataset Active Timestamp if the Pending Operational Dataset includes a different Network Master Key. This allows a Commissioner to roll back the Active Operational Dataset Active Timestamp, if needed.

In cases where a Thread Device loses its Delay Timer state (e.g. due to a reset), the Thread Device MUST first attempt to reattach using the Active Operational Dataset stored in non-volatile memory. If the Thread Device fails to find a Parent using the Active Operational Dataset, the Thread Device MUST attempt to reattach using the Pending Operational Dataset stored in non-volatile memory. For the reattach attempts, the Thread Device that has lost its Delay Timer state MUST NOT include a Pending Timestamp TLV in the MLE Child ID Request messages. If the Thread Device fails to find a Parent using the Pending Operational Dataset, the Thread Device MUST revert back to using the Active Operational Dataset and start a new Delay Timer with a conservative value that ensures expiration at or after the expected time. For example, a device that only has knowledge of the Delay Timer value when it received the Pending Operational Dataset MUST start the Delay Timer using the Delay Timer value it originally received.

8.4.3.5 Migrating Thread Network Partitions

8.4.3.5.1 Synchronizing Active Operational Datasets

A Thread Device MUST always use the Active Operational Dataset that is in use by that Thread Network Partition to properly participate in that Thread Network Partition. When attaching to a Thread Network Partition, a Thread Device MUST perform the following actions:

1. If the Parent advertises an Active Timestamp that is different than the Thread Device's local Active Timestamp, the Thread Device MUST retrieve and install the Active Operational Dataset from its Parent.
2. If the newly received Active Timestamp is newer than what the Thread Device had before attaching, the Thread Device discards the Active Operational Dataset it had before attaching.

- 1 3. If the newly received Active Timestamp is older than what the Thread Device had
- 2 before attaching, the Thread Device MUST attempt to update the Leader with its
- 3 newer Active Operational Dataset.

4 **8.4.3.5.2 Synchronizing Pending Operational Datasets**

5 If a Thread Device has a running Delay Timer when it detaches from a Thread Network
6 Partition, the Thread Device MUST continue to maintain the Delay Timer. If the Delay Timer
7 reaches zero and the Thread Device is not attached to any Thread Network Partition, the
8 Thread Device MUST update its Active Operational Dataset with the Pending Operational
9 Dataset, as described in the previous section.

10 If a detached Thread Device with a running Delay Timer reattaches to a Thread Network
11 Partition, the Thread Device MUST resynchronize its Pending Operational Dataset with the
12 Thread Network Partition as follows:

- 13 1. If the Thread Network Partition's Thread Network Data includes a Pending Timestamp
14 behind the local Pending Timestamp, the Thread Device MUST attempt to update the
15 Leader with its newer Pending Operational Dataset.
- 16 2. If the Thread Network Partition's Thread Network Data includes a Pending Timestamp
17 that is equal to or greater than the local Pending Timestamp, the Thread Device MUST
18 request the Pending Operational Dataset from a neighbor and update its Pending
19 Operational Dataset with the received values and reset the Delay Timer.

20 After attaching to a new Thread Network Partition, the Thread Device MUST NOT update its
21 Active Operational Dataset with the Pending Operational Dataset until it can resynchronize
22 its Pending Operational Dataset with the Thread Network Partition.

23 If the Thread Device detaches from the Thread Network Partition before it could
24 resynchronize its Pending Operational Dataset and the Delay Timer is already zero or
25 reaches zero, the Thread Device MUST update its Active Operational Dataset with the
26 Pending Operational Dataset.

27 **8.4.4 Joiner Protocol**

28 **8.4.4.1 Discovery of Thread Network**

29 **8.4.4.1.1 Native Discovery Messages**

30 There are two types of Thread Network Discovery Messages sent on the native IEEE
31 802.15.4 interface: Discovery Requests and Discovery Responses. These messages are
32 transmitted using the MLE protocol framework (see Chapter 4, Mesh Link Establishment).

33 When used for Thread, MLE Discovery Request and Discovery Responses MUST have the
34 MLE Security Suite set to '255' (No MLE Security) and MUST NOT include an MLE Auxiliary
35 Header.

36 **8.4.4.1.1.1 Discovery Request**

37 The Discovery Request message indicates that a Joiner or a Native Commissioner Candidate
38 is scanning for Thread Networks.

39 All types of Thread Devices MUST be capable to generate outgoing Discovery Request
40 messages containing a Thread Discovery TLV.

41 Discovery Request messages MUST have the Source Address in the IEEE 802.15.4 header
42 set to an extended (64-bit) random generated value.

1 In case a specific PAN ID of a Thread Network to be discovered is not known, Discovery
2 Request messages MUST have the Destination PAN ID in the IEEE 802.15.4 MAC header set
3 to be the Broadcast PAN ID (0xFFFF) and the Source PAN ID set to a randomly generated
4 value.

5 The Destination PAN ID in the IEEE 802.15.4 MAY be set to that of a Thread Network of
6 interest different from the Broadcast PAN ID. In this case, the PAN ID Compression MAC
7 header Frame Control flag SHALL be set to 'true' and the Source PAN ID SHALL be elided.

8 The Thread Discovery TLV for the Discovery Request contains the following Mesh
9 Commissioning Protocol TLVs (as sub-TLVs):

10 Discovery Request TLV

11 [list of one or more concatenated Extended PAN ID TLVs]

12 All Thread Routers and REEDs SHOULD process incoming Discovery Request messages at all
13 times. Processing MAY be limited only by availability of device resources to process the
14 incoming messages (see Section 8.4.4.1.1.3, **Discovery Messages Security**
Considerations.

16 When the message is identified as a Discovery Request which includes a Thread Discovery
17 TLV, the Router or REED MUST determine whether to send a Discovery Response in reply as
18 specified in Section 8.4.4.1.1.2, **Discovery Response**.

19 **8.4.4.1.1.2 Discovery Response**

20 The Discovery Response messages are Unicast messages sent back in reply to a Discovery
21 Request.

22 Discovery Response messages MUST have the IEEE 802.15.4 MAC header Source Address
23 set to the 64-bit MAC Extended Address of the Thread Router or REED and the Destination
24 Address set to the 64-bit Discovery Request Source Address.

25 If the Discovery Request was sent with a Destination PAN ID set to be the Broadcast PAN ID
26 (0xFFFF), the Discovery Response MUST set its IEEE 802.15.4 MAC header Source PAN ID
27 to the Thread Network PAN ID and the Destination PAN ID to the Discovery Request Source
28 PAN ID.

29 If the Discovery Request was sent with a Destination PAN ID set to the specific Thread
30 Network PAN ID, the Discovery Response MUST have the IEEE 802.15.4 MAC header
31 Destination PAN ID set to the Thread Network PAN ID, the PAN ID Compression MAC header
32 Frame Control flag set to 'true' and the Source PAN ID SHALL be elided.

33 On receipt of a Discovery Request message containing a Thread Discovery TLV, a Thread
34 Router or REED MUST send a Discovery Response message except for when at least one of
35 the two following situations occurs:

36 1. Joiner flag set and Joining not allowed:

- 37 • The Discovery Request TLV Joiner flag is set to '1' AND the Thread Network Partition
38 of the Router/REED does not allow Joining (there is no Active Commissioner or the
39 Active commissioner has disallowed all joiners via the Steering Data).

40 2. Extended PAN ID filter match:

- 41 • The Discovery Request payload contains one or more valid Extended PAN ID TLVs
42 AND the Router/REED Thread Network Extended PAN ID matches at least one of the
43 Extended PAN ID values.

44 If either of the two cases above applies, or the Discovery Request does not contain a Thread
45 Discovery TLV, the Router or REED SHOULD NOT prepare a reply and the Discovery Request
46 frame SHOULD be discarded.

1 The Thread Discovery TLV for the Discovery Response contains the following Mesh
2 Commissioning Protocol TLVs as sub-TLVs:

- 3 Discovery Response TLV
4 Extended PAN ID TLV
5 Network Name TLV
6 [Steering Data TLV]
7 [Joiner UDP Port TLV]
8 [Commissioner UDP Port TLV]

9 Discovery Responses SHOULD be delayed by a random time interval up to
10 DISCOVERY_MAX_JITTER.

11 All originators of the Discovery Request messages SHOULD be ready to process incoming
12 Discovery Response messages.

13 Originators of Discovery Request messages SHOULD stop processing incoming Discovery
14 Response messages after the expiration of a timer with a timeout set to DISCOVERY_TIME.
15 The timer is started at the moment of the Discovery Request transmission.

16 If the Steering Data TLV is not included, recipients of Discovery Response messages MUST
17 assume Joining is disabled (equivalent to Steering data value set to all zero).

18 If the Steering Data TLV is included, the Joiner UDP Port TLV MUST also be included. The
19 Joiner UDP Port in the Discovery Response is selected locally by the Router/REED. The value
20 of the Joiner UDP Port SHOULD be that specified in the Commissioner Dataset, if available.

21 If the Discovery Response TLV N flag is set to True, the Commissioner UDP Port TLV MUST
22 also be included. The Commissioner UDP Port is selected locally by the Router/REED.

23 **8.4.4.1.1.3 Discovery Messages Security Considerations**

24 Thread Router and REEDs MUST be specifically provisioned for immunity to Denial of Service
25 (DoS) situations that may be based on generation of multiple Discovery Requests in their
26 radio vicinity.

27 Thread Routers and REEDs MUST:

- 28 • Prioritize processing and forwarding of all other Thread data frames before Discovery
29 Messages.
- 30 • Keep minimal individual state for processing each incoming Discovery Request and
31 discard such state as quickly as possible.
- 32 • Set a quantifiable limit to internal resources allocated to replying to Discovery
33 Requests and if needed discontinue generation of Discovery Responses if the limit has
34 been reached.

35 Each originator of Discovery Requests MUST rate limit the generation of Discovery Requests
36 to sending at most 1 request per channel each DISCOVERY_RATE_LIMIT_INTERVAL.

37 **8.4.4.1.2 Joiner Discovery**

38 The Joiner discovers the Thread Network by sending a Discovery Request message on every
39 channel. The Discovery Response from the Joiner Router carries network identifiers and
40 Steering Data in the payload that helps the Joiner discover the correct network to connect
41 to.

42 The Steering Data field indicates which Joiners are currently being sought by the
43 Commissioner.

- 1 A Joiner MUST only attempt to join networks that have communicated a filter via the Steering Data that matches its Joiner ID or allows all Joiners.
- 2 In the case where multiple networks allow joining, the Joiner SHOULD use the following hunting algorithm.
 - 5 1. Collect Discovery Responses
 - 6 a. Prioritize responses with an exact match of Joiner ID in the Steering Data.
 - 7 b. Subprioritize responses by best signal strength.
 - 8 2. Attempt to join the prioritized networks one at a time, until one works.
 - 9 a. The Joiner will attempt to perform a DTLS handshake to establish the Joiner Session.
 - 10 b. If the DTLS handshake fails or there is a timeout, move on to the next network.
 - 11 3. Once the list is exhausted, the Joiner MAY rescan and repeat hunting with the new list.
 - 12 Only through a user-initiated action a device MAY form a new network as a Leader.



13

14 **Figure 8-3. Joiner Discovery Data Flow**

15 Joiners receiving Discovery Response messages MUST process these messages as valid and
16 attempt to join the respective networks, even when the value of the Protocol Version in the
17 Discovery Response TLV does not match the Joiner Protocol Version value.

18 **8.4.4.2 IEEE 802.15.4 Beacon Frame Format**

19 Thread Routers and REEDs MUST respond to IEEE 802.15.4-2006 Active Scans consisting in
20 Beacon Requests sent on the Active Operational Dataset channel by transmitting a Beacon
21 frame. This facilitates backwards compatibility with previous versions of this specification
22 and allows coexistence with other IEEE 802.15.4 networks with regard to resolving PAN ID
23 conflicts.

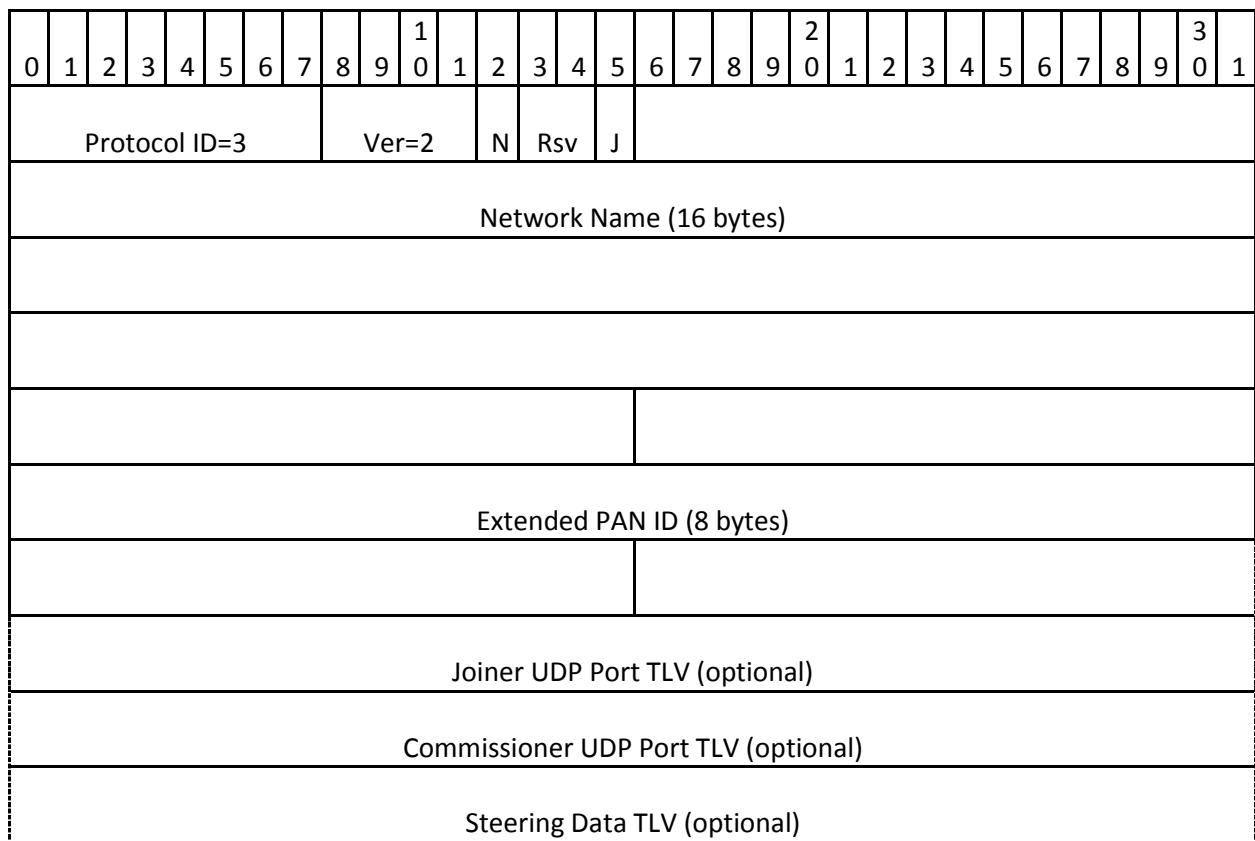
24 The format of the Beacon payload sent in response to an Active Scan is determined by the
25 B-bit in the Security Policy.

26 When the Security Policy B bit is not set (has value 0), the Beacon payload MAY be elided or
27 have an application specific value.

28 When the Security Policy B bit is set (has value 1), the format detailed in Figure 8-4 MUST
29 be used for the Beacon payload.

30

1



2

Figure 8-4. Beacon Payload

3 Protocol ID

4 This value is set to '3'.

5 J (Joining Permitted) bit

6 Set when an active Commissioner has set the Steering Data filter on this network.
7 Joiners may only attempt to join on the Joiner Port when this bit is set and they match
8 the Steering Data filter.

9 N (Native Commissioner) bit

10 When the N bit is set, signals that the device sending the Beacon frame can serve as a
11 Border Agent for a Commissioner Candidate with a native IEEE 802.15.4 interface to
12 petition to be Commissioner over the unsecured IEEE 802.15.4 interface.

13 Rsv

14 Reserved. MUST be set to zero.

15 Ver

16 Protocol Version = 2

17 Devices implementing this version of the specification MUST set the Protocol Version to
18 value 2.

- 1 Network Name
- 2 String in UTF-8 encoding matching the value field of Network Name TLV to provide human-readable name for the network. Pad with null characters if the string is shorter than 16 bytes.
- 5 Extended PAN ID
- 6 Extended network identifier for the current network.
- 7 Steering Data TLV
- 8 Steering Data indicates which Joiners are eligible to join the Thread Network. This field is included only if the active Commissioner has set a Steering Data filter. When the Steering Data is included, the Joining Permitted bit MUST be set to '1'.
- 11 Joiner UDP Port TLV
- 12 Optional TLV to specify the UDP port to use as the Joiner Port. The Joiner UDP Port TLV MUST be included when the Steering Data TLV is included and allows joining. The Joiner UDP Port in the Beacon payload is selected locally by the Router/REED. The value of the Joiner UDP Port SHOULD be that specified in the Commissioner Dataset if available.
- 16 Commissioner UDP Port TLV
- 17 Optional TLV to specify the UDP port to use as the Commissioner Port for Native Commissioners. If the N bit is set, the Commissioner UDP Port TLV MUST be included. The Commissioner UDP Port is selected locally by the Router/REED.
- 20 The beacon MUST be sent from an MAC Extended Address to provide EID-based, stable addressing during joining and for backwards compatibility.

8.4.4.3 Bloom Filters

- 23 Thread Joiner discovery uses bloom filters in order to steer Joiners to attempt joining only networks where their Joiner Identifier (Joiner ID) is expected by a Commissioner.
- 25 The bloom filter provides a compact representation for testing if the probability of a specific Identifier to be included in a specific set is higher than 0. Proper selection of bloom filter parameters also allows a low enough probability of false positives.
- 28 When included in TLVs for testing inclusion of Thread Identifiers, the bloom filter must have a modulo 8 length and is represented as a big endian bit mask. For example, a 128-bit bloom filter appears in the following order in the respective TLVs: |127|126|...|2|1|0|.
- 31 The parameters for the bloom filters used for Thread Identifiers are:
 - 32 1. $k = 2$ (two hash functions)
 - 33 2. m = the number of bits in the bloom filter
 - 34 3. n = number of Identifiers included in the set
- 35 The two hash functions are:
 - 36 1. hash1fn = CRC16-CCITT with polynomial 0x1021
 - 37 2. hash2fn = CRC16-ANSI with polynomial 0x8005
- 38 The probability of collisions can be calculated as follows:
 - 39 1. $p = \text{probability of collisions. } p = (1 - e^{(-k * n / m)})^k$
- 40 A device may set the length m as required to get a reasonably low collision probability of a false positive.

1 For a device to generate the bloom filter for a set of included Identifier:

2 • bloom_filter = 0

3 • For each Identifier in the set compute:

4 ▪ hash1_bit = 1 << (hash1fn(Identifier) % m)

5 ▪ hash2_bit = 1 << (hash2fn(Identifier) % m)

6 ▪ bloom_filter |= hash1_bit | hash2_bit

7 For a device to query whether an Identifier is part of a given set, perform the following:

8 • m = length_in_bits(Bloom Filter value)

9 • bloom_filter = Bloom Filter value

10 • hash1_bit = 1 << (hash1fn(Identifier) % m)

11 • hash2_bit = 1 << (hash2fn(Identifier) % m)

12 • value_bloom = hash1_bit | hash2_bit

13 • present_in_set = (value_bloom == (bloom_filter & value_bloom))

14 The following table shows for various values of n, the probability of collisions p when m=128
15 (16 bytes):

16 n=1 p=0.000
17 n=2 p=0.001
18 n=3 p=0.002
19 n=4 p=0.004
20 n=5 p=0.006
21 n=10 p=0.021
22 n=12 p=0.029
23 n=20 p=0.072
24 n=25 p=0.105
25 n=30 p=0.140
26 n=50 p=0.294
27 n=100 p=0.625
28 n=200 p=0.914
29 n=1000 p=1.000

30 **8.4.4.4 Joiner Provisional Join**

31 The Joiner establishes an unsecured local-only link to the Joiner Router by configuring its
32 MAC (Media Access Control) layer with the network parameters (Channel, PAN ID etc.)
33 obtained during Joiner Discovery, and sending DTLS records to a known UDP port on this
34 unsecured interface.

35 The Joiner MUST use the assigned UDP port (:MJ) as the destination port for this purpose,
36 having the value specified in the Joiner UDP Port TLV from the Discovery Response payload.
37 This port is known as the **Joiner Port**.

38 **8.4.4.5 Joiner Authentication**

39 The Joiner, throughout the Joining process, sends DTLS handshake messages without MAC
40 security transported over UDP to a Joiner Router. A Joiner Router relays these messages on
41 to the Border Agent or to itself in the collapsed case where the Joiner Router is the active
42 Border Agent. The Border Agent relays these messages further to the Commissioner. Any
43 Joiner Router or Border Agent has no knowledge of the content of these Joiner messages,
44 and only filters data on the unsecured channel based on UDP datagrams on the agreed-

1 upon Joiner Port. The Joiner Router rate-limits forwarding of such unsecured messages to
2 JOINER_RELAY_RATE_LIMIT.

3 The Joiner initially identifies itself to the Commissioner at the beginning of Joiner
4 Authentication by sending a UDP datagram with a DTLS-ClientHello handshake record
5 payload to the Joiner Router on the Joiner Port (:MJ). This initial DTLS-ClientHello is
6 intended to allow the Commissioner to return a cookie to the Joiner as a simple token,
7 which the Joiner in turn returns to the Commissioner to ensure the Joiner is genuine:

8 J → UDP({DTLS-ClientHello}) → JR

9 Upon receiving the UDP datagram, the Joiner Router encapsulates the DTLS-ClientHello
10 handshake record payload in a RLY_RX.ntf message, adding the source interface identifier
11 (IID) of the original datagram (in this case, the IID derived from the hashed EUI-64 of the
12 Joiner) and the source UDP port. The Joiner Router also adds its own RLOC. This additional
13 addressing information is used for the return path. The Joiner Router then sends this
14 RLY_RX.ntf message to the Border Agent:

15 JR → RLY_RX(J IID, J port, JR RLOC, {DTLS-ClientHello}) → BA

16 The appropriate Border Agent address to use is determined from the Thread Network Data
17 notification that was sent out earlier. Alternatively, the Joiner Router can query for the
18 information by unicasting a MGMT_GET.req message to the Leader. Forwarding of Joiner
19 messages via RLY_RX.ntf by the Joiner Router MUST be rate-limited to
20 JOINER_RELAY_RATE_LIMIT to prevent DoS (Denial of Service) attacks, as these are
21 initiated by unauthenticated devices on an unsecured link.

22 Upon receiving the RLY_RX.ntf message, the Border Agent sends it over the secure
23 Commissioning Session:

24 BA → {DTLS-AppData(RLY_RX(J IID, J port, JR RLOC, {DTLS-ClientHello}))} → C

25 Upon receiving the encapsulated DTLS-ClientHello handshake record, the Commissioner has
26 all the addressing information (Joiner IID, Joiner Port and Joiner Router RLOC) it requires in
27 order to communicate via relay back to the Joiner. The Commissioner may also check the
28 Joiner IID against its list to ensure the original DTLS-ClientHello has come from an expected
29 Joiner; this is only a cursory check as it cannot verify the Joiner until authentication has
30 completed. The Commissioner generates a DTLS-HelloVerifyRequest with a cookie and
31 encapsulates this along with the addressing information in a RLY_TX.ntf message and sends
32 it to the Border Agent:

33 C → {DTLS-AppData(RLY_TX(J IID, J Port, JR RLOC, {DTLS-HelloVerifyRequest}))} →
34 BA

35 Upon receiving the RLY_TX.ntf message, the Border Agent inspects the RLOC from the
36 RLY_TX.ntf message and uses it to send the message to the Joiner Router:

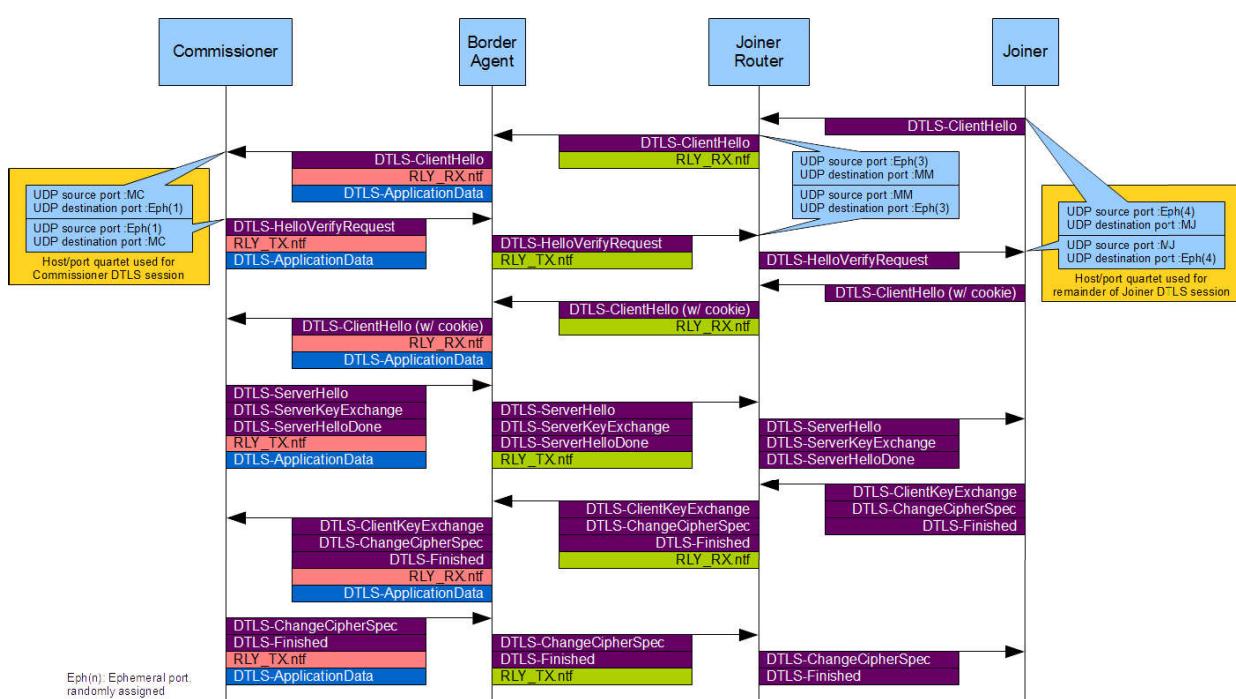
37 BA → RLY_TX(J IID, J port, JR RLOC, {DTLS-HelloVerifyRequest }) → JR

38 Upon receiving the RLY_TX.ntf message, the Joiner Router inspects the Joiner IID and Joiner
39 Port from the RLY_TX.ntf message and uses it to send a UDP datagram with the DTLS-
40 HelloVerifyRequest as payload to the Joiner:

41 JR → UDP({DTLS-HelloVerifyRequest}) → J

42 This completes the illustration of the relay mechanism for a round trip from the Joiner to the
43 Commissioner and back to the Joiner.

- 1 The Joiner then proceeds to send a DTLS-ClientHello handshake record (including the cookie from the DTLS-HelloVerifyRequest), which is relayed to the Commissioner using the same relay mechanism illustrated above.
- 2
- 3
- 4 The Commissioner, upon reception of a DTLS-ClientHello message with a cookie from one of its scanned Joiners, inspects the Joiner IID, and looks up the Joining Device Credential to use to continue the DTLS handshake for that Joiner. It then relays the combined DTLS-ServerHello, DTLS-ServerKeyEx and DTLS-ServerHelloDone handshake records back to the Joiner using the same relay mechanism illustrated above.
- 5
- 6
- 7
- 8
- 9 The two devices continue through the handshake process, leveraging up to two relay points.
- 10 If the Commissioner detects a problem, it terminates the DTLS session, which signals that
- 11 the Joiner is rejected. The Joiner then moves on to another network in the hunting
- 12 algorithm list. Upon completion of this DTLS handshake, the session that has been
- 13 established will hereby be known as the Joiner Session.
- 14 Figure 8-5 illustrates the Joiner Authentication data flow.



15 **Figure 8-5. Joiner Authentication Data Flow**

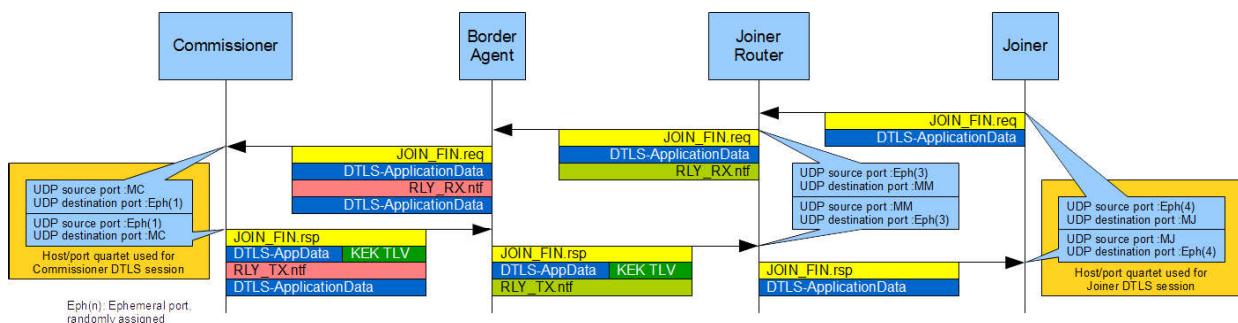
16 **8.4.5 Joiner Finalization**

17 After the Joiner is authenticated, it may negotiate whether the Commissioning application is
18 capable of fully provisioning it.

20 **Provisioning** is the process of associating a device to the appropriate service, and
21 performing any application or vendor-specific configuration.

22 The Joiner sends a JOIN_FIN.req specifying either that it simply wants the Network
23 Credentials, or that it requires a vendor-specific Commissioning application that is able to
24 provide subsequent provisioning. The provisioning application is specified using a URL that
25 redirects a mobile device to the correct application in an app store, or to a web interface

- 1 that can handle the proper provisioning. A missing or empty URL signifies that the Joiner does not require provisioning.
- 2 If the Joiner only wants the Network Credentials, the Commissioner responds with a JOIN_FIN.rsp that contains an Accept and the KEK attachment. The Joiner Session is then closed.
- 3 If the Joiner wants additional provisioning, the Commissioner responds with a JOIN_FIN.rsp that contains an Accept and the KEK attachment. The Joiner Session then remains open and is closed after provisioning.
- 4 If the Commissioner application is unable to process the JOIN_FIN.req requiring provisioning, the Commissioner responds with a JOIN_FIN.rsp that contains a Reject and the KEK attachment. The Joiner Session is then closed.
- 5 The wrong Commissioner application SHOULD present a dialog to the user with the link to navigate to the correct Commissioner application. The Joiner reattempts joining the same network anticipating that the Commissioner is now running the correct application.
- 6 Figure 8-6 illustrates the Joiner Finalization data flow.



16

17 **Figure 8-6. Joiner Finalization Data Flow**

18 **8.4.5.1 Joiner Entrust**

19 Joiner Entrust is the process of final handoff of Network Credentials to the Joiner. Once the
20 Commissioner has authenticated a Joiner, it proceeds to signal to the Joiner Router that the
21 Joiner is to be entrusted with the Network Credentials.

22 In the KEK (Key Encryption Key) method, the Commissioner provides a shared secret
23 between it and the Joiner to the Joiner Router.

24 With this mechanism, the Commissioner signals that the Joiner can be entrusted by
25 including the KEK TLV in the RLY_TX message that encapsulates DTLS application data.

26 The Joiner Router sees this KEK TLV, transmits the encapsulated DTLS message to the
27 Joiner, delays for DELAY_JOIN_ENT, and then follows that with a JOIN_ENT.ntf message
28 which is MAC-encrypted with the KEK containing the master network key.

29 The Network Credentials that are transported this way include:

- 30 • Network Master Key
- 31 • Network Mesh-Local Prefix

32 Figure 8-7 illustrates the Joiner Entrust data flow.

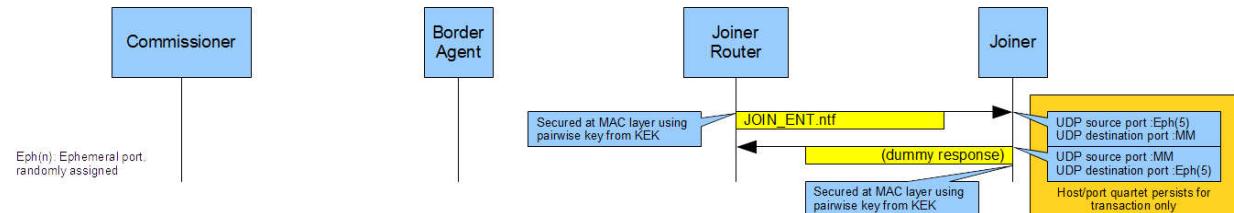


Figure 8-7. Joiner Entrust Data Flow

8.4.5.2 Joiner Provisioning

- 4 The Commissioner is provided with a mechanism to securely exchange subsequent application-specific provisioning data to the new node.
- 6 If the Joiner appealed for a specific Commissioning application and the application matched, then after the Joiner Entrust, the Joiner Session remains open for vendor-specific provisioning, and is only closed when the vendor-specific provisioning protocol specifies.
- 9 In all other cases, the Joiner Session is closed immediately after Joiner Finalization.

8.4.5.3 Joiner Session Close

- 11 In all cases, when the Joiner Session is closed, it MUST be closed using the standard DTLS-Alert mechanism, which is a round trip message exchange.
- 13 Figure 8-8 illustrates the Joiner Session closure data flow.

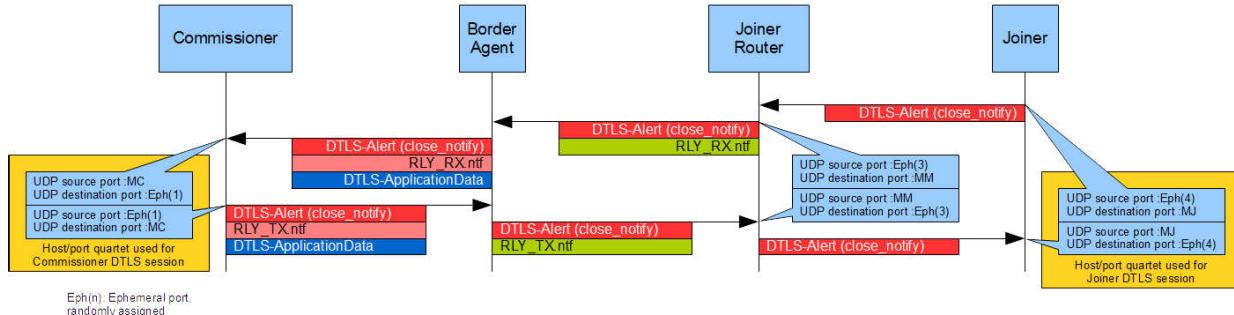


Figure 8-8. Joiner Session Closure Data Flow

8.4.6 Out-of-band Commissioning

- 17 Thread supports out-of-band commissioning. The Commissioner may extract the Network Credentials using MGMT_GET commands, and insert them securely into another device using a proprietary protocol over any convenient interface, such as Wi-Fi, BLE (Bluetooth Low Energy), or USB (Universal Serial Bus).
- 21 In practice an implementation SHOULD use the capabilities available in MeshCoP so that out-of-band commissioning can be as secure as the in-band method. The following subsections provide advisories on how to secure out-of-band commissioning and appropriate commands.

8.4.6.1 First Border Agent Commissioning

- 25 For a first device that has both a Wi-Fi/Ethernet and Thread interface, one way to leverage standard Thread Commissioning would be to create an interim Thread Network, or a fake

1 Thread Network only visible via discovery (DNS-SD or Discovery Messages) using the
2 following suggested default **Uncommissioned Parameters** easily obtained from the label:

3 Network name = "Vendor device name" or "Something on label"

4 Extended PAN ID = Device EUI-64 from label

5 Commissioning Credential = PSKd or Joining Device Credential from label

6 PSKc = hardened (PSKd)

7 **Note: The PSKc is typically user-selected, independent and unrelated to the PSKd
8 with the exception of a new device, which would derive a default PSKc from the
9 PSKd on its label.**

10 It is assumed the user would connect the device to their Wi-Fi access point using some out-
11 of-scope method, or connect directly to the new device through Wi-Fi via a SoftAP
12 (Software-enabled Access Point). Then, given the default Uncommissioned Parameters
13 above, the new device would advertise its existence as a Thread Border Agent using DNS-
14 SD. A standard Thread Commissioner would establish a DTLS-ECJPAKE Commissioner
15 Session to the device, and have access to all the MeshCoP management commands.

16 The Network Form Request command would be sent with the user-selected Network Name,
17 PSKc, and optional channel value.

18 **8.4.6.2 Secure Out-of-band Commissioning**

19 To perform secure out-of-band Commissioning of a new device using MeshCoP, a DTLS
20 Commissioner Session needs to be established over some interface such as Wi-Fi, BLE, USB,
21 or other using the default Uncommissioned Parameters. The assumption for out-of-band
22 commissioning is that the Network Credentials are known: Network Master Key, Network
23 Name, and Extended PAN ID. The desired network based on the Network Name and
24 Extended PAN ID can be found by sending a Network Scan Request command to the new
25 Thread Device. The only other Network Credential that needs to be known is the Network
26 Master Key. The Network Credentials can then be securely inserted into the new Thread
27 Device using a Network Join Request command.

28 **8.4.6.3 Out-of-band Network Commands**

29 This section defines a set of administrative commands which would generally be needed to
30 initiate a Thread Device to start functioning on a Thread Network or to leave the Thread
31 Network.

32 **8.4.6.3.1 Network Form Request**

33 This administrative command will initiate a Thread Device to form a new Thread Network.

34 Required parameters

35 Network Name

36 Network Master Key

37 PSKc

38 [Channel]

39 Response

40 Success or failure

1 **8.4.6.3.2 Network Scan Request**

2 This administrative command will initiate a Thread Device to perform Network Discovery in
3 attempt to find the appropriate Thread Network.

4 **8.4.6.3.3 Network Join Request**

5 This administrative command will initiate a Thread Device to attempt to out-of-band join a
6 device to the given Thread Network.

7 Required parameters

8 Network Master Key

9 Network Name

10 Extended PAN ID

11 Response

12 Success or failure

13 **8.4.6.3.4 Network Leave Request**

14 This administrative command will request that a device leave the Thread Network it is
15 currently active on, removing the Network Credentials from its non-volatile storage, and
16 resetting the device to an uncommissioned state.

17 Required parameters

18 (none)

19 Response

20 Success or failure

1 8.4.7 Diagrams

2 8.4.7.1 Topology

3 8.4.7.1.1 Expanded Example

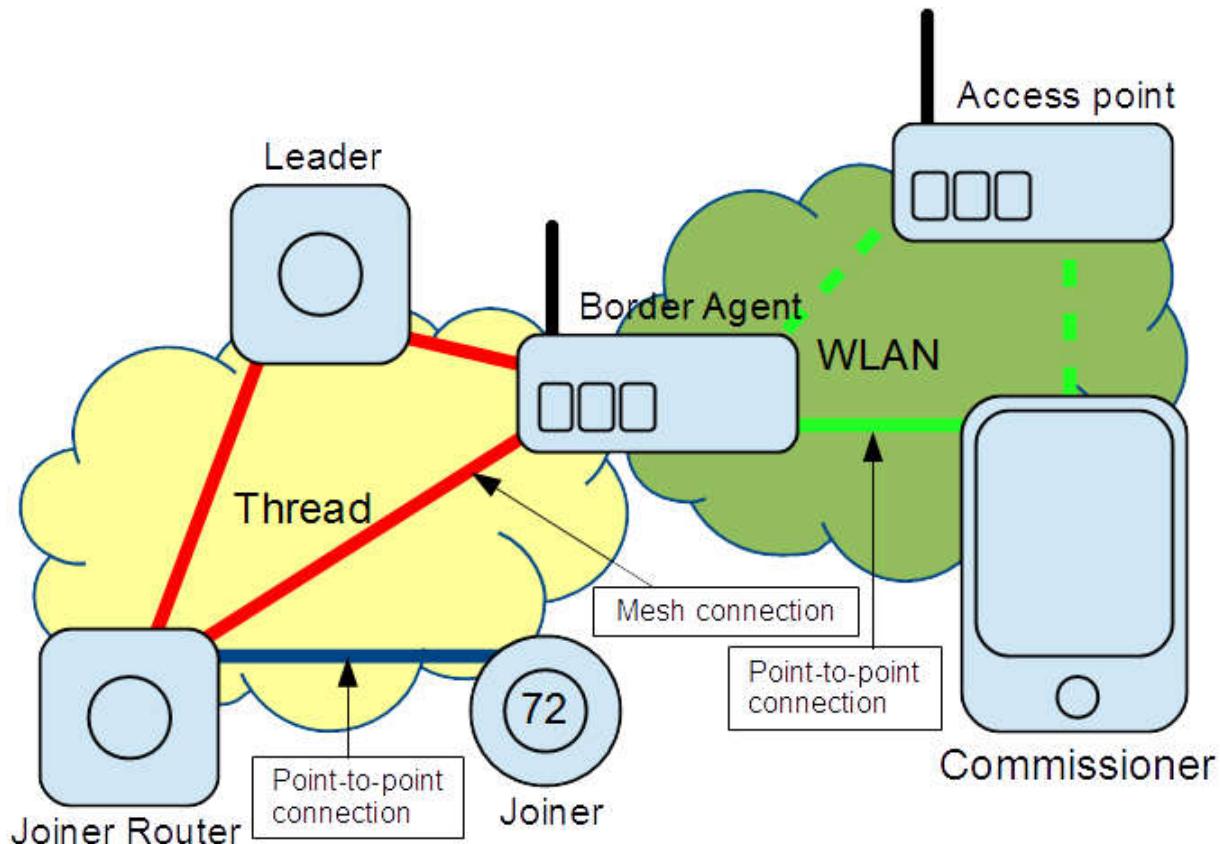
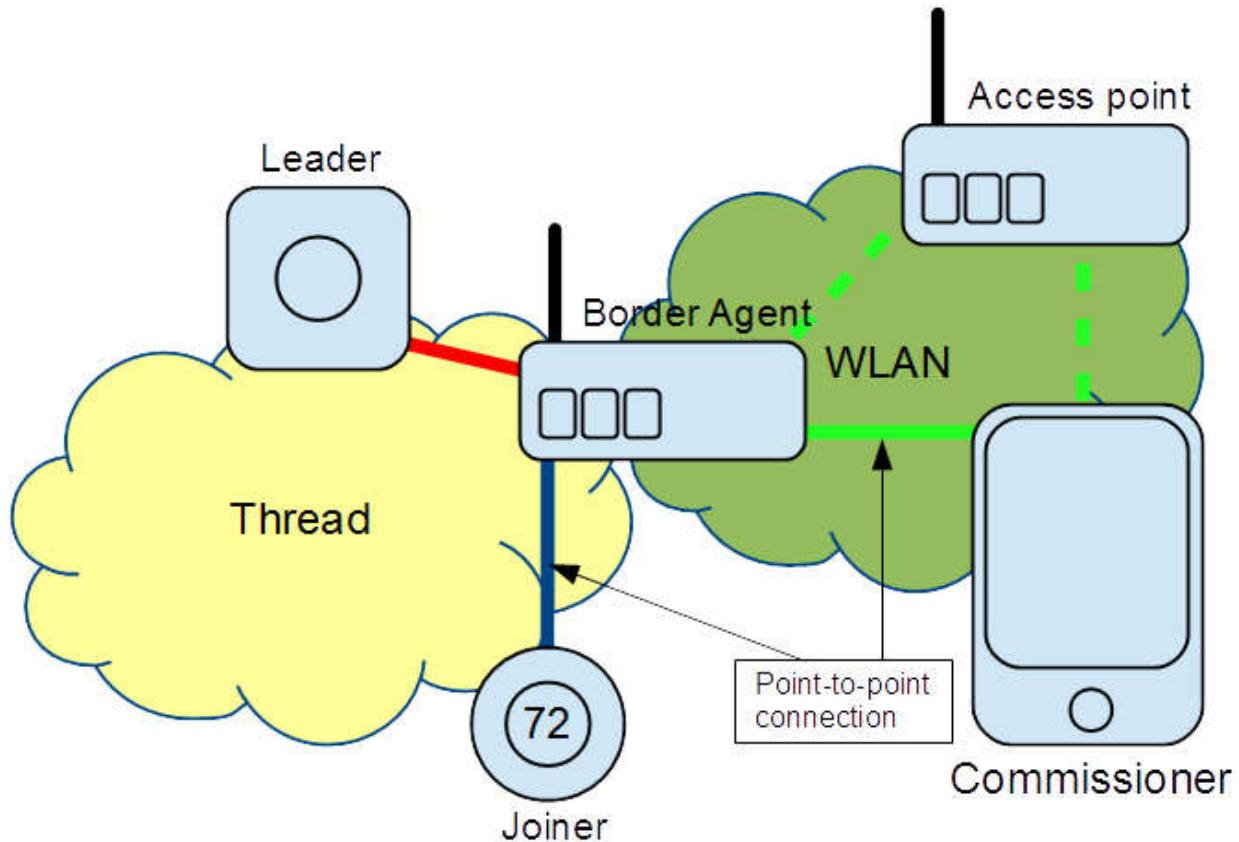


Figure 8-9. Expanded Commissioning Topology

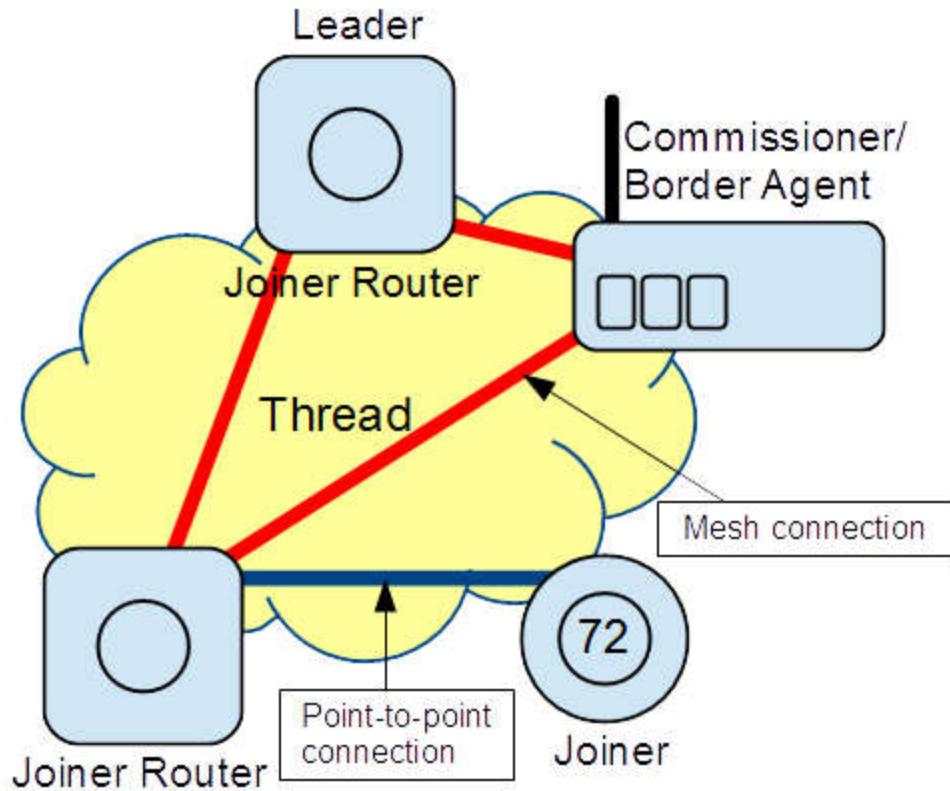
1 **8.4.7.1.2 Collapsed Case: Border Agent is Joiner Router**



2
3

Figure 8-10. Collapsed Case Topology–Border Agent is Joiner Router

1 **8.4.7.1.3 Collapsed Case: Commissioner is Border Agent**

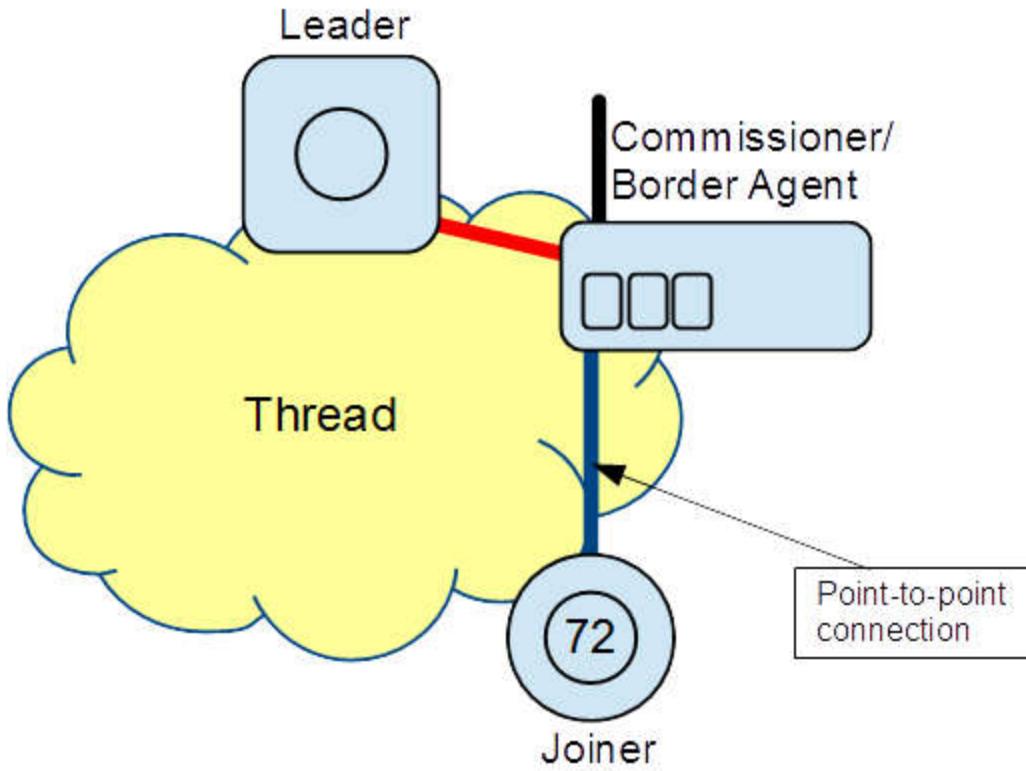


2

3

Figure 8-11. Collapsed Case Topology—Commissioner is Border Agent

1 **8.4.7.1.4 Collapsed Case: Commissioner is Joiner Router**



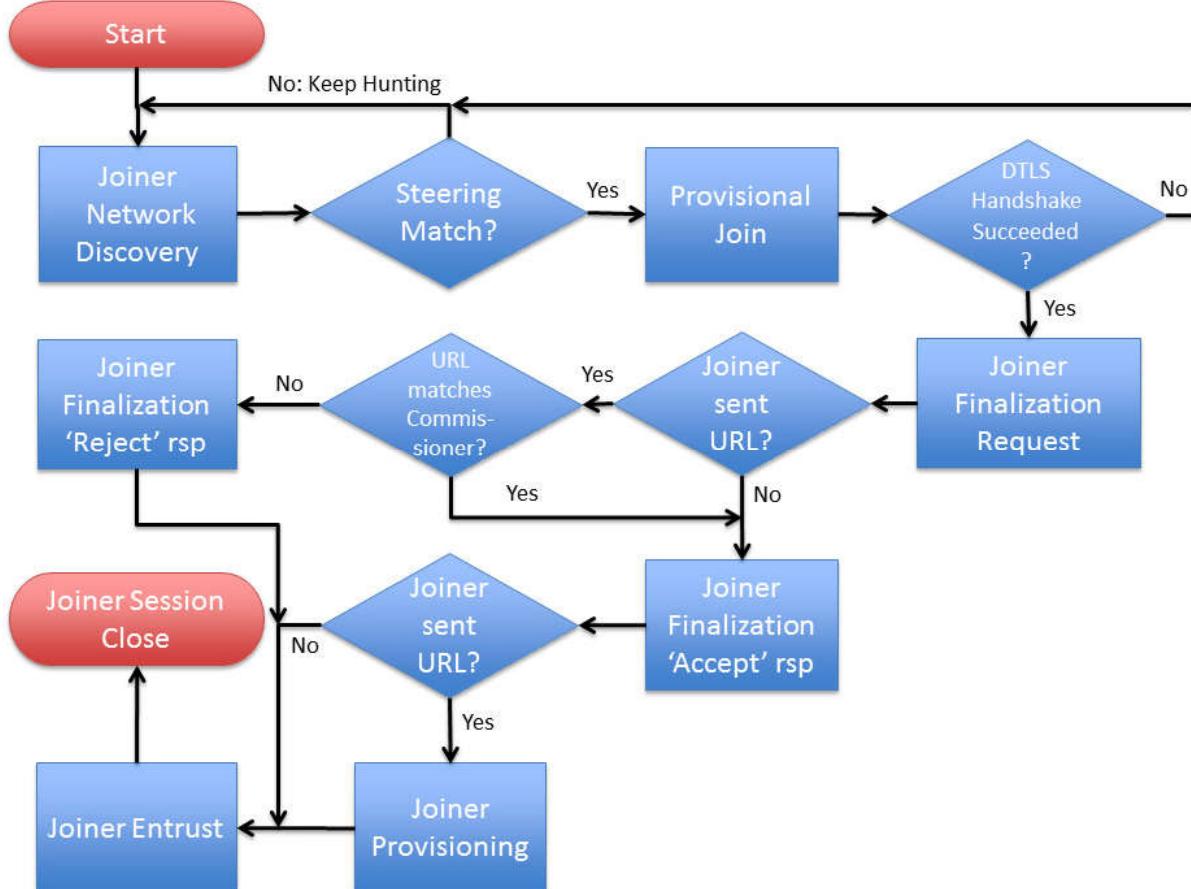
2

3

Figure 8-12. Collapsed Case Topology—Commissioner is Joiner Router

1 8.4.7.2 Flow Charts

2 8.4.7.2.1 Joiner State Machine Flow Chart

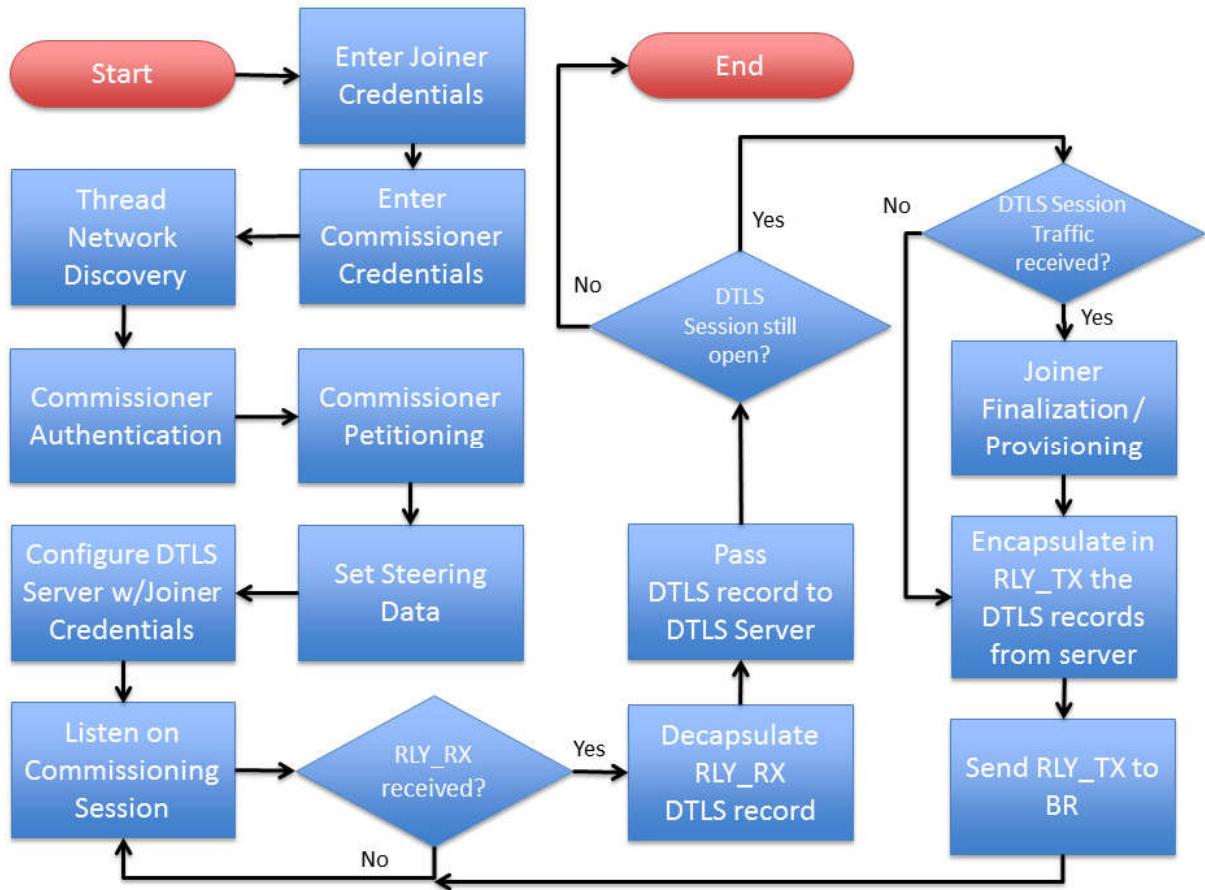


3

4

Figure 8-13. Joiner State Machine Flow Chart

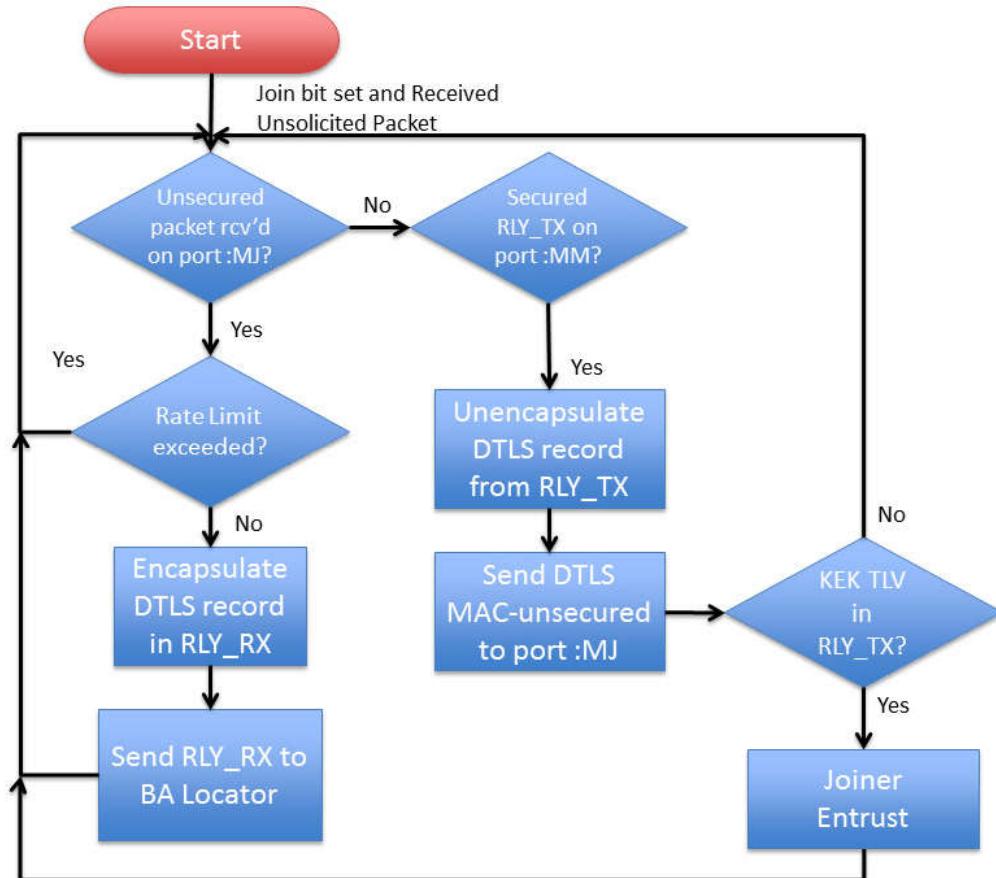
1 8.4.7.2.2 Commissioner State Machine Flow Chart



2
3

Figure 8-14. Commissioner State Machine Flow Chart

1 8.4.7.2.3 Joiner Router State Machine Flow Chart



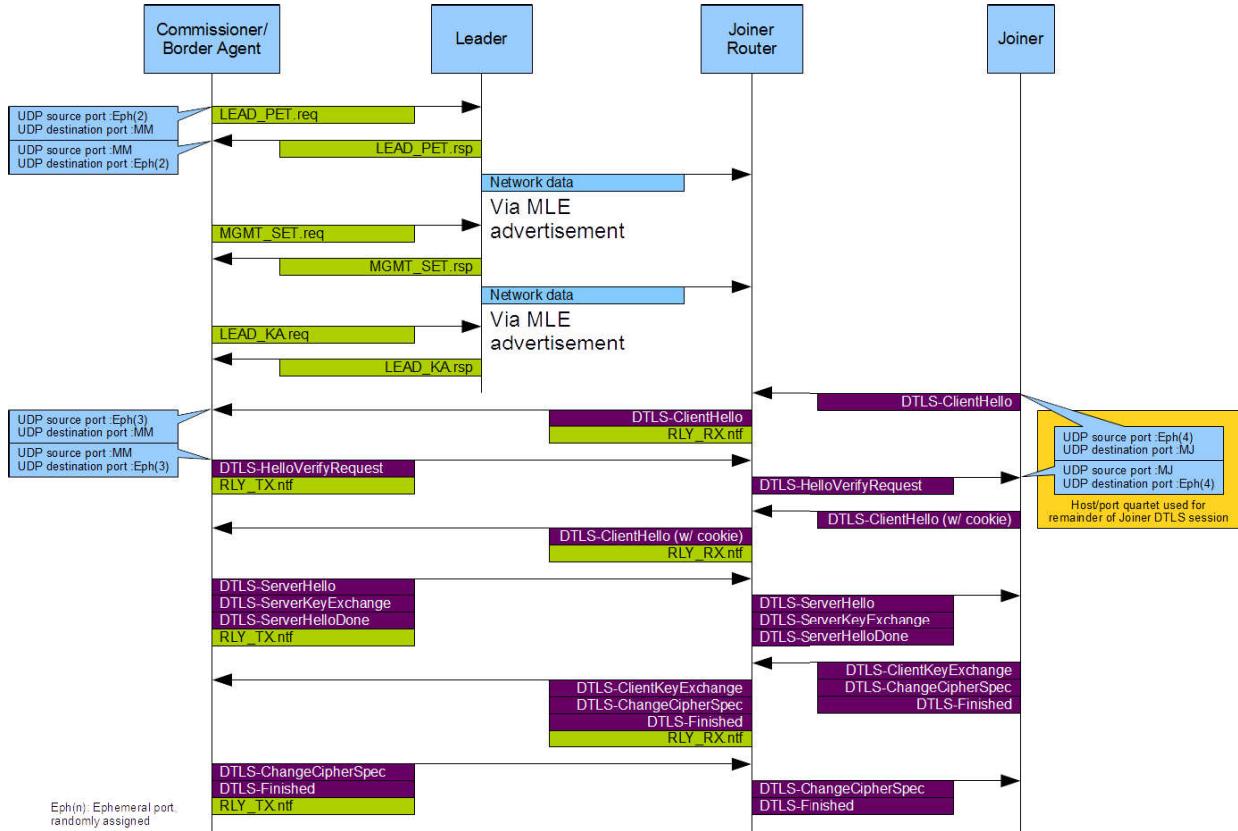
2

3

Figure 8-15. Joiner Router State Machine Flow Chart

1 8.4.7.3 Data Flows

2 8.4.7.3.1 Collapsed Thread-only Data Flow



4 **Figure 8-16. Petitioning and Authentication on Thread-only Collapsed Case**

5 8.4.7.4 Port Usage Diagrams

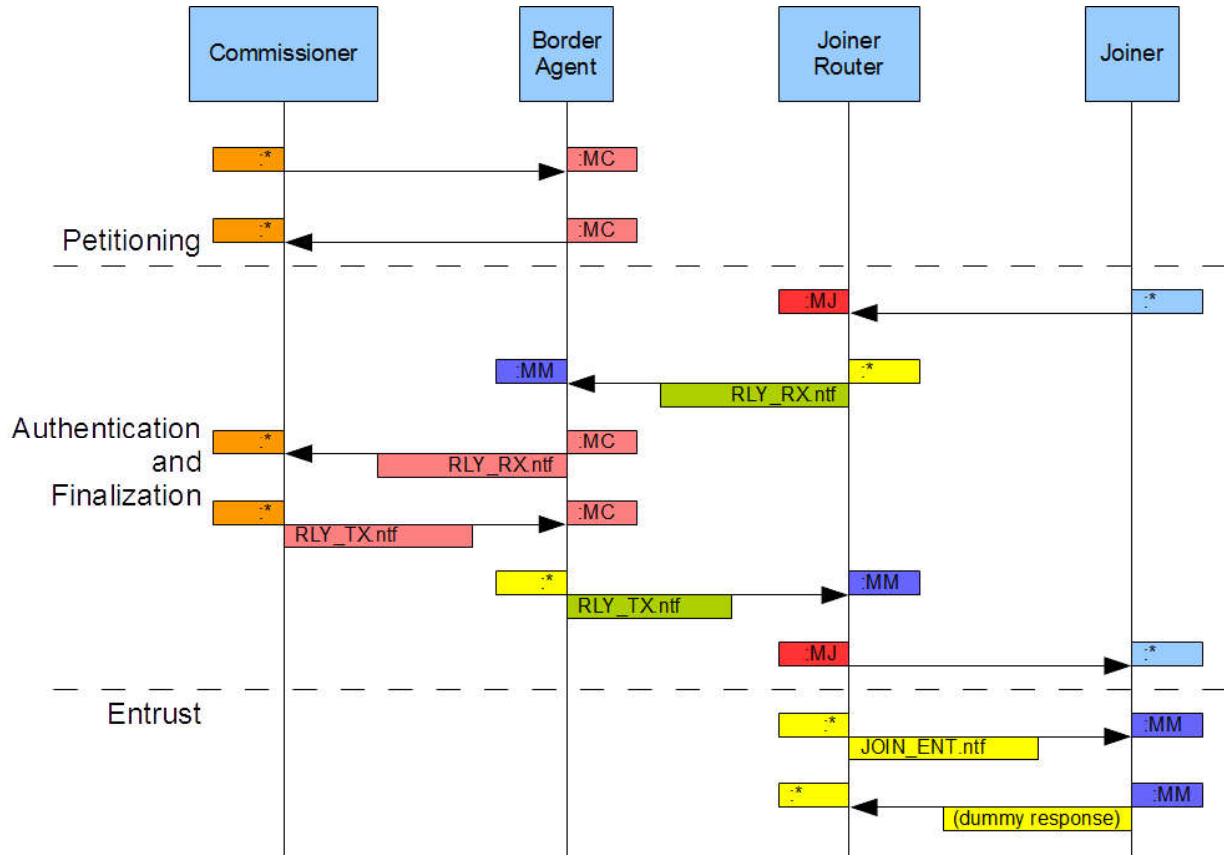
6 These diagrams help clarify which UDP ports are used for the various inter-device message
7 exchanges within MeshCoP. The terminology used is:

8 :SRC_PORT → :DST_PORT

9 where:

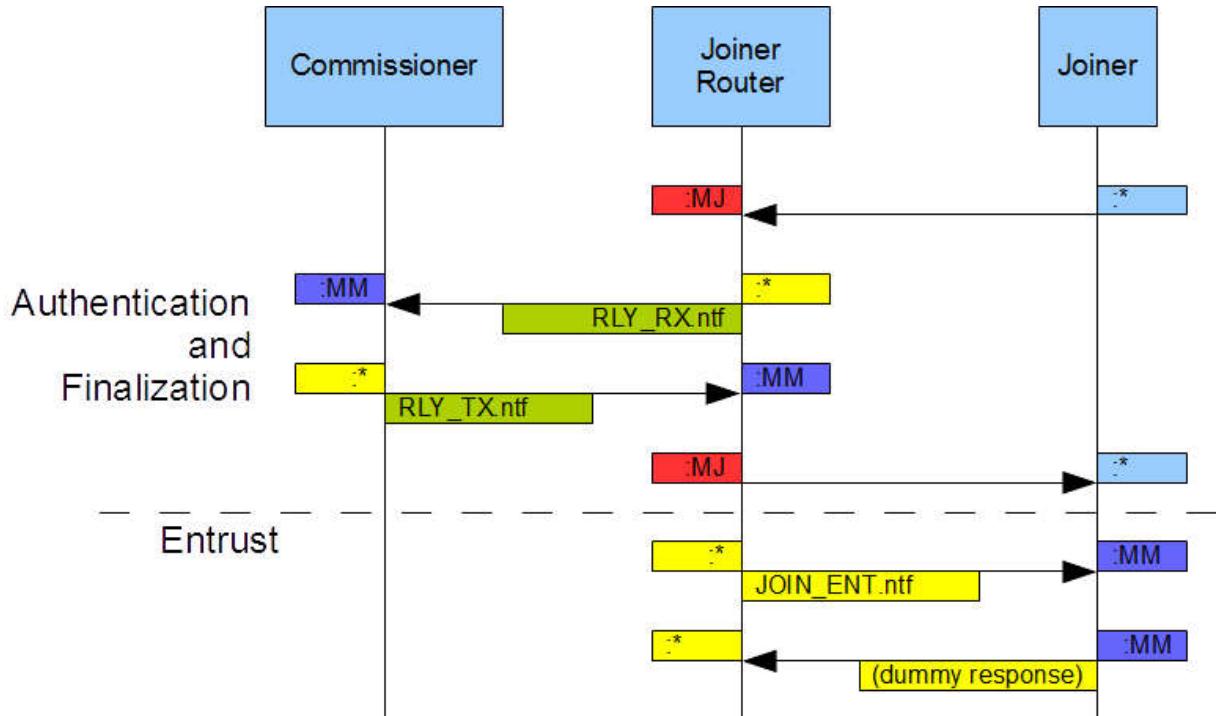
- 10 • :* means an ephemeral client source port
- 11 • :Mx is a dynamically allocated yet stable server port as defined in Section 8.12,
12 **IANA Considerations**.

1 8.4.7.4.1 External Commissioner Port Usage



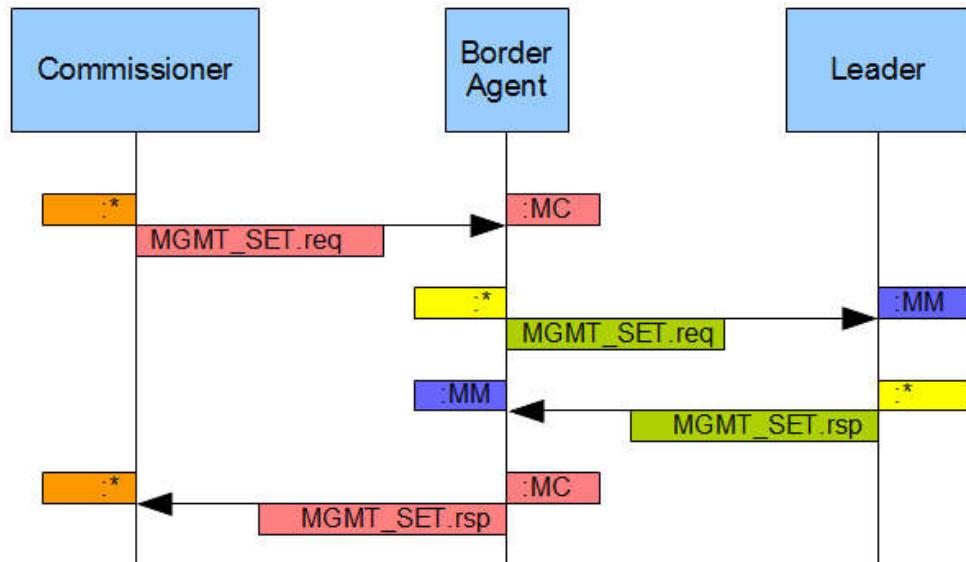
2
3 **Figure 8-17. External Commissioner Port Usage**

1 8.4.7.4.2 On-Mesh Commissioner Port Usage



2 3 **Figure 8-18. On-Mesh Commissioner Port Usage**

4 8.4.7.4.3 Commissioner Management Port Usage



5 6 **Figure 8-19. Commissioner Management Port Usage**

8.5 Message Format

MeshCoP commands use the Thread Management Framework (TMF) message format and notation described in Section 10.3, URIs and Notation, in Chapter 10, Management.

MeshCoP commands may be destined for:

- Leader (L)
- Border Agent (BA)
- Joiner Router (JR)
- Joiner (J)
- Commissioner (C)

The above abbreviations for destinations are included in angle brackets in the CoAP request URI definitions in this chapter—for example <L>, <C> or <JR>.

The MeshCoP commands can occur in the following four scopes:

- Commissioner scope: Between Commissioner to Border Agent only
- Commissioner and Thread Network scope: Between Commissioner and Border Agent and in Thread Network
- Thread Network scope: In Thread Network only
- Joiner scope: Between Joiner and Joiner Router

8.6 Commissioner Scope TMF Messages

This section covers commands that are sent between the Commissioner and its Border Agent over the Commissioning Session.

8.6.1 Petitioning Commands

Petitioning commands are used in the arbitration process whereby a Commissioner Candidate may become a Commissioner. Petitioning commands are always sent over a secured Commissioning Session.

8.6.1.1 COMM_PET.req – Commissioner Petition Request

This command is sent from the Commissioner Candidate to the Border Agent via the secured Commissioning Session. This command both identifies the Commissioner Candidate with the Border Agent and tells the Border Agent to petition the Leader on behalf of the sending Commissioner Candidate.

CoAP Request URI

coaps:// [<BA>] :MC/c/cp

Transaction Pattern

Req+Rsp_Separate (request part)

CoAP Payload

Commissioner ID TLV

- 1 Commissioner ID (inside the TLV)
- 2 A human-readable identifier for the Commissioner Candidate. This string will be advertised to other clients if the device becomes the active Commissioner.

8.6.1.2 COMM_PET.rsp – Commissioner Petition Response

It is sent as a separate response on receipt of LEAD_PET.rsp from the Leader, if the Border Agent is not also the Leader, or a timeout (TIMEOUT_LEAD_PET) event, if the Border Agent is also the Leader.

CoAP Response Code

2.04 Changed

Transaction Pattern

Req+Rsp_Separate (response part)

CoAP Payload

State TLV

[Commissioner ID TLV]

[Commissioner Session ID TLV]

State TLV

Result of the petitioning request. The Border Agent just forwards the value received by LEAD_PET.rsp. 'Accept' means the device is now the active Commissioner and a session ID will be included. 'Reject' means another device is active, the current device has been resigned, or no response was received from the Leader after MAX_RETRANSMIT attempts. If another device is active, the Commissioner ID for that device will be included.

Commissioner ID TLV

Human-readable identifier for the active Commissioner, if applicable. Otherwise elided.

Commissioner Session ID TLV

This field is only present when the State value is 'Accept', in which case the device has been elected as Commissioner. All subsequent keep-alives must use this session ID.

8.6.1.3 COMM_KA.req – Commissioner Keep Alive Request

COMM_KA.req is sent from the active Commissioner to its Border Agent, notifying the Border Agent that the active Commissioner is still alive. Commissioner Candidates use COMM_PET.req as a keep-alive, because they are continually attempting to petition to be Commissioner. A Commissioner that does not send this message in over TIMEOUT_COMM_PET expires and is rejected.

CoAP Request URI

coaps://[<BA>]:MC/c/ca

Transaction Pattern

Req+Rsp_Separate (request part)

- 1 CoAP Payload
- 2 State TLV
- 3 Commissioner Session ID TLV
- 4 State TLV
- 5 Set to 'Reject' to resign as the Commissioner. Otherwise, set to 'Accept'.
- 6 Commissioner Session ID TLV
- 7 The Leader-allocated session ID for this active Commissioner.

8.6.1.4 COMM_KA.rsp – Commissioner Keep Alive Response

- 10 COMM_KA.rsp is sent by the Border Agent to the Commissioner in response to a
- 11 COMM_KA.req. It is sent as a separate response on receipt of LEAD_KA.rsp from the Leader
- 12 or timeout.
- 13 CoAP Response Code
- 14 2.04 Changed
- 15 Transaction Pattern
- 16 Req+Rsp_Separate (response part)
- 17 CoAP Payload
- 18 State TLV
- 19 State TLV
- 20 Set to 'Reject' to reject the Commissioner. Otherwise, set to 'Accept'.

8.6.2 Proxy Commands

- 22 The UDP proxy commands allow secure, two-way communication of Thread Management
- 23 Commands between the Commissioner and any Thread Device in the Thread Network via
- 24 the Border Agent for diagnostics, network administration and general application purposes.
- 25 The Border Agent accepts all UDP traffic to the Commissioner ALOC corresponding to a
- 26 particular Commissioner Session ID. Such Commissioner Anycast traffic is encapsulated and
- 27 forwarded to the Commissioner via UDP_RX.ntf commands over the Commissioner Session.
- 28 Because traffic to the Commissioner is sent to a special Anycast Address, there are no
- 29 conflicts with ports the Border Agent has allocated for its own use. Likewise, the
- 30 Commissioner can send arbitrary UDP traffic to any Thread Device on the Thread Network
- 31 by using its Commissioner Anycast as the source address, encapsulating it in a UDP_TX.ntf
- 32 command, and forwarding to the Border Agent over the Commissioning Session.

8.6.2.1 UDP_RX.ntf – Proxy Receive UDP Notification

- 34 The Proxy Receive UDP notification message informs the Commissioner of an incoming UDP
- 35 datagram from a Thread Device on the Thread Network via the Border Agent proxy service.
- 36 The Border Agent, upon receipt of a UDP datagram to the appropriate Commissioner ALOC,
- 37 will encapsulate the datagram and relay it to the Commissioner using the secure
- 38 Commissioning Session.

1 CoAP Request URI
2 coaps://[<C>]:<CSSP>/c/ur
3 Transaction Pattern
4 Ntf/Qry/Ans_NON
5 CoAP Payload
6 IPv6 Address TLV
7 UDP Encapsulation TLV
8 IPv6 Address TLV
9 Source address of the UDP datagram from the Thread Network
10 UDP Encapsulation TLV
11 Encapsulated UDP datagram originating from the Thread Device sent by the Border
12 Agent proxy to the Commissioner.

13 **8.6.2.2 UDP_TX.ntf – Proxy Transmit UDP Notification**

14 The Proxy Transmit UDP notification message allows the Commissioner to send a UDP
15 datagram using the secure Commissioning Session into the Thread Network via the Border
16 Agent proxy service. The Border Agent sends the decapsulated UDP datagram into the
17 Thread Network using the appropriate Commissioner ALOC as the source address.

18 CoAP Request URI
19 coaps://[<BA>]:MC/c/ut
20 Transaction Pattern
21 Ntf/Qry/Ans_NON
22 CoAP Payload
23 IPv6 Address TLV
24 UDP Encapsulation TLV
25 IPv6 Address TLV
26 Destination address of the UDP datagram to the Thread Network
27 UDP Encapsulation TLV
28 Encapsulated UDP datagram destined for the Thread Device sent by the Commissioner
29 to the Border Agent proxy.

30 **8.7 Commissioner and Thread Network 31 Scope Commands**

32 This section covers commands that are both sent over the Commissioning Session and over
33 the Thread Network. These commands are sent between the Commissioner and Border
34 Agent, and also between the Border Agent and either the Leader or another Thread Device.

1 **8.7.1 Relay Commands**

2 The DTLS Relay Commands encapsulate DTLS records between the Commissioner and the
3 Joiner, and relay them over the Thread Network and non-Thread Network(s) via relay
4 Thread Devices. The DTLS Relay Commands are always sent or received by the
5 Commissioner either through the secure Commissioning Session and the mesh network or,
6 in the case of a Native Commissioner, directly across the mesh network to a Joiner Router.
7 The DTLS Relay Commands MUST NOT be sent to or received from a Joiner.

8 The DTLS Relay Commands carry DTLS records across the relay points with auxiliary data
9 that constitutes a return path for the originator as the Commissioner may not have any
10 other mechanism to discover addresses on the mesh. The Commissioner uses the return
11 path to return messages via the relay back to the originator.

12 The DTLS Relay Commands require a Border Agent with a Commissioning Session to be
13 known to the network. The Border Agent address to forward DTLS Relay Commands to is
14 determined from the Thread Network Data update when a Commissioner is assigned to the
15 network.

16 **8.7.1.1 RLY_RX.ntf – DTLS Relay Receive Notification**

17 RLY_RX.ntf relays a DTLS record towards the Commissioner, capturing the return path for
18 the Commissioner to use for a DTLS Relay Transmit notification message sent in reply back
19 to the Joiner.

20 Further propagation of DTLS Relay Receive notification messages MUST be rate-limited to
21 JOINER_RELAY_RATE_LIMIT to prevent DoS attacks, because these originate from
22 unauthenticated devices on an unsecured link.

23 CoAP Request URI

24 Joiner Router to Border Agent: `coap://[<BA>]:MM/c/rx`

25 Border Agent to Commissioner: `coaps://[<C>]:MC/c/rx`

26 Transaction Pattern

27 Ntf/Qry/Ans_NON

28 CoAP Payload

29 Joiner DTLS Encapsulation TLV

30 Joiner UDP Port TLV

31 Joiner IID TLV

32 Joiner Router Locator TLV

33 Joiner DTLS Encapsulation TLV

34 Contains the encapsulated DTLS record being sent to the Joiner from the Commissioner.

35 Joiner UDP Port TLV

36 The UDP port of the Joiner.

37 Joiner IID TLV

38 The IID (Interface Identifier) based on the MAC Extended Address of the originating
39 Joiner (see Section 8.10.2.5, **Joiner IID TLV (19)**).

- 1 Joiner Router Locator TLV
- 2 The RLOC16 of the Joiner Router.

8.7.1.2 RLY_TX.ntf – DTLS Relay Transmit Notification

4 RLY_TX.ntf relays a DTLS record back toward the Joiner. Typically, the addresses the
5 Commissioner uses are captured from a previously received RLY_RX.ntf message.

- 6 CoAP Request URI
 - 7 Commissioner to Border Agent: `coaps://[<BA>]:MC/c/tx`
 - 8 Border Agent to Joiner Router: `coap://[<JR>]:MM/c/tx`

9 Transaction Pattern

10 Ntf/Qry/Ans_NON

11 CoAP Payload

12 Joiner DTLS Encapsulation TLV

13 Joiner UDP Port TLV

14 Joiner IID TLV

15 Joiner Router Locator TLV

16 [Joiner Router KEK TLV]

17 Joiner DTLS Encapsulation TLV

18 Encapsulated DTLS record being sent to the Commissioner from the Joiner.

19 Joiner UDP Port TLV

20 The UDP port of the Joiner.

21 Joiner IID TLV

22 The IID based on the MAC Extended Address of the destination Joiner (see Section
23 8.10.2.5, **Joiner IID TLV (19)**). Used for reconstructing the link-local address of the
24 Joiner. The Commissioner inserts the field, and the Joiner Router uses it to send the de-
25 encapsulated UDP datagram to the Joiner.

26 Joiner Router Locator TLV

27 The RLOC16 of the Joiner Router. The Commissioner inserts this field, and the Border
28 Agent uses it to relay the UDP datagram across the mesh.

29 Joiner Router KEK TLV

30 When this TLV is included in the relay message, the Joiner Router is signaled that this
31 Joiner has been authenticated by the Commissioner, and is ready to be entrusted. The
32 Joiner Router relays the encapsulated DTLS message to the Joiner, and then sends the
33 JOIN_ENT.ntf message encrypted at the MAC level with the KEK to give the Joiner the
34 Network Credentials.

8.7.2 Network Management Commands

36 Network management commands allow nodes to read and set network attributes.

1 **Note: MGMT_SET.req and MGMT_SET.rsp are deprecated. Section 8.7.2.3,**
2 **MGMT_SET.req – Set Management Data Request, and Section 8.7.2.4,**
3 **MGMT_SET.rsp – Set Management Data Response, and other references to**
4 **MGMT_SET.req and MGMT_SET.rsp should be ignored.**

5 **8.7.2.1 MGMT_GET.req – Get Management Data** 6 **Request**

7 MGMT_GET.req allows the Commissioner or a Thread Device to query the current value of
8 attributes within another Thread Device.

9 CoAP Request URI

10 Commissioner to Border Agent: coaps://[<BA>]:MC/c/mg

11 Border Agent or other Device to Device: coap://[<destination>]:MM/c/mg

12 Transaction Pattern

13 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)

14 CoAP Payload

15 Get TLV

16 Get TLV

17 List of 8-bit type identifiers, defined by section 8.10.1, **Network Management TLVs**,
18 for which values are requested from the destination node.

19 **8.7.2.2 MGMT_GET.rsp – Get Management Data** 20 **Response**

21 MGMT_GET.rsp contains the attribute values sent in response to a MGMT_GET.req.

22 CoAP Response Code

23 2.04 Changed

24 Transaction Pattern

25 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)

26 CoAP Payload

27 Parameter TLVs

28 Parameter TLVs

29 A series of one or more parameter TLVs containing parameters and values that were
30 requested by the MGMT_GET.req message.

31 **8.7.2.3 MGMT_SET.req – Set Management Data Request**

32 MGMT_SET.req is sent from the Commissioner to the Leader. The response to this, the
33 command MGMT_SET.rsp, is sent from the Leader to the Commissioner.

34 CoAP Request URI

35 coap://[<L>]:MM/c/ms

36 Transaction Pattern

37 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)

- 1 CoAP Payload
- 2 Commissioner Session ID TLV
- 3 Parameter TLVs
- 4 Commissioner Session ID TLV
 - 5 This TLV is included to allow the Leader to filter out MGMT_SET.req messages that are
 - 6 sent from an older Commissioner Session. It is not used to set a value on the Leader.
- 7 Parameter TLVs
- 8 A series of one or more parameter TLVs containing parameters and values to set.

8.7.2.4 MGMT_SET.rsp – Set Management Data Response

- 11 MGMT_SET.rsp is sent from the Leader to the Commissioner in response to MGMT_SET.req.
- 12 Communicates status of attempt to set network parameters.
- 13 CoAP Response Code
 - 14 2.04 Changed
- 15 Transaction Pattern
 - 16 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)
- 17 CoAP Payload
- 18 State TLV
- 19 State TLV
 - 20 Set to 'Accept' or 'Reject' based on whether the Leader has accepted the data. If set to
 - 21 'Reject', no data has been changed.

8.7.3 Updating the Commissioner Dataset

- 23 The Leader is responsible for disseminating the Commissioner Dataset throughout its Thread
- 24 Network Partition.

8.7.3.1 Updates from a Commissioner

- 26 An active Commissioner sends a MGMT_COMMISISONER_GET.req message to retrieve one
- 27 or more Commissioner Dataset values from the Leader. The payload includes a Get TLV that
- 28 includes one or more TLV types associated with the requested Commissioner Dataset
- 29 parameters.
- 30 An active Commissioner sends a MGMT_COMMISISONER_SET.req message to change one
- 31 or more Commissioner Dataset values in the Leader. The payload MUST include the
- 32 Commissioner Session ID and the Commissioner Dataset values encoded in appropriate
- 33 TLVs.

8.7.3.2 MGMT_COMMISISONER_GET.req – Get Commissioner Dataset Request

- 36 While a Commissioner is active, the Commissioner gets the Commissioner Dataset
- 37 parameters by sending a MGMT_COMMISISONER_GET.req to the Leader.

- 1 CoAP Request URI
- 2 Commissioner to Border Agent: coaps://[<BA>]:MC/c/cg
- 3 Border Agent to Leader: coap://[<L>]:MM/c/cg
- 4 Transaction Pattern
- 5 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)
- 6 CoAP Payload
- 7 [Get TLV]
- 8 Get TLV
- 9 Included when requesting a subset of the Commissioner Dataset and indicates one or
- 10 more TLV types associated with the requested Commissioner Dataset parameters. Not
- 11 included when requesting the entire Commissioner Dataset.
- 12 A Thread Device MUST ignore any TLVs other than the Get TLV included in the CoAP
- 13 payload.
- 14 A Thread Device MUST ignore any TLV types included in the Get TLV that are not included in
- 15 the Commissioner Dataset.

8.7.3.3 MGMT_COMMISISONER_GET.rsp – Get Commissioner Dataset Response

- 18 The MGMT_COMMISISONER_GET.rsp message is a response that contains the requested
- 19 information in the CoAP payload.
- 20 CoAP Response Code
- 21 2.04 Changed
- 22 Transaction Pattern
- 23 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)
- 24 CoAP Payload
- 25 Commissioner Dataset TLVs
- 26 Commissioner Dataset TLVs
- 27 Encodes values of requested Commissioner Dataset parameters. If the
- 28 MGMT_COMMISISONER_GET.req did not contain a Get TLV, the
- 29 MGMT_COMMISISONER_GET.rsp message MUST include the entire Commissioner
- 30 Dataset.

8.7.3.4 MGMT_COMMISISONER_SET.req – Set Commissioner Dataset Request

- 33 While a Commissioner is active, the Commissioner sets Commissioner Dataset values by
- 34 sending a MGMT_COMMISISONER_SET.req to the Leader.
- 35 CoAP Request URI
- 36 Commissioner to Border Agent: coaps://[<BA>]:MC/c/cs
- 37 Border Agent to Leader: coap://[<L>]:MM/c/cs
- 38 Transaction Pattern

- 1 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)
- 2 CoAP Payload
- 3 Commissioner Session ID TLV
- 4 Commissioner Dataset TLVs
- 5 Commissioner Session ID TLV
- 6 Includes the Commissioner Session ID provided by the Leader in the COMM_PET.rsp message.
- 7
- 8 Commissioner Dataset TLVs
- 9 Encodes all other Commissioner Dataset parameters. A Commissioner MUST NOT set the
- 10 following values:
 - 11 • Commissioner Session ID
 - 12 • Border Agent Locator

8.7.3.5 MGMT_COMMISISONER_SET.rsp – Set Commissioner Dataset Response

- 15 The MGMT_COMMISISONER_SET.rsp message contains a State TLV indicating the result of
- 16 the MGMT_COMMISISONER_SET.req.
- 17 CoAP Response Code
- 18 2.04 Changed
- 19 Transaction Pattern
- 20 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)
- 21 CoAP Payload
- 22 State TLV
- 23 State TLV
- 24 Encodes the result of executing the MGMT_COMMISISONER_SET.req message. 'Accept'
- 25 indicates that all TLVs in the MGMT_COMMISISONER_SET.req message were accepted.
- 26 'Reject' indicates that none of the TLVs in the MGMT_COMMISISONER_SET.req message
- 27 were accepted.

8.7.3.6 TLV Encoding

- 29 The Commissioner Dataset parameters are encoded using the following MeshCoP TLVs when
- 30 communicated to other devices.

Commissioner Dataset Parameter	MeshCoP TLV
Border Agent Locator	Border Agent Locator TLV
Commissioner Session ID	Commissioner Session ID TLV
Steering Data	Steering Data TLV

8.7.3.7 Leader Behavior

When the Leader establishes an active Commissioner, the Leader MUST perform the following actions:

1. Set the Border Agent Locator to the RLOC16 of the Border Agent that is proxying the Commissioner's messages.
2. Increment the Commissioner Session ID.

When the Leader resigns an active Commissioner, the Leader MUST perform the following actions:

1. Remove the Border Agent Locator from the Commissioner Dataset.
2. Remove the Steering Data from the Commissioner Dataset.
3. Increment the Commissioner Session ID.

The Leader receives new Commissioner Dataset values from the active Commissioner in MGMT_COMMISISONER_SET.req messages. On receiving a valid MGMT_COMMISISONER_SET.req message from the active Commissioner, the Leader MUST update its Commissioner Dataset with the received values. A Leader MUST store all TLV types, even those types that it does not understand.

The Leader MUST reject any MGMT_COMMISISONER_SET.req messages that does not include a Commissioner Session ID that matches the local value. If the Leader rejects the MGMT_COMMISISONER_SET.req, the Leader MUST send a MGMT_COMMISISONER_SET.rsp message with a State TLV that indicates Reject.

If the Leader accepts the MGMT_COMMISISONER_SET.req, the Leader MUST send a MGMT_COMMISISONER_SET.rsp message with a State TLV that indicates Accept.

8.7.4 Updating the Active Operational Dataset

Any changes to the Active Operational Dataset MUST be associated with a new Active Timestamp. The Active Timestamp is used to establish priority. When two or more Active Operational Datasets are present the one with the later Active Timestamp is used.

Changes to Active Operational Dataset parameters may either be done directly, in which case they take effect immediately, or may be done using the Pending Operational Dataset, in which case they take place after a delay. The following Active Operational Dataset values affect the ability for neighboring devices to communicate:

- Channel
- Mesh-Local Prefix
- PAN ID
- Network Master Key

Changes that modify one or more of these values MUST take effect after a delay by using the Pending Operational Dataset, with a delay of at least DELAY_TIMER_MINIMAL. The Commissioner should adjust the Pending delay to allow Pending Dataset propagation based on current network size and consistency conditions if that information is available to the Commissioner. Otherwise the Commissioner should use a delay of at least DELAY_TIMER_DEFAULT if the Network Master Key does not change and of at least DELAY_TIMER_EXTENDED if the Network Master Key does change. Changes that do not modify any of these values SHOULD be done as immediate updates. Such changes MAY take effect after a delay by using the Pending Operational Dataset if a delay is needed for

1 operational reasons; it should be noted that delayed changes are less reliable for
2 parameters that do not affect connectivity.
3 The Commissioner may choose to update parts of the Active Operational Dataset for various
4 administrative reasons (such as changing the channel to avoid a noisy one). The
5 Commissioner will pass the new parameters to the Leader who in turn will disseminate the
6 Active Operational Dataset through the Leader's Thread Network Partition. Disseminating a
7 new Active Operational Dataset through a Thread Network Partition requires first providing
8 the Leader with the new Active Operational Dataset.

8.7.4.1 Updates from a Commissioner

An active Commissioner sends a MGMT_ACTIVE_SET.req message to change one or more Active Operational Dataset values in the Leader.

A MGMT_ACTIVE_SET.req message originated by a Commissioner MUST include:

1. The Commissioner Session ID provided by the Leader in the COMM_PET.rsp message.
2. An Active Timestamp value ahead of the value stored on the Leader.
3. At most one TLV for each Active Operational Dataset parameter.

A MGMT_ACTIVE_SET.req message originated by a Commissioner MUST NOT include any Active Operational Dataset parameters that affect connectivity.

The Active Timestamp SHOULD be set to the current Coordinated Universal Time (UTC) value when the Commissioner generates the Active Operational Dataset.

The Commissioner MAY set an Active Timestamp value that is ahead of UTC time if:

1. The Commissioner does not know the current UTC time.
2. The Leader has an Active Timestamp value that is ahead of the current UTC time when the Commissioner generates the new Active Operational Dataset.

In either case above, the Commissioner generates the Active Timestamp value by taking the current Active Timestamp value from the Leader and adding a random 15-bit value to the Ticks field, carrying over to the seconds field, if necessary.

If a Leader has an existing Pending Operational Dataset, the Leader MUST only accept new Pending Operational Datasets that have a newer Pending Timestamp. As a result, if the Leader has a Pending Timestamp that is set to the maximum value, the Commissioner MUST wait for the existing Delay Timer to expire before issuing a new Pending Operational Dataset.

8.7.4.2 Updates from a Thread Device

A Thread Device that has a newer Active Operational Dataset, as determined by the Active Timestamp included in Thread Network Data, MUST provide its newer Active Operational Dataset to the Leader by sending a MGMT_ACTIVE_SET.req to the Leader.

A MGMT_ACTIVE_SET.req message originated by a Thread Device MUST include the entire Active Operational Dataset as values encoded in appropriate TLVs.

A MGMT_ACTIVE_SET.req message originated by a Thread Device MUST NOT include a Commissioner Session ID.

8.7.4.3 MGMT_ACTIVE_GET.req – Get Active Operational Dataset Request

The Commissioner and Thread Devices may retrieve Active Operational Dataset values from the Leader by sending a MGMT_ACTIVE_GET.req.

- 1 CoAP Request URI
- 2 Commissioner to Border Agent: `coaps://[<BA>]:MC/c/ag`
- 3 Border Agent or other Device to Device: `coap://[<L>]:MM/c/ag`
- 4 Transaction Pattern
- 5 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)
- 6 CoAP Payload
- 7 [Get TLV]
- 8 Get TLV
- 9 Included when requesting a subset of the Active Operational Dataset and indicates one or more TLV types associated with the requested Active Operational Dataset parameter values. Not included when requesting the entire Active Operational Dataset.
- 12 A Thread Device MUST ignore any TLVs other than the Get TLV included in the CoAP payload.
- 14 A Thread Device MUST ignore any TLV types included in the Get TLV that are not included in the Active Operational Dataset.

16 **8.7.4.4 MGMT_ACTIVE_GET.rsp – Get Active Operational Dataset Response**

- 18 The Leader and other Thread Devices MUST send a MGMT_ACTIVE_GET.rsp in response to receiving a MGMT_ACTIVE_GET.req message.
- 20 CoAP Response Code
- 21 2.04 Changed
- 22 Transaction Pattern
- 23 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)
- 24 CoAP Payload
- 25 Active Operational Dataset TLVs
- 26 Active Operational Dataset TLVs
- 27 Encodes values for the requested Active Operational Dataset parameters. If the MGMT_ACTIVE_GET.req did not contain a Get TLV, the MGMT_ACTIVE_GET.rsp message MUST include the entire Active Operational Dataset.

30 **8.7.4.5 MGMT_ACTIVE_SET.req – Set Active Operational Dataset Request**

- 32 The Commissioner and Thread Devices may provide a new Active Operational Dataset to the Leader by sending a MGMT_ACTIVE_SET.req.
- 34 CoAP Request URI
- 35 `coap://[<L>]:MM/c/as`
- 36 Transaction Pattern
- 37 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)

- 1 CoAP Payload
- 2 [Commissioner Session ID TLV]
- 3 Active Timestamp TLV
- 4 Active Operational Dataset TLVs
- 5 Commissioner Session ID TLV
 - 6 Includes the Commissioner Session ID provided by the Leader in the COMM_PET.rsp message. When originated by a Commissioner, the MGMT_ACTIVE_SET.req message MUST include the Commissioner Session ID. When originated by a Thread Device, the MGMT_ACTIVE_SET.req message MUST NOT include the Commissioner Session ID.
- 10 Active Timestamp TLV
 - 11 Encodes the Active Timestamp associated with the new Active Operational Dataset.
- 12 Active Operational Dataset TLVs
 - 13 Encodes all other Active Operational Dataset parameters.

8.7.4.6 MGMT_ACTIVE_SET.rsp – Set Active Operational Dataset Response

- 16 The Leader MUST send a MGMT_ACTIVE_SET.rsp in response to receiving a MGMT_ACTIVE_SET.req message.
- 18 CoAP Response Code
 - 19 2.04 Changed
- 20 Transaction Pattern
 - 21 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)
- 22 CoAP Payload
- 23 State TLV
- 24 State TLV
 - 25 Encodes the result of executing the MGMT_ACTIVE_SET.req message. ‘Accept’ indicates that all TLVs in the MGMT_ACTIVE_SET.req message were accepted. ‘Reject’ indicates that none of the TLVs in the MGMT_ACTIVE_SET.req message were accepted.

8.7.4.7 TLV Encoding

- 29 The Active Operational Dataset parameters are encoded using the following MeshCoP TLVs when communicated to other devices (see Table 8-6).

31 **Table 8-6. Active Operational Dataset Parameter TLV Encoding**

Active Operational Dataset Parameter	MeshCoP TLV
Active Timestamp	Active Timestamp TLV
Channel	Channel TLV
Channel Mask	Channel Mask TLV

Active Operational Dataset Parameter	MeshCoP TLV
Extended PAN ID	Extended PAN ID TLV
Mesh-Local Prefix	Mesh-Local Prefix TLV
Network Master Key	Network Master Key TLV
Network Name	Network Name TLV
PAN ID	PAN ID TLV
PSKc	PSKc TLV
Security Policy	Security Policy TLV

8.7.4.8 Leader Behavior

The Leader receives new Active Operational Dataset values via MGMT_ACTIVE_SET.req messages. Presence of a Commissioner Session ID TLV in the MGMT_ACTIVE_SET.req payload indicates that a Commissioner originated the MGMT_ACTIVE_SET.req message. Conversely, absence of a Commissioner Session ID TLV in the MGMT_ACTIVE_SET.req payload indicates that a Thread Device originated the MGMT_ACTIVE_SET.req message. On receiving a MGMT_ACTIVE_SET.req message with Active Operational Dataset values, the Leader updates its Active Operational Dataset with the values included in the message. Any Active Operational Dataset values not included in the MGMT_ACTIVE_SET.req retain their existing values.

A Leader updates the Active Operational Dataset in response to receiving a MGMT_ACTIVE_SET.req as follows:

1. Remove all TLV types included in the MGMT_ACTIVE_SET.req message from the local Active Operational Dataset storage.
2. Copy all TLVs from the MGMT_ACTIVE_SET.req message to the local Active Operational Dataset storage.

A Leader MUST store all TLV types, even those types that it does not understand.

If the MGMT_ACTIVE_SET.req is originated by a Thread Device and includes any parameters that affect connectivity, the Leader MUST delay the update using the Pending Operational Dataset. This handles the case where a newly-attached device has a newer Active Operational Dataset. After receiving the newer Active Operational Dataset, the Leader has to delay the change and update the Pending Operational Dataset to distribute the new parameter values. Rather than updating the Active Operational Dataset directly, the Leader MUST generate a Pending Operational Dataset as follows:

1. Apply the received parameter values to the Pending Operational Dataset.
2. Set the Pending Timestamp to the same value as the Active Timestamp.
3. Set the Delay Timer to DELAY_TIMER_DEFAULT.

- 1 A Leader MUST reject any MGMT_ACTIVE_SET.req message containing Active Operational
2 Dataset values that meet any of the following conditions:
3 1. The request includes a Commissioner Session ID and the Commissioner Session ID does
4 not match the Leader's Commissioner Session ID.
5 2. The request includes a Commissioner Session ID and also includes any Active
6 Operational Dataset parameters that affect connectivity.
7 3. The request does not include an Active Timestamp that is ahead of the locally stored
8 value.
9 4. The request contains any Active Operational Dataset values that are out-of-range (e.g. a
10 Channel value not supported by the PHY).
11 5. The CoAP Payload length exceeds MAX_ACTIVE_OPERATIONAL_DATASET_SIZE.
12 If the Leader rejects the MGMT_ACTIVE_SET.req, the Leader MUST send a
13 MGMT_ACTIVE_SET.rsp message with a State TLV that indicates Reject.
14 If the Leader accepts the MGMT_ACTIVE_SET.req, the Leader MUST send a
15 MGMT_ACTIVE_SET.rsp message with a State TLV that indicates Accept.

16 **8.7.5 Updating the Pending Operational Dataset**

- 17 Any changes to the Pending Operational Datasets MUST be associated with a new Pending
18 Timestamp. The Pending Timestamp is used to establish priority. When two or more Pending
19 Operational Datasets are present the one with the later Pending Timestamp is used.
20 If a Thread Device receives a newer Pending Operational Dataset before an existing running
21 Delay Timer expires, the existing Delay Timer is cancelled and the newly received Pending
22 Operational Dataset is installed.
23 The Leader is responsible for disseminating the Pending Operational Dataset within its
24 Thread Network Partition. Disseminating a new Pending Operational Dataset through a
25 Thread Network Partition requires first providing the Leader with the new Pending
26 Operational Dataset.

27 **8.7.5.1 Updates from a Commissioner**

- 28 An active Commissioner sends a MGMT_PENDING_SET.req message to change one or more
29 Pending Operational Dataset values in the Leader. The payload MUST include the
30 Commissioner Session ID and the Pending Operational Dataset values encoded in
31 appropriate TLVs.
32 Commissioners SHOULD avoid updating Pending Operational Dataset values when the
33 Leader has a running Delay Timer that is less than DELAY_TIMER_DEFAULT. This avoids
34 creating a situation where some devices act on the original Pending Operational Dataset
35 while others discard it in favor of the second one.
36 Commissioners updating the Pending Operational Dataset are subject to the following
37 constraints:
38 1. The MGMT_PENDING_SET.req message MUST include a Pending Timestamp value ahead
39 of the value stored on the Leader.
40 2. The MGMT_PENDING_SET.req message MUST include at most one TLV for each value
41 being set.

- 1 The Pending Timestamp SHOULD be set to the current Coordinated Universal Time (UTC) value on the Commissioner when the Commissioner generates the Pending Operational Dataset.
- 2 The Commissioner MAY set a Pending Timestamp value that is ahead of UTC time if:
 - 3 1. The Commissioner does not know the current UTC time.
 - 4 2. The Leader has a Pending Timestamp value that is ahead of the current UTC time when the Commissioner generates the new Active Operational Dataset.
- 5 In either case above, the Commissioner generates the Pending Timestamp value by taking the current Pending Timestamp value from the Leader and adding a random 15-bit value to the Ticks field, carrying over to the Seconds field, if necessary.
- 6 If a Leader has an existing Pending Operational Dataset, the Leader MUST only accept new Pending Operational Datasets that have a newer Pending Timestamp. As a result, if the Leader has a Pending Timestamp that is set to the maximum value, the Commissioner MUST wait for the existing Delay Timer to expire before issuing a new Pending Operational Dataset.

16 **8.7.5.2 Updates from a Thread Device**

- 17 A Thread Device that has a newer Pending Operational Dataset, as determined by the Pending Timestamp included in Thread Network Data, MUST provide its newer Pending Operational Dataset to the Leader by sending a MGMT_PENDING_SET.req to the Leader. The payload MUST include the entire Pending Operational Dataset as values encoded in appropriate TLVs. All Pending Operational Dataset values MUST be unmodified from the values that were received, except the Delay Timer which encodes the current value when the MGMT_PENDING_SET.req message is generated. Unlike the Commissioner, a Thread Device MUST NOT include a Commissioner Session ID.

25 **8.7.5.3 MGMT_PENDING_GET.req – Get Pending Operational Dataset Request**

- 27 The Commissioner and Thread Devices may retrieve Pending Operational Dataset values from the Leader by sending a MGMT_PENDING_GET.req.

29 CoAP Request URI

30 Commissioner to Border Agent: coaps://[<BA>]:MC/c/pg
31 Border Agent or other Device to Device: coap://[<L>]:MM/c/pg

32 Transaction Pattern

33 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)

34 CoAP Payload

35 [Get TLV]

36 Get TLV

37 Included when requesting a subset of the Pending Operational Dataset and indicates one or more TLV types associated with the requested Pending Operational Dataset parameter values. Not included when requesting the entire Pending Operational Dataset.

40 A Thread Device MUST ignore any TLVs other than the Get TLV included in the CoAP payload.

1 A Thread Device MUST ignore any TLV types included in the Get TLV that are not included in
2 the Pending Operational Dataset.

3 **8.7.5.4 MGMT_PENDING_GET.rsp – Get Pending 4 Operational Dataset Response**

5 The Leader and other Thread Devices MUST send a MGMT_PENDING_GET.rsp in response to
6 receiving a MGMT_PENDING_GET.req message.

7 CoAP Response Code

8 2.04 Changed

9 Transaction Pattern

10 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)

11 CoAP Payload

12 Delay Timer TLV

13 Pending Operational Dataset TLVs

14 Delay Timer TLV

15 Encodes the Pending Operational Dataset Delay Timer.

16 Pending Operational Dataset TLVs

17 Encodes values for the requested Pending Operational Dataset parameters. If the
18 MGMT_PENDING_GET.req did not contain a Get TLV, the MGMT_PENDING_GET.rsp
19 message MUST include the entire Pending Operational Dataset.

20 If there is no valid Pending Operational Dataset available, the CoAP Payload of the
21 MGMT_PENDING_GET.rsp MUST be empty.

22 **8.7.5.5 MGMT_PENDING_SET.req – Set Pending 23 Operational Dataset Request**

24 The Commissioner and Thread Devices may provide a new Pending Operational Dataset to
25 the Leader by sending a MGMT_PENDING_SET.req.

26 CoAP Request URI

27 coap://[<L>]:MM/c/ps

28 Transaction Pattern

29 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)

30 CoAP Payload

31 [Commissioner Session ID TLV]

32 Pending Timestamp TLV

33 Active Timestamp TLV

34 Delay Timer TLV

35 Pending Operational Dataset TLVs

36 Commissioner Session ID TLV

37 Includes the Commissioner Session ID provided by the Leader in the COMM_PET.rsp
38 message. Only included if the MGMT_PENDING_SET.req message is originated by a
39 Commissioner.

- 1 Pending Timestamp TLV
- 2 Encodes the Pending Timestamp associated with the new Pending Operational Dataset.
- 3 Active Timestamp TLV
- 4 Encodes the Active Timestamp associated with the new Pending Operational Dataset.
- 5 Delay Timer TLV
- 6 Encodes the Delay Timer associated with the new Pending Operational Dataset.
- 7 Pending Operational Dataset TLVs
- 8 Encodes all other Pending Operational Dataset parameters.

8.7.5.6 MGMT_PENDING_SET.rsp – Set Pending Operational Dataset Response

- 9 The Leader MUST send a MGMT_PENDING_SET.rsp in response to receiving a MGMT_PENDING_SET.req message.
- 10
- 11 CoAP Response Code
- 12 2.04 Changed
- 13 Transaction Pattern
- 14 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)
- 15 CoAP Payload
- 16 State TLV
- 17 State TLV
- 18 Encodes the result of executing the MGMT_PENDING_SET.req message. ‘Accept’ indicates that all TLVs in the MGMT_PENDING_SET.req message were accepted by the Leader. ‘Reject’ indicates that none of the TLVs in the MGMT_PENDING_SET.req message were accepted by the Leader.

8.7.5.7 TLV Encoding

- 25 The Pending Operational Dataset parameters are encoded using the following MeshCoP TLVs when communicated to other devices (see Table 8-7).
- 26

Table 8-7. Pending Operational Dataset Parameter TLV Encoding

Pending Operational Dataset Parameter	MeshCoP TLV
Active Timestamp	Active Timestamp TLV
Channel	Channel TLV
Channel Mask	Channel Mask TLV
Delay Timer	Delay Timer TLV
Extended PAN ID	Extended PAN ID TLV

Pending Operational Dataset Parameter	MeshCoP TLV
Mesh-Local Prefix	Mesh-Local Prefix TLV
Network Master Key	Network Master Key TLV
Network Name	Network Name TLV
PAN ID	PAN ID TLV
Pending Timestamp	Pending Timestamp TLV
PSKc	PSKc TLV
Security Policy	Security Policy TLV

8.7.5.8 Leader Behavior

The Leader receives new Pending Operational Dataset values via MGMT_PENDING_SET.req messages. Presence of a Commissioner Session ID TLV in the MGMT_PENDING_SET.req payload indicates that a Commissioner originated the MGMT_PENDING_SET.req message. Conversely, absence of a Commissioner Session ID TLV in the MGMT_PENDING_SET.req payload indicates that a Thread Device originated the MGMT_PENDING_SET.req message.

On receiving a MGMT_PENDING_SET.req message with Operational Dataset values, the Leader updates its Operational Dataset with the values included in the message. Any Pending Operational Dataset values not included in the MGMT_PENDING_SET.req are copied from the Active Operational Dataset.

A Leader updates the Pending Operational Dataset in response to receiving a MGMT_PENDING_SET.req as follows:

1. Clear the local Pending Operational Dataset storage.
2. Copy all TLVs from the MGMT_PENDING_SET.req message to the local Pending Operational Dataset storage.
3. If the updated Delay Timer carries a value less than DELAY_TIMER_MINIMAL (when the Pending Dataset does not update the Master Key) or DELAY_TIMER_DEFAULT (when the Master Key does change), set the Delay Timer to DELAY_TIMER_MINIMAL or DELAY_TIMER_DEFAULT respectively.
4. Copy any TLVs from the local Active Operational Dataset storage whose types were not included in the MGMT_PENDING_SET.req message to the local Pending Operational Dataset storage.

A Leader MUST store all TLV types, even those types that it does not understand.

A Leader MUST reject any MGMT_PENDING_SET.req message containing Pending Operational Dataset values that meet any of the following conditions:

1. The request includes a Commissioner Session ID and the Commissioner Session ID does not match the Leader's Commissioner Session ID.
2. The request does not include a Pending Timestamp that is ahead of the locally stored value.
3. The request contains any Operational Dataset values that are out-of-range (e.g. a Channel value not supported by the PHY).

- 1 4. The CoAP Payload length exceeds MAX_PENDING_OPERATIONAL_DATASET_SIZE.
- 2 5. If the Active Timestamp is set to a lower value than the current Active Timestamp
3 without changing the Network Master Key.
- 4 If the Leader rejects the MGMT_PENDING_SET.req, the Leader MUST send a
5 MGMT_PENDING_SET.rsp message with a State TLV that indicates Reject.
- 6 If the Leader accepts the new Pending Operational Dataset values, the Leader MUST send a
7 MGMT_PENDING_SET.rsp message.

8.7.6 Concurrent Updates

The Leader may receive Active or Pending Operational Dataset updates from either a Commissioner or Thread Devices within its Thread Network Partition, as specified in previous sections. If a Leader has an active Commissioner and accepts an updated Active or Pending Operational Dataset from a Thread Device, the Leader MUST send a MGMT_DATASET_CHANGED.ntf message to the Commissioner.

8.7.6.1 MGMT_DATASET_CHANGED.ntf –Dataset Changed Notification

The Leader notifies an active Commissioner that an Active or Pending Operational Dataset has changed by sending a MGMT_DATASET_CHANGED.ntf message to the Commissioner via the Border Agent proxy.

CoAP Request URI

coap://[<BA>]:MM/c/dc

Transaction Pattern

Ntf/Qry/Ans_CON

CoAP Payload

None

8.7.7 Orphaned End Devices

An update to the Channel and/or PAN ID may result in orphaned End Devices if they failed to receive the new Pending Operational Dataset. Because End Devices rely on a Parent to provide connectivity, End Devices that cannot find a Parent on its existing Channel and PAN ID SHOULD search for Parents on other Channels or PAN IDs. While a Network Master Key change may also result in orphaned End Devices, an End Device cannot attempt to attach to a network without the necessary Master Key.

An End Device searches for Parents on other Channels or PAN IDs using MLE Announce messages. To search a channel, the End Device sends an MLE Announce message on that channel with an Active Timestamp with the 'U' bit set and a time value of all zeros. The End Device then waits for responses. Any network device that hears the MLE Announce message will have a later Active Timestamp and will respond with an MLE Announce message containing its PAN ID and Active Timestamp.

If the initial MLE Announce search on the channels indicated by the Active Commissioning Data Scan Mask does not succeed, an End Device MAY retry as indicated by the application.

- 1 If the MLE Announce search is ultimately unsuccessful, this may indicate a persistent
2 connectivity issue or a Network Master Key change and the user should be guided to check
3 the network connectivity or initiate a scan based on network Discovery Messages followed
4 by recommissioning.

5 **8.7.8 Merging Channel and PAN ID Partitions**

6 An update to the Channel and/or PAN ID parameters may cause a Thread Network Partition
7 if a subset of Routers or REEDs fail to receive the new Pending Operational Dataset. This
8 section describes mechanisms that may be used to reunite a network that has become
9 partitioned across two or more channels or PAN IDs.

10 Devices operating on different channels and/or PAN IDs necessarily have different Active
11 Operational Datasets. The one with the latest Active Timestamp takes priority; when faced
12 with a choice between two Active Operational Datasets, a device MUST choose the Active
13 Operational Dataset with the later Active Timestamp value.

14 **8.7.8.1 MGMT_ANNOUNCE_BEGIN.ntf –Announce Begin 15 Notification**

16 A Commissioner may send a MGMT_ANNOUNCE_BEGIN.ntf command to initiate Operational
17 Dataset Announcements. A MGMT_ANNOUNCE_BEGIN.ntf message contains parameters
18 used for driving the Operational Dataset Announcements. A MGMT_ANNOUNCE_BEGIN.ntf
19 may be unicast or multicast.

20 CoAP Request URI

21 `coap://[<destination>]:MM/c/ab`

22 Transaction Pattern

23 Ntf/Qry/Ans_CON (unicast) or Ntf/Qry/Ans_NON (multicast)

24 CoAP Payload

25 Commissioner Session ID TLV

26 Channel Mask TLV

27 Count TLV

28 Period TLV

29 Commissioner Session ID TLV

30 Includes the Commissioner Session ID provided by the Leader in the COMM_PET.rsp
31 message.

32 Channel Mask TLV

33 Identifies the set of channels used to transmit MLE Announce messages.

34 Count TLV

35 Identifies the number of MLE Announce transmissions per channel.

36 Period TLV

37 Identifies the period in milliseconds between successive MLE Announce transmissions.

1 **8.7.9 Avoiding PAN ID Conflicts**

2 Any change to the Channel or PAN ID parameters may result in a PAN ID conflict with
3 neighboring IEEE 802.15.4 networks. This section describes mechanisms that allow a
4 Commissioner to query whether a specific PAN ID is in use on one or more channels. Prior
5 to changing the Channel or PAN ID, the Commissioner SHOULD use these mechanisms to
6 detect if a PAN ID conflict may occur.

7 **8.7.9.1 MGMT_PANID_QUERY.qry – PAN ID Query 8 Notification**

9 The Commissioner sends a MGMT_PANID_QUERY.qry message to request one or more
10 devices to perform an IEEE 802.15.4 Active Scan and determine if a specific PAN ID value is
11 in use. A MGMT_PANID_QUERY.qry may be unicast or multicast.

12 CoAP Request URI

13 `coap://[<destination>]:MM/c/pq`

14 Transaction Pattern

15 Ntf/Qry/Ans_CON (unicast) or Ntf/Qry/Ans_NON (multicast)

16 CoAP Payload

17 Commissioner Session ID TLV

18 Channel Mask TLV

19 PAN ID TLV

20 Commissioner Session ID TLV

21 Includes the Commissioner Session ID provided by the Leader in the COMM_PET.rsp
22 message.

23 Channel Mask TLV

24 Specifies the ScanChannels and ChannelPage when performing an IEEE 802.15.4 Active
25 Scan.

26 PAN ID TLV

27 Specifies the PAN ID that is used to check for conflicts.

28 A Thread Device that receives a MGMT_PANID_QUERY.qry message MUST delay its IEEE
29 802.15.4 Active Scan operation by SCAN_DELAY after receiving the
30 MGMT_PANID_QUERY.qry message.

31 **8.7.9.2 MGMT_PANID_CONFLICT.ans – PAN ID Conflict 32 Notification**

33 A Thread Device that performs an IEEE 802.15.4 Active Scan in response to receiving a
34 MGMT_PANID_QUERY.qry checks if the specified PAN ID is in use by another IEEE 802.15.4
35 network. If the Active Scan resulted in the device receiving an IEEE 802.15.4 Beacon from a
36 different IEEE 802.15.4 network and the Source PAN matches the PAN ID, the Thread
37 Device MUST unicast a MGMT_PANID_CONFLICT.ans to the source of the
38 MGMT_PANID_QUERY.qry message.

39 CoAP Request URI

1 coap://[<destination>]:MM/c/pc

2 Transaction Pattern

3 Ntf/Qry/Ans_CON

- 1 CoAP Payload
- 2 Channel Mask TLV
- 3 PAN ID TLV
- 4 Channel Mask TLV
- 5 Indicates the channel(s) where the PAN ID conflict was detected.
- 6 PAN ID TLV
- 7 Specifies the conflicting PAN ID.

8.7.10 Collecting Energy Scan Information

When selecting a new Channel for a Thread Network, it may be desirable to request energy measurement data from Thread Devices. A Commissioner MAY send a MGMT_ED_SCAN.qry message to one or more devices to request an energy measurement on one or more channels. The MGMT_ED_SCAN.qry may be unicast or multicast.

8.7.10.1 MGMT_ED_SCAN.qry – Energy Detect Scan Notification

The MGMT_ED_SCAN.qry message may be used to request a Thread Device to measure and report energy measurements on one or more channels. A Thread Device receiving the MGMT_ED_SCAN.qry performs IEEE 802.15.4 ED Scans on the specified channels.

- CoAP Request URI
coap://[<destination>]:MM/c/es
- Transaction Pattern
Ntf/Qry/Ans_CON (unicast) or Ntf/Qry/Ans_NON (multicast)
- CoAP Payload
 - Commissioner Session ID TLV
 - Channel Mask TLV
 - Count TLV
 - Period TLV
 - Scan Duration TLV
- Commissioner Session ID TLV
Includes the Commissioner Session ID provided by the Leader in the COMM_PET.rsp message.
- Channel Mask TLV
Specifies the ScanChannels and ChannelPage when performing an IEEE 802.15.4 ED Scan.
- Count TLV
Identifies the number of IEEE 802.15.4 ED Scans per channel.
- Period TLV
Identifies the period between successive IEEE 802.15.4 ED Scans.

1 Scan Duration TLV

2 Identifies the IEEE 802.15.4 ScanDuration to use when performing an IEEE 802.15.4 ED
3 Scan.

4 When servicing a MGMT_ED_SCAN.qry message, a Thread Device MUST perform IEEE
5 802.15.4 ED Scans using a single channel at a time – the ScanChannels parameter MUST
6 specify only one channel. Count specifies the number of ED scan rounds. Each ED scan
7 round consists of performing one ED scan for each channel in numerical order, separated by
8 Period milliseconds. If the Channel Mask TLV selects N channels, the Thread Device MUST
9 perform N * Count IEEE 802.15.4 ED Scans. This gives flexibility for how a Commissioner
10 can instruct a device to perform energy scans. Future versions of the specification may
11 elaborate on what can be done with the data.

12 A Thread Device that receives a MGMT_ED_SCAN.qry message MUST delay its IEEE
13 802.15.4 ED Scan operation by SCAN_DELAY after receiving the MGMT_ED_SCAN.qry
14 message.

15 Servicing a MGMT_ED_SCAN.qry message may take a significant amount of time. If a
16 Thread Device notices that a Commissioner is no longer active on the network, as indicated
17 by the presence of the Border Agent Locator TLV in the Commissioner Dataset, the Thread
18 Device MUST stop servicing the MGMT_ED_SCAN.qry message.

19 **8.7.10.2 MGMT_ED_REPORT.ans –Energy Detect Report
20 Notification**

21 A Thread Device reports the results of its IEEE 802.15.4 ED scan in a
22 MGMT_ED_REPORT.ans message. The MGMT_ED_REPORT.ans is unicast to the source of
23 the MGMT_ED_SCAN.qry message that triggered the scan.

24 CoAP Request URI

25 coap://[<destination>]:MM/c/er

26 Transaction Pattern

27 Ntf/Qry/Ans_CON

28 CoAP Payload

29 Channel Mask TLV

30 Energy List TLV

31 Channel Mask TLV

32 Indicates the channels that have energy measurements reported in the Energy List TLV.

33 Energy List TLV

34 A list of energy measurement values in dBm corresponding to the channels selected in
35 the Channel Mask TLV. With N selected channels in the Channel Mask TLV, the first N
36 entries represent the first energy measurement values for the N channels. The next N
37 entries represent the second energy measurement values for the N channels. And so on.

38 A Thread Device MUST delay its MGMT_ED_REPORT.ans message until after the requested
39 scan operation is complete. The delay is computed by:

- 40 • Count * NumChannels * (ScanDuration + Period) + 500ms

8.8 Thread Network Scope Commands

This section covers commands that are only sent over the Thread Network between the Border Agent and Leader or other Thread Device.

8.8.1 Leader Commands

8.8.1.1 LEAD_PET.req – Leader Petition Request

LEAD_PET.req is sent from the Border Agent to the Leader to petition for the Commissioner role. Sent upon completion of a Commissioning Session handshake, triggered by any COMM_PET.req.

CoAP Request URI

coap://[<L>]:MM/c/lp

Transaction Pattern

Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)

CoAP Payload

Commissioner ID TLV

Commissioner ID TLV

A human-readable identifier for the Commissioner Candidate. This string will be advertised to other clients if the device becomes the active Commissioner.

8.8.1.2 LEAD_PET.rsp – Leader Petition Response

LEAD_PET.rsp is sent from the Leader to the Border Agent in response to LEAD_PET.req. Communicates whether or not the Leader will let the petitioning Commissioner Candidate take the role.

CoAP Response Code

2.04 Changed

Transaction Pattern

Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)

CoAP Payload

State TLV

[Commissioner ID TLV]

[Commissioner Session ID TLV]

State TLV

Result of the petitioning request. ‘Accept’ means the device is now the active Commissioner, and a session ID is included. ‘Reject’ means another device is active, or the current device resigned. If another device is active, the Commissioner ID for the active device is included.

Commissioner ID TLV

Human-readable identifier for the active Commissioner.

1 Commissioner Session ID TLV

2 This field is only present when the State value is 'Accept', in which case the device has
3 been elected as Commissioner. All subsequent keep-alives must use this session ID.

4 **8.8.1.3 LEAD_KA.req – Leader Petition Keep Alive 5 Request**

6 LEAD_KA.req is sent from the Border Agent to the Leader to notify the Leader that the
7 active Commissioner is still alive, or that the Commissioner has resigned from being active.
8 The message is triggered when a COMM_KA.req is received. A Commissioning Session
9 expires at the Leader TIMEOUT_LEAD_PET seconds after being established or after the last
10 LEAD_KA.rsp was sent. If the Leader receives a LEAD_KA.req referencing a non-active
11 Commissioning Session, it responds with a LEAD_KA.rsp with State = 'Reject'.

12 On receipt of a LEAD_KA.req, the Leader MUST attempt to find the Commissioning Session
13 based on the Commissioner Session ID TLV value. If the corresponding Commissioning
14 Session does not exist, the Leader MUST send a LEAD_KA.rsp with State TLV value set to
15 'Reject' to the originator of the LEAD_KA.req. If the corresponding Commissioning Session
16 exists, the Leader MUST inspect the State TLV.

17 If the State TLV value is 'Reject', the Leader MUST:

- 18 1. Delete the corresponding Commissioning Session.
- 19 2. Send a LEAD_KA.rsp with State TLV value set to 'Reject' to the originator of the
20 LEAD_KA.req.

21 If the State TLV value is 'Accept', the Leader MUST:

- 22 1. Reset the corresponding Commissioning Session timeout timer.
23 Note: If the Commissioning Session timer on the Leader expires, the Leader MUST
24 delete the corresponding Commissioning Session.
- 25 2. Send a LEAD_KA.rsp with State TLV value set to 'Accept' to the originator of the
26 LEAD_KA.req.

27 CoAP Request URI

28 coap://[<L>]:MM/c/la

29 Transaction Pattern

30 Req+Rsp_Piggybacked or Req+Rsp_Separate (request part)

31 CoAP Payload

32 State TLV

33 Commissioner Session ID TLV

34 State TLV

35 Set to 'Reject' if the Commissioner has resigned. Otherwise, set to 'Accept'.

36 Commissioner Session ID TLV

37 The Leader-allocated session ID for this active Commissioner.

38 **8.8.1.4 LEAD_KA.rsp – Leader Petition Keep Alive 39 Response**

40 LEAD_KA.rsp is sent by the Leader to the Border Agent in response to a LEAD_KA.req.

- 1 CoAP Response Code
- 2 2.04 Changed
- 3 Transaction Pattern
- 4 Req+Rsp_Piggybacked or Req+Rsp_Separate (response part)
- 5 CoAP Payload
- 6 State TLV
- 7 State TLV
- 8 Set to 'Reject' if a different Commissioner has meanwhile become active and thus the
- 9 Keep Alive request cannot be granted. Otherwise, set to 'Accept'

10 **8.8.2 Thread Network Data Update**

11 The Leader propagates its Commissioner and Operational Datasets to all routers and router-
12 eligible devices using the Thread Network Data mechanism the Leader uses to propagate
13 changes to network-wide data (see Section 5.15, Network Data and Propagation, in Chapter
14 5, Network Layer). This update occurs to announce any change to the Commissioner and
15 Operational Datasets, for example, whether or not a Commissioner is up. The Thread
16 Network Data is updated and propagated through the Thread Network at the same time as
17 LEAD_PET.rsp is sent.

18 **8.8.2.1 Commissioning Data TLV**

19 This is the definition for the Commissioning Data TLV per Section 5.18.5, Commissioning
20 Data TLV in Chapter 5, Network Layer that is propagated via the Thread Network Data.

21 Commissioning Data TLV

22 The Thread Network Data carries a Commissioning Data TLV that MUST contain:

23 Commissioner Session ID TLV

24 And these optional TLVs are included when the Commissioner is active:

25 Border Agent Locator TLV

26 [Steering Data TLV]

27 If a Thread Device receives a newer Operational Dataset Timestamp than it possesses, it
28 will query the neighbor with the newer timestamp via MGMT_GET.req for the following
29 user-settable fields:

30 PSKc TLV

31 Network Name TLV

32 Security Policy TLV

1 Commissioner Session ID TLV

2 The Leader-allocated session ID for this active Commissioner. This number increments
3 anytime there is a change of active Commissioner state at the Leader: either an elected
4 transition from a 'no Commissioner' state to an 'active Commissioner' state, or a
5 transition due to a resign action from an 'active Commissioner' state to a 'no
6 Commissioner' state, or when a new active Commissioner replaces the current active
7 Commissioner.

8 Operational Dataset Timestamp TLV

9 The UTC (Coordinated Universal Time) time the Operational Dataset was last changed by
10 the user. This is used to ensure that the user's most recent changes are retained and
11 propagated during merging of Thread Network Partitions.

12 PSKc TLV

13 The PSKc is derived from the Commissioning Credential passphrase that a Commissioner
14 needs to connect to and authenticate with the Thread Network. All devices in the
15 network learn the PSKc through this Thread Network Data.

16 Security Policy TLV

17 This bit mask specifies network administrator preferences for which security-related
18 operations are allowed or disallowed.

19 Network Name TLV

20 This human-settable network name is included because it may be changed, and needs to
21 be consistently propagated across the network.

22 Border Agent Locator TLV

23 The RLOC16 of the Border Agent for this Commissioner. Presence of this field indicates
24 that a Commissioner is active on the network; otherwise, joining is disabled.

25 Steering Data TLV

26 Indicates which Joiners should join this network. A zero value indicates that joining is
27 disabled, which is the default for a new Commissioner. Only present when Joining is
28 enabled.

29 **8.8.2.2 Thread Network Data Change Callback**

30 Thread Network Data changes at the lower layer cause events which need to be handled. If
31 for example, the Thread Device changes Thread Network Partitions, the new Thread
32 Network Partition may have an older PSKc in the Thread Network Data.

33 Upon receiving such a Thread Network Data change event, the device compares the
34 Operational Dataset Timestamp field from the lower layer with the one at the higher layer.
35 The Thread Network Data at the lower layer is then set to be the most recent Operational
36 Dataset. This provides Thread Network Partition tolerance for the network parameters that
37 the user can administer.

38 **8.8.2.3 Thread Network Data Change Command**

39 Any updates to the Operational Dataset on the Leader via MGMT_SET commands MUST also
40 trigger writing that updated Operational Dataset to the Thread Network Data. The
41 Operational Dataset Timestamp MUST always increment forward and be more recent than
42 the last.

8.8.2.4 Thread Network Data Update after Attach

When a device attaches and receives new Thread Network Data from the Thread Network Partition it is attaching to, it must compare the timestamps of the Operational Dataset with what it has stored in non-volatile memory. If the device has a newer Operational Dataset than the Leader's, it registers the new Operational Dataset with a MGMT_SET.req command.

8.9 Joiner Scope Commands

8.9.1 Joiner Entrust Commands

These commands provide the means for the Joiner to acquire the final master key material from the Joiner Router.

The entrust commands have special security handling: they are sent using MAC security with the KEK as the AES-128 encryption key in pairwise key mode (key id mode = 0).

8.9.1.1 JOIN_ENT.ntf – Joiner Entrust Notification

This message performs the final, secure transfer of Network Credentials to the Joiner. It is sent by the Joiner Router after it has received a KEK TLV within a RLY_TX message destined for the requesting Joiner. The Joiner Router first relays DTLS record from the RLY_TX, and then sends this message using the KEK. A Joiner MAY calculate the KEK after it has generated the DTLS KeyBlock, but it cannot authenticate the KEK until the DTLS handshake is complete. As such, the Joiner inevitably requires some processing time between the final message of the DTLS handshake to validate the KEK and insert it into the key table within the MAC. To provide time for the Joiner to prepare for receiving this message with the KEK key, the Joiner Router MUST impose a delay of DELAY_JOIN_ENT between forwarding the DTLS record contained in a RLY_TX that includes the KEK TLV, and sending this message.

Note: This message and the dummy response returned use KEK-encrypted MAC security described in Section 8.9.1, Joiner Entrust Commands.

Upon receiving the Network Credentials, the Joiner delays attaching with those credentials until after the Joiner Finalization and Joiner Provisioning phases are complete.

CoAP Request

coap://[<J>]:MM/c/je

CoAP Payload

Network Master Key TLV

Network Mesh Local Prefix TLV

Network Extended PAN ID TLV

Network Name TLV

Active Timestamp TLV

Channel Mask TLV

PSKc TLV

Security Policy TLV

Network Key Sequence Counter TLV

1 The payload includes Network Management TLVs containing the Network Credentials for the
2 network. All TLVs are required. The Commissioner Dataset fields are included in case the
3 node is immediately orphaned. The key sequence will be sent again during the attach.

4 **8.9.2 Joiner Session Commands**

5 These commands are sent using the secured Joiner Session after the Joiner has been
6 authenticated to perform any subsequent Joiner Provisioning if required.

7 **8.9.2.1 JOIN_FIN.req – Joiner Finalization Request**

8 JOIN_FIN.req is sent by the Joiner to the Commissioner using the Joiner Session to
9 negotiate whether subsequent provisioning is required and to determine whether a
10 provisioning-capable Commissioner is running.

11 CoAP Request URI

12 `coap://[<JR>]:MJ/c/jf`

13 Transaction Pattern

14 Req+Rsp_Piggybacked (request part)

15 CoAP Payload

16 State TLV

17 Vendor Name TLV

18 Vendor Model TLV

19 Vendor SW Version TLV

20 [Vendor Data TLV]

21 Vendor Stack Version TLV

22 [Provisioning URL TLV]

23 State TLV

24 Set to ‘Accept’ or ‘Reject’ based on whether the Joiner has completed authentication
25 successfully. If set to ‘Reject’, the Commissioner must append the KEK TLV to the
26 response.

27 Vendor Name TLV

28 Name of the product vendor of the Joiner device.

29 Vendor Model TLV

30 Name of the product model of the Joiner device.

31 Vendor SW Version TLV

32 Software version string of the Joiner device.

33 Vendor Data TLV

34 Vendor-specific provisioning data for the Joiner device.

35 Vendor Stack Version TLV

36 Software version string of the Thread stack software.

1 Provisioning URL TLV

2 This string provides a means for the Joiner to have a generic Commissioner application
3 steer the user to the correct application so this device may be fully provisioned to the
4 correct vendor service. If this field is not present, the Joiner is requesting connection to
5 the Thread network by any Commissioner that supports standard Thread commissioning
6 and requires no subsequent vendor-specific provisioning.

7 **8.9.2.2 JOIN_FIN.rsp – Joiner Finalization Response**

8 JOIN_FIN.rsp is sent from the Commissioner to the Joiner using the Joiner Session to inform
9 the Joiner whether the correct Commissioning application is running that can support proper
10 provisioning and to prepare the Joiner Router to receive final Network Credentials.

11 Upon receipt of this message, the Joiner determines the next state for the Joiner Session:

- 12 1. Any Commissioner ('Accept' received in State field and Joiner did not include
13 Provisioning URL TLV in JOIN_FIN.req): Joiner Session closes.
- 14 2. Wrong Commissioner ('Reject' received in State field): Joiner Session closes.
- 15 3. Correct Commissioner ('Accept' received in State field and Joiner included Provisioning
16 URL TLV in JOIN_FIN.req): Switch to vendor provisioning mode. Joiner Session is closed
17 by the vendor specific provisioning protocol outside the scope of this specification.

18 CoAP Response Code

19 2.04 Changed

20 Transaction Pattern

21 Req+Rsp_Piggybacked (response part)

22 CoAP Payload

23 State TLV

24 State

25 Set to 'Reject' if the Provisioning URL in the JOIN_FIN.req does not match a URL the
26 Commissioner application can handle. Set to 'Accept' if there is a match or in the
27 absence of a Provisioning URL in the JOIN_FIN.req. When encapsulated as DTLS
28 Application Data in a RLY_TX.ntf message, the RLY_TX.ntf message will include the KEK
29 TLV.

30 **8.9.3 Joiner Provisioning Commands**

31 These commands are optionally sent on-mesh by an un-provisioned Joiner device that has
32 attached but still needs to steer the user to the right Commissioner application for
33 provisioning.

34 **8.9.3.1 JOIN_APP.req – Joiner Application Request**

35 JOIN_APP.req is sent on-mesh by the Joiner to the Commissioner to indicate subsequent
36 provisioning is required and steer the user to a provisioning-capable Commissioner
37 application.

38 A Joiner that requires provisioning MAY send this message after it initially attaches to the
39 network and whenever there is a change in the active Commissioner until it is properly
40 provisioned. Before sending the message, the Joiner MUST acquire the full Network Data in

1 order to determine whether there is an active Commissioner and the correct Anycast
2 Address to use for that Commissioner.

3 CoAP Request URI
4 `coap://[<Commissioner Anycast>]:MM/c/ja`

5 Transaction Pattern
6 Req+Rsp_Piggybacked (request part)

7 CoAP Payload
8 Provisioning URL TLV
9 Vendor Name TLV
10 Vendor Model TLV
11 Vendor SW Version TLV
12 [Vendor Data TLV]
13 Vendor Stack Version TLV

14 Provisioning URL TLV
15 This string provides a means for the Joiner to have a generic Commissioner application
16 steer the user to the correct application so this device may be fully provisioned to the
17 correct vendor service.

18 Vendor Name TLV
19 Name of the product vendor of the Joiner device.

20 Vendor Model TLV
21 Name of the product model of the Joiner device.

22 Vendor SW Version TLV
23 Software version string of the Joiner device.

24 Vendor Data TLV
25 Vendor-specific provisioning data for the Joiner device.

26 Vendor Stack Version TLV
27 Software version string of the Thread stack software.

28 **8.9.3.2 JOIN_APP.rsp – Joiner Application Response**

29 JOIN_APP.rsp is sent from the Commissioner to the Joiner to inform the Joiner whether the
30 Commissioning is able to support the requested provisioning application.

31 Upon receipt of this message, the Joiner determines the next state for the Joiner Session:

- 32 1. Wrong Commissioner ('Reject' received): Joiner waits for change in Commissioner
33 Session ID before sending JOIN_APP.req again.
- 34 2. Correct Commissioner ('Accept' received): The active Commissioner is running the
35 correct application to provision the Joiner. Subsequent details on the provisioning
36 protocol are out-of-scope of Thread and left up to the application.

37 CoAP Response Code
38 2.04 Changed

- 1 Transaction Pattern
- 2 Req+Rsp_Piggybacked (response part)
- 3 CoAP Payload
- 4 State TLV
- 5 State
 - 6 Set to 'Reject' if the Provisioning URL in the JOIN_APP.req does not match a URL the Commissioner application can handle. Set to 'Accept' if there is a match of the Provisioning URL provided in the JOIN_APP.req.

9 **8.10 MeshCoP TLV Formats**

- 10 All command payloads within the MeshCoP are encoded using the TLV format defined by the TMF where the type and length are one byte each and the length field contains the length of the value in bytes. There are no alignment requirements and no padding.
- 13 The extended length TLV format defined by TMF is used if the length is greater than 254 bytes.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type										Length										Value														

15 **Figure 8-20. General Thread Management TLV Format**

- 16 To allow for future extensions, when processing incoming TLVs defined as having a fixed Length in this version of the specification, then the TLV MUST NOT be discarded and the Value field(s) spanning the specified fixed Length MUST still be processed as valid even when the actual incoming TLV Length is larger than the specified fixed Length.

20 **8.10.1 Network Management TLVs**

- 21 Network management TLVs are used by MGMT_SET and MGMT_GET commands. A MGMT_SET.req command using Network management TLVs SHALL only be accepted by the Leader and the Leader SHALL act only on Network management TLVs marked with write permission (Write-only or Read/Write). A MGMT_GET.req requesting Network management TLVs MAY be accepted by any Thread Device and the Thread Device SHALL act only responds with Network management TLVs marked with read permission (Read-only or Read/Write).

8.10.1.1 Channel TLV (0)

Permissions: Read-only

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x00								Length = 3								ChannelPage								Channel ...										

Figure 8-21. Channel TLV (0)

ChannelPage

An unsigned 8-bit value that specifies the channel page.

Channel

An unsigned 16-bit value that identifies the channel within the channel page.

8.10.1.1.1 Supported Channel Pages

Table 8-8 lists the supported channel pages.

Table 8-8. Supported Channel Pages

Channel Page Value	Description
0	2.4 GHz O-QPSK PHY as defined in Section 6.5 of [IEEE802154]
1 – 255	Reserved

8.10.1.1.2 Supported Channels

The following channels are supported.

Channel Value	Description
0-10	Reserved
11-26	2.4 GHz channels as defined in Section 6.1.2.1 of [IEEE802154]
27 – 65534	Reserved

8.10.1.2 PAN ID TLV (1)

Permissions: Read-only

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x01								Length = 2								PAN ID																		

Figure 8-22. PAN ID TLV (1)

PAN ID

An unsigned 16-bit value that specifies the PAN ID.

8.10.1.3 Extended PAN ID TLV (2)

Permissions: Read-only

Note: If this field is changed (such as by a Commissioner), the PSKc TLV MUST also be updated, because the PSKc is derived from the Extended PAN ID.

When the Extended PAN ID TLV is used in Discovery Response messages, it indicates current network Extended PAN ID. When the Extended PAN ID is used in Discovery Request messages, it indicates Extended PAN ID values that should not answer to Discovery Responses.

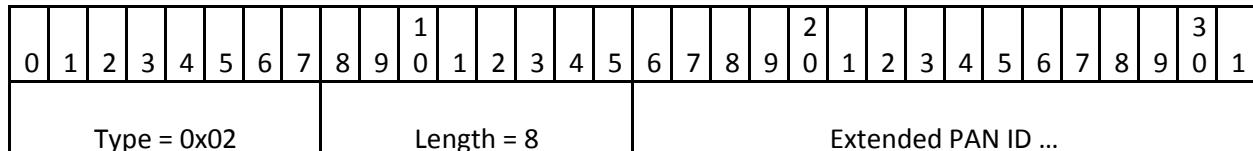


Figure 8-23. Extended PAN ID TLV (2)

Extended PAN ID

The extended PAN identifier for the network.

8.10.1.4 Network Name TLV (3)

Permissions: Read/Write

Max length: 16 bytes

Note: If this field is changed (such as by a Commissioner), the PSKc TLV MUST also be updated, because the PSKc is derived from the Network Name.

When the Network Name TLV is used in Discovery Response messages, it indicates the current network name.

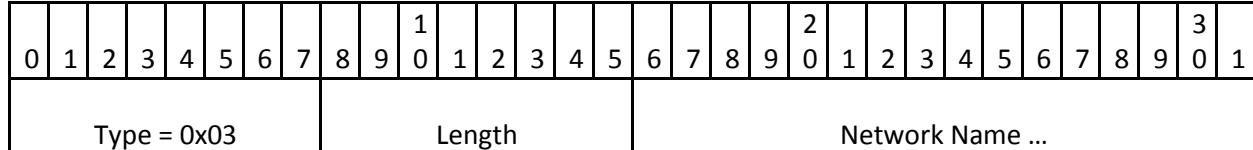


Figure 8-24. Network Name TLV (3)

Network Name

A human-readable utf-8 string to identify the network over-the-air, similar to the Service Set Identifier (SSID) in Wi-Fi. The maximum length of this field is 16 bytes.

8.10.1.5 PSKc TLV (4)

Permissions: Write-only (such as a password entry field)

Max length: 16 bytes

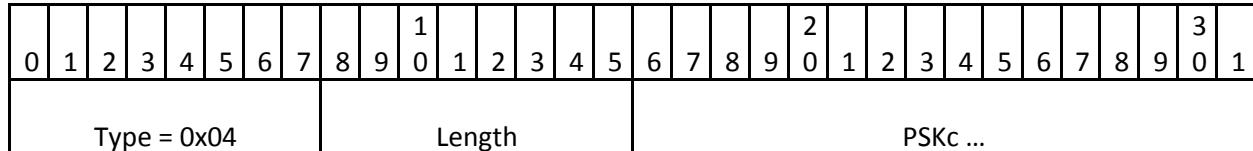


Figure 8-25. PSKc TLV (4)

1 PSKc

2 The key stretched version of the Commissioning Credential for the network. This datum
3 is write-only, and is only changeable by an active Commissioner.

4 **8.10.1.6 Network Master Key TLV (5)**

5 Permissions: Read-only when Security Policy O-bit is set (1), none otherwise

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x05										Length = 16										Network Key ...														

6 **Figure 8-26. Network Master Key TLV (5)**

7 Network Key

8 An 128-bit key

9 **8.10.1.7 Network Key Sequence Counter TLV (6)**

10 Permissions: Read-only when Security Policy O-bit is set (1), none otherwise

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x06										Length = 4										Network Key Sequence Counter ...														

11 **Figure 8-27. Network Key Sequence Counter TLV (6)**

12 Network Key Sequence Counter

13 Key sequence counter value to force a key rotation using the same master key.

14 **8.10.1.8 Network Mesh-Local Prefix TLV (7)**

15 Permissions: Read-only

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x07										Length = 8										Mesh Local Prefix ...														

16 **Figure 8-28. Network Mesh-Local Prefix TLV (7)**

17 Mesh-Local Prefix

18 The prefix used for realm-local traffic within the mesh.

19 **8.10.1.9 Steering Data TLV (8)**

20 Permissions: Read/Write

21 Max length: 16 bytes

22 Steering Data indicates which Joiners are eligible to join the Thread Network.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type = 0x08										Length										Bloom Filter ...												

Figure 8-29. Steering Data TLV (8)

Bloom Filter

This field contains the Bloom Filter as provided by the Commissioner, and specified in Section 8.4.4.3, **Bloom Filters**, to signal which set of Joiner Identifiers (Joiner ID) are permitted to join.

A Bloom Filter of all zeros signals no Joiners are permitted to join.

A Bloom Filter of all ones signals all Joiners are permitted to join.

A Bloom Filter of length = 1 is sufficient for the all or none cases.

A Joiner ID is based on the factory-assigned global IEEE EUI-64 identifier and the MAC Extended Address (link-layer address) assigned to the Thread interface of an originating Joiner.

The Joiner ID consists is the MAC Extended Address of the Joiner Thread interface prior to obtaining the network security credentials. The value of the MAC Extended Address is based on computing SHA-256 on the factory-assigned global IEEE EUI-64 identifier as specified in Table 3-2, Chapter 3, PHY/MAC/6LoWPAN.

To indicate that only a set of specific Joiners are allowed to join, a Commissioner derives the eligible Joiner IDs based on user input which provides the values of the factory assigned IEEE EUI-64 identifiers. The Commissioner then computes the Bloom Filter on the Joiner ID set in order to set the Steering Data value.

8.10.1.10 Border Agent Locator TLV (9)

Permissions: Read-only

Max length: 2 bytes

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Type = 0x09										Length = 2										Border Agent Locator												

Figure 8-30. Border Agent Locator TLV (9)

Border Agent Locator

The RLOC16 of the Border Agent currently acting as the agent for the Commissioner.

8.10.1.11 Commissioner ID TLV (10)

Permissions: Read-only (settable in LEAD_PET.req)

Max length: 64 bytes

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x0A										Length										Commissioner ID ...														

Figure 8-31. Commissioner ID TLV (10)

Commissioner ID

A human-readable utf-8 string provided as identification by the Commissioner. Useful for display on a User Interface (UI) in the case a device has been rejected as the Commissioner to allow the user to know what device currently has the lock on the Commissioner role.

8.10.1.12 Commissioner Session ID TLV (11)

Permissions: Read-only

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x0B										Length = 2										Commissioner Session ID														

Figure 8-32. Commissioner Session ID TLV (11)

Commissioner Session ID

A sequential session ID generated by the Leader to uniquely identify messages regarding a designated Commissioner. This number is incremented anytime there is a change of active Commissioner state at the Leader: either an elected transition from a 'no Commissioner' state to an 'active Commissioner' state, or a transition due to a resign action from an 'active Commissioner' state to a 'no Commissioner' state, or when a new active Commissioner replaces the current active Commissioner.

8.10.1.13 Active Timestamp TLV (14)

Permissions: Read/Write

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x0E										Length = 8										Timestamp Seconds														
Timestamp Ticks																U																		

Figure 8-33. Active Timestamp TLV (14)

- 1 Timestamp Seconds
 - 2 A 48-bit unsigned integer that encodes a Unix Time value.
 - 3 Timestamp Ticks
 - 4 A 15-bit unsigned integer that encodes the fractional Unix Time value in 32.768 kHz
 - 5 resolution.
 - 6 U bit
 - 7 A 1-bit flag that indicates the time was obtained from an authoritative source: either
 - 8 NTP (Network Time Protocol), GPS (Global Positioning System), cell network, or other
 - 9 method. Note that the U bit is the least significant bit when the entire timestamp is
 - 10 represented as a single 64-bit number.

8.10.1.14 Commissioner UDP Port TLV (15)

- ## 12 Permissions: Read/Write

Figure 8-34. Commissioner UDP Port TLV (15)

- 14 UDP Port
15 UDP Port for communicating with the Commissioner

8.10.1.15 Security Policy TLV (12)

- 17 The Security Policy bits provide an administrator a way to enable or disable certain security
18 related behaviors.

- ## 19 Permissions: Read/Write

Figure 8-35. Security Policy TLV (12)

- 21 Rotation Time
22 The value for *thrKeyRotation* in units of hours.
23 O bit
24 Obtaining the Master Key for out-of-band commissioning is enabled when this is set.
25 Defaults to 1. When disabled, all nodes on the Thread Network Partition MUST NOT allow
26 extraction of the Master Network Key from the network layer, thus disallowing
27 depending behaviors such as transfer of the credential outside of the network via
28 MeshCoP MGMT_GET.rsp or any other out-of-band means.

1 N bit

2 Native Commissioning using PSKc is allowed when this is set. Defaults to 1. This enables
3 an unsecured IEEE 802.15.4 interface on any nodes capable of acting as a Border Agent
4 and that have also implemented the PSKc authentication mechanism. This interface can
5 be used by an external, off-mesh Native Commissioner that only knows the PSKc but
6 does not yet have the network credentials.

7 R bit

8 Thread 1.x Routers are enabled when this is set. Defaults to 1. This bit, when cleared,
9 indicates that only devices running a future version of the specification may act as a
10 routing device, and router-capable Thread 1.x devices SHALL NOT attempt to register as
11 an Active Router or Leader.

12 C bit

13 This indicates that external Commissioner authentication is allowed using PSKc. Defaults
14 to 1. When cleared, it indicates that Commissioning Sessions based on the raw PSKc are
15 not allowed to be established and that changes to the Commissioner Dataset by
16 on-mesh nodes are not allowed.

17 B bit

18 Thread 1.x Beacons are enabled when this is set. Defaults to 1. This bit, when cleared,
19 indicates that the beacon payload is either empty or set by the application, and that the
20 only way to discover the network is via Discovery messages.

21 Reserved

22 Reserved bits; must be set to 1.

8.10.1.16 Pending Timestamp TLV (51)

24 Permissions: Read/Write

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1							
Type = 0x33										Length = 8										Timestamp Seconds																			
Timestamp Ticks															U																								

25 **Figure 8-36. Pending Timestamp TLV (51)**

26 Timestamp Seconds

27 A 48-bit unsigned integer that encodes a Unix Time value.

28 Timestamp Ticks

29 A 15-bit unsigned integer that encodes the fractional Unix Time value in 32.768 kHz
30 resolution.

31 U bit

32 A 1-bit flag that indicates the time was obtained from an authoritative source: either
33 NTP (Network Time Protocol), GPS (Global Positioning System), cell network, or other

method. Note that the U bit is the least significant bit when the entire timestamp is represented as a single 64-bit number.

8.10.1.17 Delay Timer TLV (52)

Permissions: Read/Write

0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1		
Type = 0x34	Length = 4	Time Remaining ...

Figure 8-37. Delay Timer TLV (52)

Time Remaining

An unsigned 32-bit number in milliseconds.

8.10.1.18 Channel Mask TLV (53)

Permissions: Read/Write

0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1		
Type = 0x35	Length	Channel Mask Entries [...]

Figure 8-38. Channel Mask TLV (53)

Channel Mask Entries [...]

A list of Channel Mask Entries.

8.10.1.18.1 Channel Mask Entry

0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1		
ChannelPage	MaskLength	ChannelMask ...

Figure 8-39. Channel Mask Entry

ChannelPage

An unsigned 8-bit integer that specifies the channel page, as defined in IEEE 802.15.4-2006. For IEEE 802.15.4-2006 2.4 GHz PHY, the ChannelPage is 0

MaskLength

An unsigned 8-bit integer that indicates the ChannelMask length in octets.

ChannelMask

A variable-length bit mask that identifies the channels within the channel page (1 = selected, 0 = unselected). The channels are represented in most significant bit order. For example, the most significant bit of the left-most byte indicates channel 0. If channel 0 and channel 10 are selected, the mask would be: 80 20 00 00. For IEEE 802.15.4-2006 2.4 GHz PHY, the ChannelMask is 27 bits and Mask Length is 4.

8.10.2 MeshCoP Protocol Command TLVs

These fields are used across many commands across the MeshCoP, or specifically by the relay mechanism. These fields are not readable or writeable via MGMT_GET or MGMT_SET.

8.10.2.1 Get TLV (13)

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x0D	Length										TLV Type Array ...																							

Figure 8-40. Get TLV (13)

TLV Type Array

An array of 8-bit TLV Type codes to get values for.

8.10.2.2 State TLV (16)

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3
Type = 0x10	Length = 1										State														

Figure 8-41. State TLV (16)

State

Generic value encoded as a two's complement 8-bit signed integer used as a status code for response messages or to signal a state change in request messages:

- 1 = Accept
- 0 = Pending
- -1 = Reject

8.10.2.3 Joiner DTLS Encapsulation TLV (17)

This TLV can use either the regular TLV format as shown in Figure 8-42 or the extended TLV format defined within TMF if a longer value size is needed.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x11	Length										Encapsulated DTLS Record ...																							

Figure 8-42. Joiner DTLS Encapsulation TLV (17)

Len

Length of encapsulated DTLS record

Note: Longer length MAY be achieved with a three byte Len field, where the first byte is an escape (0xFF) and the following two bytes are the 16-bit length.

- 1 Encapsulated DTLS Record
- 2 The complete DTLS record to be sent via the relay node.

8.10.2.4 Joiner UDP Port TLV (18)

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x12										Length = 2										UDP Port														

4 **Figure 8-43. Joiner UDP Port TLV (18)**

- 5 UDP Port
- 6 The UDP Port for the Joiner

8.10.2.5 Joiner IID TLV (19)

- 8 This TLV contains the IID corresponding to an accompanying DTLS Encapsulation TLV. When present in a RLY_TX message, the IID represents the final destination of the DTLS record.
- 9 When present in a RLY_RX message, the IID represents the original source of the DTLS record. In both cases, the corresponding link local address can be formed to determine the destination or source respectively.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x13										Length = 8										Joiner IID ...														

13 **Figure 8-44. Joiner IID TLV (19)**

- 14 Joiner IID
- 15 The Joiner IID is based on the MAC Extended Address (link-layer address) of the originating Joiner prior to obtaining the network security credentials.
- 16 The value of the MAC Extended Address is based on computing SHA-256 on the factory-assigned global IEEE EUI-64 identifier as specified in Table 3-2, Chapter 3, PHY/MAC/6LoWPAN.
- 20 The Joiner IID value is obtained by inverting the "u" bit of the MAC Extended Address as defined in Section 2.5.1 and Appendix A of [\[RFC 4291\]](#).

22 8.10.2.6 Joiner Router Locator TLV (20)

- 23 This TLV is used to specify which Joiner Router to use for a relay. A Joiner Router will store its own RLOC16 within this TLV when generating RLY_RX messages. A Border Agent will forward to the RLOC16 stored here when forwarding RLY_TX messages. A Joiner Router must confirm its own address is stored in this TLV before forwarding a RLY_TX message.

27

0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1																												
Type = 0x14	Length = 2	Relay Address																										

Figure 8-45. Joiner Router Locator TLV (20)

3 Relay Address

4 The RLOC16 of the Joiner Router. This field contains the Joiner Router RLOC16 and is
5 used for reconstructing the mesh-local address of the Joiner Router.

6 **8.10.2.7 Joiner Router KEK TLV (21)**

0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3 4 5 6 7 8 9 3 0 1																												
Type = 0x15	Length = 16	KEK ...																										

Figure 8-46. Joiner Router KEK TLV (21)

8 KEK

9 The KEK (Key Encryption Key) is a single use pairwise key that is negotiated between
10 the Joiner and the Commissioner as part of establishing the Joiner Session. This single
11 use pairwise key is then used by the Joiner Router to encrypt the Network Credentials,
12 (including the Network Master Key) sent to the Joiner. The Joiner Router obtains the KEK
13 from a RLY_TX.ntf message containing this TLV.

14 The KEK is derived by performing a SHA256 hash of the 40-byte KeyBlock from the
15 Joiner Session, and taking the bottom 128 bits. The DTLS KeyBlock has this format:

16 client_write_key[16]
17 server_write_key[16]
18 client_write_IV[4] – client write initialization vector
19 server_write_IV[4] – server write initialization vector

20 For more information, see Section 7.4, DTLS, in Chapter 7, Security.

21 **8.10.2.8 Count TLV (54)**

22 Permissions: Read/Write

0 1 2 3 4 5 6 7 8 9 1 0 1 2 3 4 5 6 7 8 9 2 0 1 2 3																												
Type = 0x36	Length = 1	Count																										

Figure 8-47. Count TLV (54)

24 Count

25 An unsigned 8-bit integer that specifies the number of times to perform the requested
26 operation.

8.10.2.9 Period TLV (55)

Figure 8-48. Period TLV (55)

Period

An unsigned 16-bit value that specifies the period in milliseconds between successive operations.

8.10.2.10 Scan Duration TLV (56)

Figure 8-49. Scan Duration TLV (55)

ScanDuration

An unsigned 16-bit value that specifies the duration in milliseconds to use when performing a scan. This value is not the same as the value that is passed directly to the IEEE MLME-SCAN.req. If the MAC has an MLME-SCAN interface that supports using a millisecond time value, then that interface MAY be used with the value of the ScanDuration TLV. If the MAC does not support an MLME-SCAN interface with the ability to use a ScanDuration specified in milliseconds, the ScanDuration TLV shall be translated as follows:

```

if ScanDurationTLV <= 15.36
    n = 0
else
    n = ceiling( log2(ScanDurationTLV / 15.36) )
    // ceiling(x) is the smallest integer value that is greater than or
equal to x.
    // log2(x) is the logarithm base 2 of x

```

```
if (n > 14)
    n = 14
```

8.10.2.11 Energy List TLV (57)

Figure 8-50. Energy List TLV (57)

1 EnergyMeasurementList

2 A variable-length list of unsigned 8-bit values that indicate the measured energy level in
3 dBm.

4 **8.10.3 TMF Provisioning and Discovery TLVs**

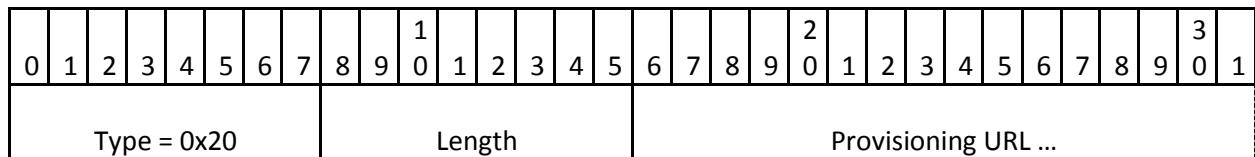
5 This set of fields is primarily used by the Joiner Finalization to negotiate that the correct
6 Commissioner application is running. These fields may also be queried from any Thread
7 Device using MGMT_GET to enable a basic device discovery mechanism.

8 **8.10.3.1 Provisioning URL TLV (32)**

9 Permissions: Read-only

10 Max length: 64 bytes

11



12 **Figure 8-51. Provisioning URL TLV (32)**

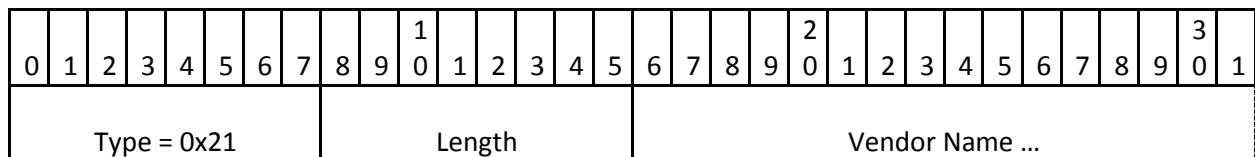
13 Provisioning URL

14 A URL encoded as a utf-8 string provided by the Joiner to communicate to the user
15 which Commissioning application is best suited to properly provision it to the appropriate
16 service.

17 **8.10.3.2 Vendor Name TLV (33)**

18 Permissions: Read-only

19 Max length: 32 bytes



20 **Figure 8-52. Vendor Name TLV (33)**

21 Vendor Name

22 A human-readable product vendor name string in utf-8 format.

8.10.3.3 Vendor Model TLV (34)

Permissions: Read-only

Max length: 32 bytes

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x22										Length										Vendor Model ...														

Figure 8-53. Vendor Model TLV (34)

Vendor Model

A human-readable product model string

8.10.3.4 Vendor SW Version TLV (35)

Permissions: Read-only

Max length: 16 bytes

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x23										Length										Vendor SW Version ...														

Figure 8-54. Vendor SW Version TLV (35)

Vendor SW Version

A utf-8 string that specifies the product software version running on the Joiner device.

8.10.3.5 Vendor Data TLV (36)

Permissions: Vendor determined

Max length: 64 bytes

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x24										Length										Vendor Data ...														

Figure 8-55. Vendor Data TLV (36)

Vendor Data

A product vendor-defined data structure to guide vendor-specific provisioning.

8.10.3.6 Vendor Stack Version TLV (37)

Permissions: Read-only

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x25										Length										Stack Vendor OUI														
										Build										Rev		Minor		Major										

Figure 8-56. Vendor Stack Version TLV (37)

Stack Vendor OUI

OUI (Organizationally Unique Identifier as registered at IEEE) for vendor of the Thread stack software

Major

Major Version number of the Thread stack

Minor

Minor Version number of the Thread stack

Rev

Revision number of the Thread stack

Build

Build number of the Thread stack

8.10.3.7 UDP Encapsulation TLV (48)

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1									
Type = 0x30										1	1	1	1	1	1	1	Length																										
UDP Source Port																UDP Destination Port																											
UDP Payload ...																																											

Figure 8-57. UDP Encapsulation TLV (48)

UDP Source Port

Identifies the sender's port and SHOULD be assumed to be the port to reply to if needed. If not used, then it SHOULD be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.

UDP Destination Port

Identifies the receiver's port and is required. Similar to the UDP Source Port number, if the client is the destination host, then the port number will likely be an ephemeral port

1 number. If the destination host is the server, then the port number will likely be a well-known port number.
2

3 UDP Payload

4 The payload portion of the encapsulated UDP datagram.

5 **8.10.3.8 IPv6 Address TLV (49)**

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x31	Length = 16										IPv6 Address ...																						

6 **Figure 8-58. IPv6 Address TLV (49)**

7 IPv6 Address

8 An IPv6 address

9 **8.10.3.9 Discovery Request TLV (128)**

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x80	Length=2										Ver=2	J	Rsv																					

10 **Figure 8-59. Discovery Request TLV (128)**

11 Ver

12 Protocol Version = 2

13 Originators of Discovery Requests implementing this version of the specification MUST set the Protocol Version to value 2.

14 J

15 Joiner Flag

16 When set to '1' indicates the Discovery Request is generated by a device performing network discovery in order to subsequently join via in-band commissioning.

17 Devices that generate the Discovery Request without the intention to actively join with in-band commissioning SHOULD set the Joiner flag to '0'.

18 Rsv

19 Reserved for future use. MUST be set to '0' by devices implementing this specification.

8.10.3.10 Discovery Response TLV (129)

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x81								Length=2								Ver=2		N	Rsv															

Figure 8-60. Discovery Response TLV (129)

Ver

Protocol Version = 2

Originators of Discovery Requests implementing this version of the specification MUST set the Protocol Version to value 2.

N (Native Commissioner) flag

When the N flag is set, signals that the device sending the Discovery Response can serve as a Border Agent for a Commissioner Candidate with a native IEEE 802.15.4 interface to petition to be Commissioner over the unsecured IEEE 802.15.4 interface.

Rsv

Reserved for future use. MUST be set to '0' by devices implementing this specification.

8.11 Parameters and Constants

Table 8-9 is a glossary of constants used in this chapter, along with a brief description and the default for each constant.

Table 8-9. Glossary of Constants

Constant Name	Description	Default
JOINER_RELAY_RATE_LIMIT	The maximum amount of unsecured messages a Joiner Router will forward per second.	10 per second
COMM_PET_ATTEMPT_DELAY	The minimum delay between failed attempts to petition by a Commissioner Candidate.	5 seconds
COMM_PET_RETRY_COUNT	The number of times a Commissioner Candidate may retry sending the COMM_PET.req message per attempt to petition.	2 retries

Constant Name	Description	Default
COMM_PET_RETRY_DELAY	The minimum delay between COMM_PET.req messages per attempt to petition.	1 second
DELAY_JOIN_ENT	The minimum delay for a Joiner Router between forwarding the DTLS record within a RLY_TX message containing a KEK TLV and sending a subsequent JOIN_ENT.ntf message.	50 ms
TIMEOUT_COMM_PET	The maximum amount of time between consecutive COMM_KA.req keep-alive messages before a Commissioner device is rejected.	50 seconds
TIMEOUT_LEAD_PET	The maximum amount of time between consecutive LEAD_KA.req keep-alive messages before a Commissioner device is rejected.	50 seconds
MAX_DELAY_TIMER	Maximum Delay Timer value for a Pending Operational Dataset.	72 hours
DELAY_TIMER_MINIMAL	Minimum Delay Timer value for a Pending Operational Dataset (as set by the Commissioner and enforced by the Leader).	30 seconds
DELAY_TIMER_DEFAULT	Default Delay Timer value for a Pending Operational Dataset generated by the Leader. Minimum value enforced by the Leader when updating the Master Key.	5 minutes

Constant Name	Description	Default
DELAY_TIMER_EXTENDED	Recommended minimum Delay Timer value for a Pending Operational Dataset set by a Commissioner when updating the Network Master Key.	8 hours
MAX_ACTIVE_OPERATIONAL_DATASET_SIZE	Maximum size of the Active Operational Dataset when encoded with TLVs.	[Thread Conformance specification]
MAX_PENDING_OPERATIONAL_DATASET_SIZE	Maximum size of the Pending Operational Dataset when encoded with TLVs.	[Thread Conformance specification]
SCAN_DELAY	Time delay before starting an IEEE 802.15.4 Scan operation.	1 second
DISCOVERY_MAX_JITTER	Maximum jitter time used to delay generation of Discovery Responses.	250 ms
DISCOVERY_TIME	Time an originator of a Discovery Request should wait for incoming Discovery Responses on a channel.	300 ms
DISCOVERY_RATE_LIMIT_INTERVAL	Interval used to rate limit a device when sending multiple consecutive Discovery Requests on the same channel.	100 ms

¹ **8.12 IANA Considerations**

- ² Table 8-10 lists the ports used in MeshCoP and their default values. Because both ports are in the dynamic/private UDP port range, IANA registration for these ports is not needed.

1

Table 8-10. MeshCoP Ports

Service name	Port abbreviation	Default port number	Transport	Description
meshcop	:MC	49191	udp, tcp	Thread Commissioner Port
threadjoin	:MJ	(none)	udp	Thread Joiner Port

2

CHAPTER 9 BORDER ROUTER**Contents**

3	9.1 Introduction.....	9-2
4	9.2 Overview.....	9-2
5	9.2.1 Common Characteristics.....	9-2
6	9.2.2 Border Router Availability.....	9-3
7	9.2.3 Global Addresses.....	9-3
8	9.2.4 Supplemental Unique Local Addresses	9-3
9	9.2.5 Network Data	9-4
10	9.2.6 Coping with IPv4 Infrastructure	9-4
11	9.2.7 Packet Filtering	9-5
12	9.2.8 Role in Commissioning.....	9-5
13	9.3 Functional Specification	9-5
14	9.3.1 Participation in the Interior Network	9-5
15	9.3.1.1 Configuration of Network Data	9-6
16	9.3.1.2 Propagation of Network Data	9-7
17	9.3.1.3 DHCPv6 Server for Global EID Assignment.....	9-7
18	9.3.2 Packet Forwarding Between Interior and Exterior Interfaces	9-7
19	9.3.3 Participation in Exterior Networks.....	9-8
20	9.3.3.1 General Functionality	9-8
21	9.3.3.2 Border Routers as Customer Edge Devices.....	9-9
22	9.3.4 Participation in Commissioning.....	9-9
23	9.3.5 Participation in Port Control	9-9
24	9.3.6 Security Considerations.....	9-10
25	9.4 Example of Operation and Role Fulfillment	9-10
26	9.4.1 Functional Roles of a Border Router	9-10
27	9.4.2 Border Router Role Transitions.....	9-11
28	9.4.2.1 Network Formation by the Border Router as Leader	9-12
29	9.4.2.2 Joining with Co-located Commissioner.....	9-13
30	9.4.2.3 Address Assignment, External Routing	9-13
31	9.4.2.4 Petitioning by External Commissioner.....	9-14
32	9.4.2.5 Joining with External Commissioner	9-15
33	9.4.2.6 Transitioning from the Leader Role	9-16
34	9.4.2.7 Multiple Border Routers	9-18

9.1 Introduction

In the context of a **Thread Network**, a **Border Router** is a device that provides connectivity of nodes in the Thread Network to other devices in external networks such as the Internet, local home and building IP networks, or VPNs (Virtual Private Networks) per Figure 9-1.

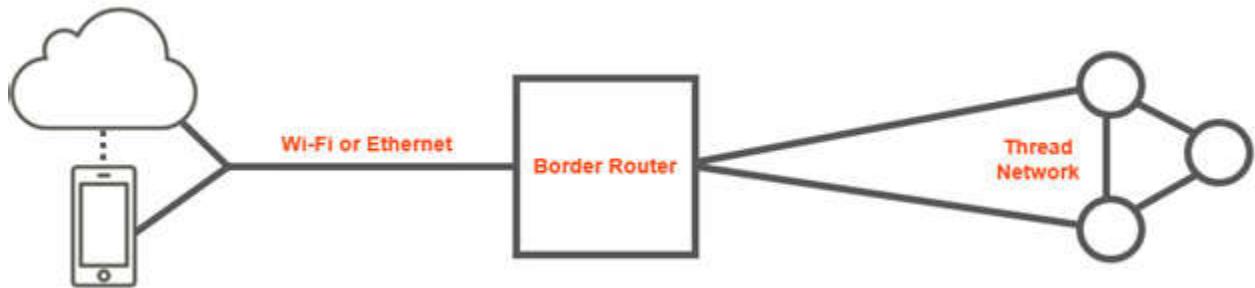


Figure 9-1. High Level Overview of Border Router Role

9.2 Overview

9.2.1 Common Characteristics

The Border Router role will be implemented by devices or products that share certain system characteristics from a networking perspective.

At the **physical and link layers**, a Border Router forms a single system that includes both an IEEE 802.15.4 link-layer interface to be used for the Thread Network as well as at least one supplemental IP link-layer interface used by an exterior network (Wi-Fi or Ethernet being the more common).

At the **network layer**, a Border Router performs standard IP packet routing based on source and destination addresses contained within an IP header:

- Outgoing packets from the Thread Network interface will be forwarded to the exterior interface(s).
- Packets from exterior interface(s) will be forwarded to the Thread Network interface and then routed further in the Thread mesh network toward their end destination.
- Packet filtering or address translation may be performed based on firewall, system, or infrastructure settings.

Also at the network layer, a Border Router MAY participate in an exterior routing protocol, advertise global IPv6 prefixes, and handle global scoped address allocation for nodes within the Thread Network.

At the **transport layer**, a Border Router SHOULD be transparent to end-to-end IP communication.

From a **commissioning** perspective, a Border Router SHALL implement a Mesh Commissioning Protocol **Border Agent** that allows secure, user-initiated joining of new

- 1 devices to the Thread Network or network management functionality by means of a
- 2 Commissioner on an exterior network.
- 3 At the **application layer**, a Border Router MAY provide optional services, such as acting as
- 4 a proxy for service discovery operations on behalf of devices on the Thread Network.

5 **9.2.2 Border Router Availability**

- 6 Communication between devices within the Thread Network can take place without any
- 7 active Border Router participating in the Thread Network. A Thread Network may have
- 8 multiple active Border Routers connecting to one or more external networks. This has the
- 9 advantage of providing redundancy, resilience, and prevents a single point of failure (SPOF)
- 10 for external communication.
- 11 If a Border Router implementing a commissioning Border Agent is not available, external
- 12 Commissioners will be unable to authenticate new Joiner devices or perform network
- 13 management functions.

14 **9.2.3 Global Addresses**

- 15 A Thread Network only operates using IPv6. The use of IPv4 addressing is not supported for
- 16 communication within the Thread Network.
- 17 Individual Thread Network devices support participation in the global IPv6 infrastructure,
- 18 such as being part of the IPv6 Internet. This is achieved by means of GUAs (Global Unicast
- 19 Addresses) that are described in [\[RFC 4291\]](#).
- 20 Each Thread node can be assigned at least one GUA when the upstream infrastructure for
- 21 delegating a global prefix via a Border Router is available.
- 22 The Border Router sends information on the global prefixes it serves to the Thread Leader,
- 23 which adds it to a Network Data set, and then distributes it within the Thread Network.
- 24 Thread Border Routers can obtain global prefix assignments by participating in an exterior
- 25 prefix distribution protocol such as DHCPv6-PD, L2TP-VPN, or HNCP.

26 **9.2.4 Supplemental Unique Local Addresses**

- 27 Thread Network devices support Unique Local IPv6 Unicast Addresses (ULAs) that are
- 28 described in [\[RFC 4193\]](#).
- 29 A Thread Network uses a specific category of ULAs for the purposes of mesh routing and
- 30 management within the network. In the context of the Thread Network, these are called
- 31 MLAs (Mesh Local Addresses)—either ML-EID (Mesh-Local Endpoint Identifier) or ML-RLOC
- 32 (Mesh-Local Routing Locator)—and are identified by a ULA prefix referred to as the MLP
- 33 (Mesh Local Prefix).
- 34 Communication using MLAs is not routable to the exterior of a Thread Network via a Border
- 35 Router.
- 36 Supplemental ULA prefixes MAY be used to assign other ULAs to Thread interfaces. This
- 37 allows creation of IPv6 fabrics spanning multiple in-premises site-local subnets and wide-
- 38 area virtual private networks.

- Such ULAs are useful when either upstream WAN (Wide Area Network) infrastructure does not provide means for IPv6 global prefix delegation or when the application use case either does not need or specifically precludes routing on the Internet.
- Assignment of supplementary ULA prefixes is handled identically to global prefixes from the perspective of how they are provisioned to the Thread Network by the Border Router.
- Supplementary site-local ULA prefixes may be generated by Customer Edge routers adhering to recommendations in [\[RFC 7084\]](#).

9.2.5 Network Data

- Thread interior networks do not use the IPv6 Neighbor Discovery protocol. Global prefixes and Supplementary ULA prefixes are not distributed using Router Advertisement messages. Instead, prefixes are advertised in Network Data messages from the Thread Leader.
- The TMF (Thread Management Framework) protocol is used for notification of Network Data from the Border Routers to the Leader. TMF is based on CoAP (Constrained Application Protocol) that is described in [\[RFC 7252\]](#).
- When forwarding packets with an exterior destination originating from an interior node, regular IP mesh routing within the Thread Network is used to reach the Border Router providing an external route as advertised in the Network Data (see Figure 9-2).

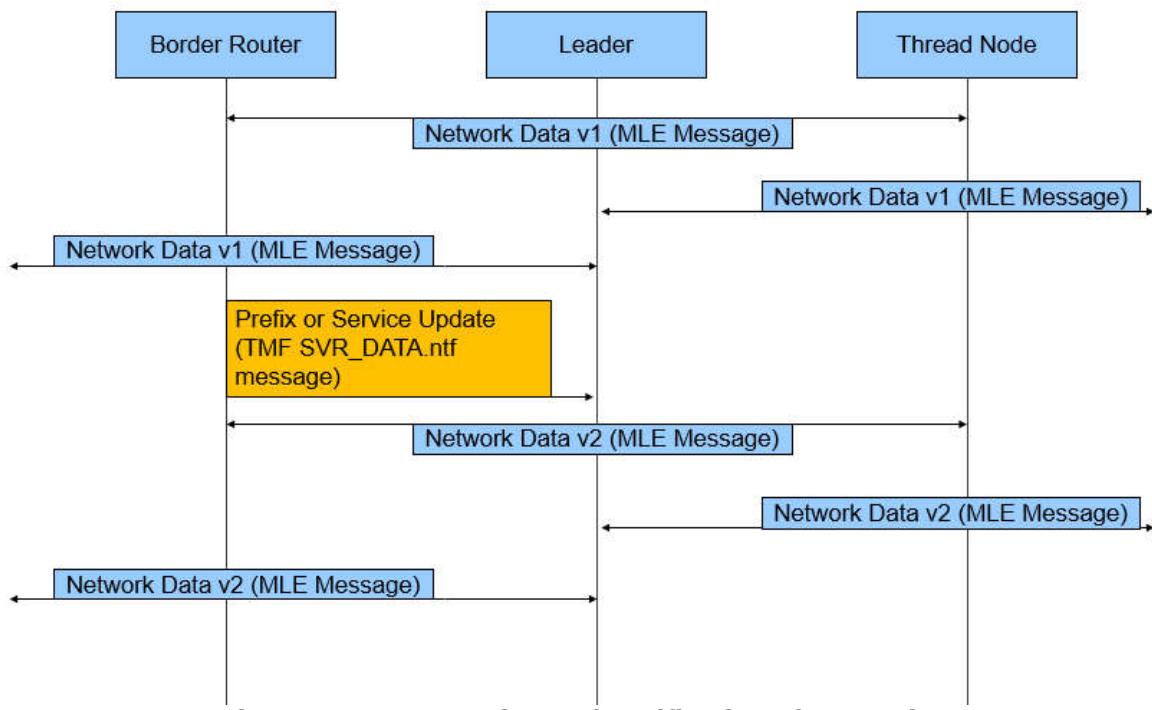


Figure 9-2. Propagation and Notification of Network Data

9.2.6 Coping with IPv4 Infrastructure

- A large existing base of Customer Edge and Customer Premises devices, such as home and small business routers, modems, or access points, either do not provide IPv6 functionality

- 1 or make it complex to configure. These devices provide IPv4-only services and address
2 configuration for their network.
- 3 From the perspective of communication between Thread devices and exterior networks, the
4 Border Routers are responsible for coexistence and coping with IPv4-only site-local subnets
5 in residential networks.
- 6 While less desirable than the usage of native IPv6, in this case Border Routers SHOULD
7 implement a form of NAT (Network Address Translation) so that the application use case
8 remains intact.

9.2.7 Packet Filtering

- 10 Given the limited bandwidth and power constraints of most types of Thread nodes, the
11 Border Router has an essential role in managing packet filtering to avoid intentional or
12 accidental flooding or DoS (Denial of Service) within the Thread Network based on traffic
13 originating from the exterior networks.
- 14 Beyond firewalling to prevent security attacks, Border Routers are expected to provide rate
15 limiting or ultimately deny forwarding of otherwise legitimate ingress application traffic
16 when it is detected that excessive unicast or multicast data flow may be disruptive for the
17 constrained interior network.
- 18 However, such rules SHOULD NOT preclude end-to-end communication of Thread devices
19 with other devices on exterior networks. In particular, Border Router firewall default
20 configurations SHOULD permissively allow Thread nodes to initiate outgoing traffic to
21 exterior end points or servers. The transmission of Thread-initiated outgoing traffic SHOULD
22 also further result in firewall traversal of the associated return traffic that is received in
23 response from the exterior end points. (For more information on default configuration
24 recommendations for reflective session state applying to stateful firewalls, see Section 4.2
25 of [\[RFC 4864\]](#)).

9.2.8 Role in Commissioning

- 27 Border Routers play a role in commissioning of new devices to the Thread Network. A
28 Border Router implements a commissioning Border Agent which acts as a relay for
29 messages between external Commissioners (such as a mobile device) and devices on the
30 interior network. Such devices can be the Leader (which arbitrates between multiple
31 petitioning Commissioners), or the Joiner Router for a new device seeking to join the Thread
32 Network.
- 33 See Chapter 8, Mesh Commissioning Protocol, for more information on the Border Agent
34 role in commissioning.

9.3 Functional Specification

9.3.1 Participation in the Interior Network

- 37 A Border Router MUST implement all interior network functionality defined in the other
38 chapters of this specification which apply to a Thread node. This includes potential Leader
39 capabilities when the node is eligible to act as an interior router.

1 A Border Router MUST adhere to the Border Router and Server(s) normative behavior
2 defined in Chapter 5, Network Layer and Chapter 8, Mesh Commissioning Protocol.

3 It SHOULD be possible to administratively configure a Border Router with regard to the
4 specifics of the services it provides to the interior network.

5 **9.3.1.1 Configuration of Network Data**

6 The Border Router attributes listed in Table 9-1 and Table 9-2 provide recommended
7 mechanisms for the Border Router administrative configuration of Network Data.

8 **Note: The following tables describing attributes are conceptual only. A particular
9 implementation MAY represent the data in other forms if appropriate and MAY
10 choose to allow only a subset of parameters to be administratively configured.**

11 **Table 9-1. Border Router Network Data Attributes**

Attribute	Type	Comment
<i>thrBrPrefixSet</i>	Set of External Prefixes	Table containing an entry for each external prefix for which the Border Router MAY contribute to Network Data.

12

13

Table 9-2. Set of External Prefixes Entry

Subfield	Type	Comment
<i>thrBrPrefixLength</i>	uint8	Prefix length in bits
<i>thrBrPrefixValue</i>	uint8[16]	Prefix value
<i>thrBrPrefixFlags</i>	uint16 bitmap	P_preference (2-bits) P_preferred (1-bit) P_slaac (1-bit) P_dhcp (1-bit) P_configure (1-bit) P_default (1-bit) P_on_mesh (1-bit) P_nd_dns (1-bit) Reserved (7-bits) The flags data are defined in the Border Router TLV of the Thread Network Data.
<i>thrBrPrefixLifetime</i>	uint32	Prefix Lifetime (seconds)
<i>thrBrPrefixAdvertised</i>	Boolean	Flag that, if true, indicates that a Border Router TLV SHALL be advertised for prefix in the Network Data; lifetime duration threshold rules will still be applied to determine the actual advertisement or setting of the P_stable flag respectively

Subfield	Type	Comment
<i>thrBrExternalRouteFlags</i>	uint8 (000000xx)	Value of R_preference as contained in a Has Route TLV of the Thread Network Data
<i>thrBrExternalRouteLifetime</i>	uint32	External Route Data Lifetime (seconds)
<i>thrBrExternalRouteAdvertised</i>	Boolean	Flag that indicates whether a Has Route TLV SHALL be advertised for prefix in the Network Data; lifetime duration threshold rules will still be applied to determine the actual advertisement or setting of the R_stable flag respectively

1 **9.3.1.2 Propagation of Network Data**

2 When administratively configured, the Border Router sends a SVR_DATA.ntf POST message
3 through TMF to the Leader in order to update the Network Data TLVs as described in
4 Chapter 5, Network Layer.

5 **9.3.1.3 DHCPv6 Server for Global EID Assignment**

6 A Border Router MUST provide DHCPv6 server functionality to clients on the interior network
7 for address assignment purposes based on a prefix in the *thrBrPrefixSet* when setting the
8 P_dhcp flag to 1 for the prefix in the *thrBrPrefixFlags* sub-field and when
9 *thrBrPrefixIsAdvertised* is set to True. The DHCPv6 server functionality MUST adhere to
10 [\[RFC 3315\]](#) and [\[RFC 5007\]](#) with the configuration detailed in Chapter 10, Management.

11 Provisioning of globally-scoped EIDs (Endpoint Identifiers) addresses distributed via the
12 DHCPv6 server on a Border Router is out of the scope of this specification and MAY be
13 specific to the server vendor.

14 **9.3.2 Packet Forwarding Between Interior 15 and Exterior Interfaces**

16 A Border Router SHOULD provide IP layer routing services between interior and exterior
17 networks when their respective set of interfaces are configured and active.

18 When a 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) fragment from
19 the interior network reaches a Border Router and it is determined, based on the IPv6 source
20 and destination addresses, that the packet should be forwarded to an exterior network, the
21 Border Router MUST perform necessary 6LoWPAN fragment reassembly and decompression.
22 After the packet is reassembled, the Border Router MUST apply any administratively-
23 configured IP layer address or protocol translation. If the forwarding of the resulted packet
24 conforms to local forwarding and filtering policies, the Border Router MUST then forward the
25 packet onto the exterior network interface.

26 When an IP packet from the exterior network reaches a Border Router and it is determined,
27 based on source and destination addresses and local forwarding and filtering policies, that it
28 should be forwarded to the interior network, the Border Router MUST first apply any
29 administratively-configured IP layer address or protocol translation. The resulting IPv6

1 packet is then subject to 6LoWPAN compression and, if needed, 6LoWPAN fragmentation
2 based on the 6LoWPAN configuration of the interior network. The Border Router will then fill
3 in the 6LoWPAN mesh header for the fragments based on the Thread routing specification
4 and forward each fragment on the interior network 6LoWPAN interface. Fragments are
5 forwarded within the Thread Network Partition based on the Thread routing algorithm and
6 EID-to-RLOC address mapping protocol described in Chapter 5, Network Layer.

7 **9.3.3 Participation in Exterior Networks**

8 **9.3.3.1 General Functionality**

9 While this specification provides a set of references and recommendations on features and
10 services to be provisioned for the participation of a Border Router to exterior networks, it
11 does not generally mandate a specific configuration for such exterior networks or interfaces.
12 This allows Border Routers to be deployed in a wider set of exterior infrastructure
13 configurations that support vendor-specific features and differentiation while preserving
14 consistency for interior Thread Network behavior.

15 A Border Router SHOULD at a minimum act as an IPv4 host when participating in exterior
16 networks that support IPv4.

17 If a Border Router is participating in an exterior network that consists of a site-local IPv4
18 subnet, it MUST act as a DHCPv4 client for DHCPv4 servers in the subnet under user control
19 for configuring the host as specified in [\[RFC 1918\]](#) and [\[RFC 2131\]](#).

20 If a Border Router is participating in an exterior network that consists in a site-local IPv4
21 subnet, it MUST enable participation in link-local multicast for the subnet and provide
22 multicast Domain Name System (mDNS) and DNS Service Discovery (DNS-SD) services for
23 Thread Commissioning as specified in Chapter 8, Mesh Commissioning Protocol and
24 [\[RFC 6761\]](#), [\[RFC 6762\]](#), and [\[RFC 6763\]](#).

25 A Border Router SHOULD at a minimum act as an IPv6 host when participating in exterior
26 networks that support IPv6.

27 If a Border Router is participating in an exterior network that supports IPv6, it MUST
28 implement [\[RFC 4861\]](#) and [\[RFC 4862\]](#) for host configuration on the exterior network. It
29 MAY also adhere to [\[RFC 6434\]](#) and its normative references.

30 If a Border Router is participating in an exterior network that consists in a site-local IPv6
31 subnet, it MUST enable link-local multicast for the subnet and provide mDNS and DNS-SD
32 services for Thread Commissioning as specified in Chapter 8, mesh Commissioning Protocol
33 and [\[RFC 6761\]](#), [\[RFC 6762\]](#), and [\[RFC 6763\]](#).

34 When participating in exterior networks that support IPv6, a Border Router SHOULD
35 participate in IPv6 prefix delegation with the purpose of distributing the available prefixes
36 downstream for configuring the interior network hosts. At a minimum, a Border Router
37 SHOULD act as a DHCPv6-PD client in the exterior IPv6 network as described in [\[RFC 3633\]](#).

38 When participating in exterior networks that support IPv4, a Border Router SHOULD enable
39 a NAT64 service for address translation of IPv6 packets originating from or IPv4 packets
40 addressed to interior network hosts.

41 When a NAT64 service is available, a Border Router SHOULD perform stateless address and
42 protocol version translation as described in [\[RFC 6144\]](#) and [\[RFC 6145\]](#).

43 When a NAT64 service is available, a Border Router MAY perform stateful address and
44 protocol version translation as described in [\[RFC 6146\]](#).

- 1 A Border Router MAY optionally act as an IP router between multiple exterior IP domains. It
2 MAY participate in such exterior routing domains with a routing protocol such as RIP
3 (Routing Information Protocol), OSPF (Open Shortest Path First), IS-IS (Intermediate
4 System-to-Intermediate System), or others as needed by the exterior network
5 configuration.
- 6 [\[RFC 7368\]](#) details supplemental architectural principles, considerations and constraints
7 related to the deployment of IPv6 devices and addressing infrastructure for residential home
8 networks with optional applicability to Thread Border Routers.

9 9.3.3.2 Border Routers as Customer Edge Devices

10 A subset of Border Routers MAY also act as Customer Edge Devices that connect end-user
11 or home networks to a service provider infrastructure. In that role, recommendations in
12 [\[RFC 7084\]](#) MAY be applied to the functionality of the node on the exterior networks
13 (applying both to upstream infrastructure as well as downstream LANs (Local Area
14 Networks)).

15 9.3.4 Participation in Commissioning

16 A Border Router MUST participate in Thread new device commissioning by implementing a
17 Border Agent that relays commissioning data between a Commissioner which may be
18 connected to an exterior network and the Leader, Joiner, and Joiner Router nodes on the
19 interior network.

20 When participating in commissioning, a Border Router MUST implement all normative
21 Border Agent functionality for its role as specified in Chapter 8, Mesh Commissioning
22 Protocol.

23 When relaying Thread commissioning data, the Border Router is an IP layer endpoint for
24 both interior and exterior sides of the relayed communication. As such, the Border Router
25 MUST provide the subset of TMF CoAP infrastructure services over UDP (User Datagram
26 Protocol) described in Chapter 8, Mesh Commissioning Protocol, and based on [\[RFC 7252\]](#).

27 A Border Router deployed in the same exterior site-local subnet with a Commissioner
28 SHOULD advertise its endpoint to be discovered through mDNS and DNS-SD as specified in
29 Chapter 8, Mesh Commissioning Protocol. A Commissioner MAY also use a vendor-specific
30 mechanism for Border Agent discovery within the exterior network fabric (for example, by
31 means of a dedicated application or Internet server that facilitates the discovery process).

32 9.3.5 Participation in Port Control

33 A Border Router MAY use the Port Control Protocol set as detailed in [\[RFC 6887\]](#),
34 [\[RFC 6970\]](#), [\[RFC 7225\]](#), [\[RFC 7291\]](#), and [\[RFC 7488\]](#).

35 The PCP (Port Control Protocol) allows an IPv6 or IPv4 host to control how incoming IPv6 or
36 IPv4 packets are translated and forwarded by an NAT or a simple firewall, and also allows a
37 host to optimize its outgoing NAT keep-alive message.

38 In the PCP client role, a Border Router MAY configure packet and port forwarding in
39 upstream customer edge devices running PCP servers on behalf of nodes in the interior
40 network.

41 A Border Router MAY act in a PCP server role to facilitate packet and port forwarding for
42 interior network nodes that implement PCP clients.

- 1 With similar purposes, a Border Router MAY also participate in other filtering, forwarding, or
2 endpoint discovery protocols in vendor-specific configurations.

3 **9.3.6 Security Considerations**

- 4 A Border Router MUST implement the generic Thread security features as specified in
5 Chapter 7, Security, related to the use of MAC (Media Access Control) and MLE (Mesh Link
6 Establishment) security in the interior network and DTLS (Datagram Transport Layer
7 Security) over UDP (User Datagram Protocol) in the context of Thread Commissioning for
8 both interior and exterior networks
- 9 A Border Router MUST also implement the security provisioning for Commissioning relay for
10 an exterior network Commissioner as specified in Chapter 8, Mesh Commissioning Protocol.
- 11 As a customer premises equipment providing IPv6 service, a Border Router SHOULD adhere
12 to recommendations and normative references specified in [\[RFC 4949\]](#), [\[RFC 6092\]](#), and
13 [\[RFC 6434\]](#).

14 **9.4 Example of Operation and Role 15 Fulfillment**

16 **9.4.1 Functional Roles of a Border Router**

17 In the context of operation within the interior network, the Border Router provides a set of
18 discrete services and functions. Some are related to the multiple interface capability, while
19 others are orthogonal to it.

20 Some services are optional at runtime because the network conditions or system
21 configuration can dynamically enable or disable these functions and services. Table 9-3
22 summarizes these services, along with the events that activate or de-activate the functions.

23 **Table 9-3. Border Router Roles and Functions**

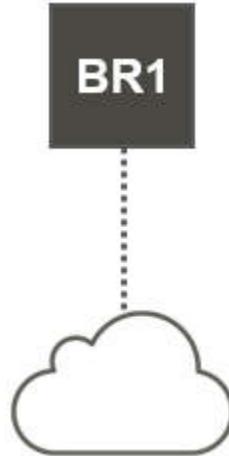
Border Router Functional Role	Role Change Events & Conditions
Server for external prefixes	Role can be enabled, disabled, or configured by a high-level application on the Border Router. The high-level application configures Thread Network Data by means of the <i>thrBrPrefixSet</i> to indicate SLAAC or DHCPv6 external prefixes that the interior network can use to assign global or supplemental ULA addresses.
Server for external route	Role can be enabled, disabled, or configured by a high-level application on the Border Router. The high-level application configures Thread Network Data by means of the <i>thrBrPrefixSet</i> to indicate that the Border Router provides an external route for source addresses using external prefixes.
Commissioning Border Agent	Managed by the Thread Network stack. Configured by a high-level application.

Border Router Functional Role	Role Change Events & Conditions
Leader	Managed autonomously by the Thread Network stack (orthogonal to server for external services role).
Commissioner	Managed by a high-level application on the Border Router. Must petition Leader and get favorable outcome to enter role (orthogonal to server for external services role).
Joiner Router	Managed by the Thread Network stack.
Packet and port filtering	Managed by a high-level application on the Border Router.

1 **9.4.2 Border Router Role Transitions**

2 The following sections detail a typical Thread Network formation and subsequent operation
3 from the perspective of a Border Router. Initially, several functions and roles are collapsed
4 to the same device that is used to form a new Thread Network and then acts as a Leader.
5 As more devices join the Thread Network and the user chooses exterior Commissioners,
6 some of the roles become distributed.

1 9.4.2.1 Network Formation by the Border Router as 2 Leader



3
4 **Figure 9-3. BR1 starts a Thread Network as Leader, activates external interface.**

5 Node Border Router 1 (BR1) has both a Thread and an external network interface available
6 and can act as a Border Router. However, as it is essentially the single device in the Thread
7 Network, minimal Border Router functions are active at this point.

8 It is expected that the exterior interface becomes configured and provisioned for access to
9 the home network (LAN or WLAN) and the Internet. For simplicity, this scenario assumes
10 the exterior interface is also used to configure delegation of a globally scoped /64 IPv6
11 prefix for node address assignment on the Thread Network. These configurations are added
12 to the *thrBrPrefixSet*.

13 **Border Router exercises roles:** Leader, Commissioning Border Agent, Joiner Router

1 9.4.2.2 Joining with Co-located Commissioner



2

3 **Figure 9-4. BR1 acts as an internal Commissioner for joining R1.**

4 A user initiates joining of a new node (R1) to the network. R1 only has a Thread interface
5 available for communication. Assuming BR1 is provided with an adequate UI (User
6 Interface), it may also be used to act as an on-mesh Commissioner. The commissioning
7 session is native to the Thread Network and no data flows across the Thread Network
8 border are necessary to finalize commissioning and joining of R1.

9 **Border Router exercises roles:** Leader, Commissioner, Joiner Router, and Commissioning
10 Border Agent

11 9.4.2.3 Address Assignment, External Routing

12 After R1 joins the Thread Network, it becomes a Router and will start receiving routing and
13 MLE advertisements containing Network Data from BR1 acting as a Leader. The Leader and
14 the Server for external prefix network data are collapsed on BR1. R1 then uses the
15 information in the incoming advertisements its EID addresses based on external global or
16 supplemental ULA prefixes.

17 R1 can subsequently use the globally scoped addressing and the external route via BR1 to
18 interact with nodes on the home WLAN or the Internet at the application layer.

19 **Border Router exercises roles:** Leader, Server for external prefixes, Server for external
20 route, Packet and port filtering, Joiner Router, Commissioning Border Agent

1 9.4.2.4 Petitioning by External Commissioner

2



3 **Figure 9-5. External Commissioner registers with BR1.**

- 4 A user initiates petitioning for a new Commissioner (C) external to the Thread network. The
5 new Commissioner can be a mobile device connected to the same WLAN network segment
6 or another device that can otherwise establish a connection to the exterior interface of BR1.
7 The network credential is used to complete a DTLS handshake between C and BR1 that
8 opens a secure commissioning session between the nodes. As BR1 also exercises the Leader
9 role, the message flow for petitioning takes place only on the external network.
10 This scenario assumes that BR1, in its Leader role, successfully registers the new device as
11 a Commissioner. BR1 will subsequently advertise itself within the Thread Network as a
12 Border Agent for relay purposes.
13 **Border Router exercises roles:** Leader, Commissioning Border Agent, Joiner Router

1 9.4.2.5 Joining with External Commissioner

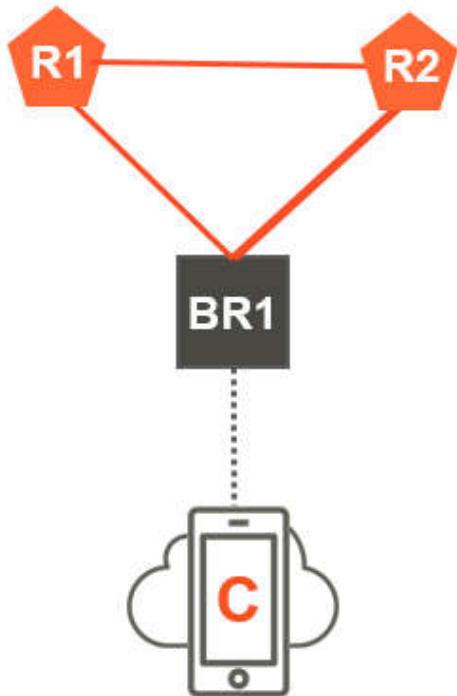


Figure 9-6. BR1 relays commissioning for C to allow R2 to join.

4 A user initiates joining of a new node (R2) to the network. R2 only has a Thread interface
5 available for communication. External Commissioner (C) is used after C completes
6 petitioning with BR1 as described in the previous subsection.

7 Based on Thread Network topology, either BR1 or R1 can be selected by the Joiner as a
8 Joiner Router. In the simplest case, when BR1 is chosen, an initial DTLS handshake
9 message is sent only from R2 to BR1. Otherwise, if R1 is chosen as the Joiner Router, the
10 DTLS handshake message between R2 and BR1 is forwarded by R1 by means of TMF
11 messages within the secured Thread Network.

12 BR1 further relays the DTLS handshake from the Joiner to the Commissioner by means of
13 messages exchanged across the secure commissioning session established after petitioning
14 on the external network.

15 If the Commissioner-to-Joiner handshake completes successfully and BR1 receives Entrust
16 authorization from the Commissioner for the Joiner, BR1 will use the resulting Key
17 Encryption Key to securely distribute network parameters and the master key to Joiner R2.

18 Once R2 joins, it will also receive Network Data advertisements from BR1 acting as a Leader
19 and can assign global scoped address.

Border Router exercises roles: Leader, Server for external prefixes, Server for external route, Packet and port filtering, Joiner Router, Commissioning Border Agent

1 9.4.2.6 Transitioning from the Leader Role



2

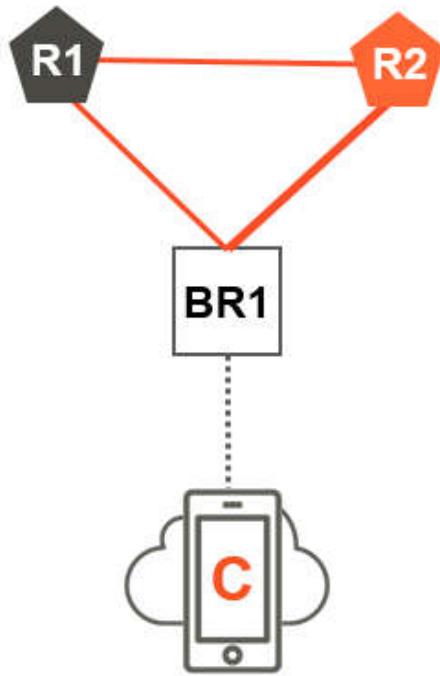
3 **Figure 9-7. BR1 becomes unavailable. R1 takes over Leader role.**

4 A user may temporarily power off device BR1 (for example, for maintenance purposes). As
5 BR1 is also the Leader, when it is no longer active in the network for a defined time interval,
6 another router takes over the Leader role.

7 This scenario assumes R1 becomes the new Leader. When a new Leader starts, the Network
8 Data is reset. Without a Border Router active in the Thread Network, the new Network Data
9 advertised by R1 also no longer contains information for external routing. Thread Network
10 devices can continue to be temporarily addressed using the global scope prefix addresses
11 that were previously assigned.

12 Powering BR1 back on results in BR1 synchronizing and acknowledging R1 as the new
13 Leader and subsequently BR1 uses the SVR_DATA.ntf TMF message to dispatch its global
14 prefix information to the Leader R1. R1 starts propagating the Network Data advertising
15 BR1 services to the network. Routing to the exterior network is restored.

16 **Border Router exercises roles:** Not applicable



1
2

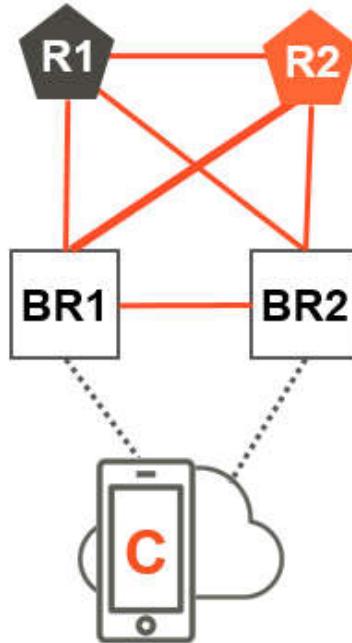
Figure 9-8. BR1 returns to the network. R1 remains Leader.

3 If joining a new device to the network becomes necessary, a new petitioning session needs
4 to establish a Commissioner. The new Leader (R1) is used for the petitioning arbitration
5 between multiple potential Commissioners.

6 BR1 uses TMF messages to interact with R1. BR1 acts as a Border Agent on behalf of
7 Commissioners on the external network, or to register itself as an on-mesh Commissioner.
8 Commissioning then proceeds as described in the previous subsections.

9 **Border Router exercises roles:** Server for external prefixes, Server for external route,
10 Packet and port filtering, Joiner Router, Commissioning Border Agent

1 9.4.2.7 Multiple Border Routers



2
3 **Figure 9-9. BR2 joins the network. Both BR1 and BR2 provide external routing.**

4 Assume a new device (BR2) with border routing capabilities joins the Thread Network. BR2
5 is connected to the same exterior LAN segment as BR1.

6 Both BR1 and BR2 will use TMF to send their information to the Leader so the Network Data
7 includes them as part of the Prefix and External Route sets for the external prefix.

8 **BR1 exercises roles:** Server for external prefixes, Server for external route, Packet and
9 port filtering, Joiner Router, Commissioning Border Agent

CHAPTER 10 MANAGEMENT

Contents

3	10.1 Introduction.....	10-3
4	10.2 Framework Overview	10-3
5	10.3 URIs and Notation.....	10-3
6	10.3.1 URIs	10-3
7	10.3.2 CoAP Request Notation	10-4
8	10.3.3 CoAP Response Notation	10-5
9	10.4 CoAP Message Format.....	10-5
10	10.4.1 Header Format.....	10-5
11	10.4.2 CoAP Options.....	10-5
12	10.4.2.1 Mandatory Options	10-5
13	10.4.2.2 Options Not Recommended.....	10-5
14	10.4.3 Payload Format.....	10-6
15	10.4.4 CoAP Response Codes.....	10-7
16	10.5 CoAP Parameters	10-7
17	10.6 CoAP Message Delivery	10-7
18	10.6.1 Reliability	10-7
19	10.6.2 TMF IPv6/UDP Addressing Rules.....	10-8
20	10.6.3 TMF Received Message Processing.....	10-8
21	10.7 TMF Message Types.....	10-8
22	10.7.1 TMF Request.....	10-9
23	10.7.2 TMF Response.....	10-9
24	10.7.3 TMF Query	10-9
25	10.7.4 TMF Answer.....	10-9
26	10.7.5 TMF Notification	10-9
27	10.7.6 CoAP Mapping.....	10-9
28	10.8 TMF Message Transaction Patterns	10-10
29	10.8.1 Unicast TMF Messages	10-10
30	10.8.2 Multicast TMF Messages	10-10
31	10.8.3 Token Usage.....	10-11
32	10.8.4 CoAP Request/Response Pairing	10-11
33	10.8.5 TMF Request, Piggybacked TMF Response	10-11
34	10.8.6 TMF Request, Separate TMF Response	10-12

1	10.8.6.1 Example Scenario	10-12
2	10.8.7 Confirmable TMF Notification, Query or Answer, Piggybacked Dummy Response	10-13
3	10.8.8 Non-confirmable TMF Notification, Query or Answer.....	10-13
5	10.9 Commissioner Transactions	10-14
6	10.9.1 Native or External Commissioner with Thread Device	10-14
7	10.9.1.1 Destined for Thread Device	10-15
8	10.9.1.2 Originating from Thread Device	10-16
9	10.9.2 Native or External Commissioner with Border Agent	10-17
10	10.9.3 On-mesh Commissioner	10-17
11	10.10 Message Mapping.....	10-18
12	10.10.1 Address Resolution API (/a/...)	10-18
13	10.10.2 MeshCop API (/c/...).....	10-19
14	10.10.3 Diagnostic API (/d/...).....	10-21
15	10.11 Network Diagnostics.....	10-21
16	10.11.1 Overview	10-21
17	10.11.1.1 Device Diagnostics	10-21
18	10.11.1.2 Network Diagnostics	10-21
19	10.11.1.3 Link Diagnostics	10-22
20	10.11.2 Diagnostic Commands	10-22
21	10.11.2.1 DIAG_GET.req – Get Diagnostic Request.....	10-22
22	10.11.2.2 DIAG_GET.rsp – Get Diagnostic Response	10-22
23	10.11.2.3 DIAG_GET.qry – Get Diagnostic Query.....	10-23
24	10.11.2.4 DIAG_GET.ans – Get Diagnostic Answer.....	10-23
25	10.11.2.5 DIAG_RST.ntf – Reset Diagnostic Notification	10-23
26	10.11.3 Diagnostic TLVs Overview	10-24
27	10.11.4 Diagnostic TLVs	10-25
28	10.11.4.1 MAC Counters TLV (9)	10-25
29	10.11.4.2 Battery Level TLV (14).....	10-26
30	10.11.4.3 Supply Voltage TLV (15)	10-26
31	10.11.4.4 Child Table TLV (16)	10-26
32	10.11.4.5 Channel Pages TLV (17).....	10-27
33	10.11.4.6 Type List TLV (18).....	10-27
34	10.11.4.7 Max Child Timeout TLV (19)	10-28
35	10.12 Persistent Data	10-28
36	10.13 IANA Considerations.....	10-29

10.1 Introduction

2 Managing Thread Networks requires many messages in a variety of circumstances, including
3 commissioning, aggregating and distributing shared network data, channel changes and
4 PAN (Personal Area Network) ID changes. This chapter describes the TMF (Thread
5 Management Framework) and how the CoAP (Constrained Application Protocol) [\[RFC 7252\]](#)
6 is used for TMF messages.

10.2 Framework Overview

8 TMF messages use CoAP requests and CoAP responses. TMF messages that use a CoAP
9 request may be unicast or multicast. TMF messages that use a CoAP response are always
10 unicast. TMF transactions are based on a defined set of transaction patterns (defined in
11 Section 10.8, TMF Message Transaction Patterns). Each transaction pattern is defined as a
12 sequence of specific CoAP message types. An implementation SHOULD NOT allow the use of
13 the Thread Management Port by any application other than Thread management as defined
14 in this specification.

10.3 URIs and Notation

10.3.1 URIs

17 For CoAP requests, this specification uses URIs (Uniform Resource Identifiers) that:

- 18 • Use the 'coap' or 'coaps' scheme notation.
 - 19 ▪ `coap` means unsecured at transport layer, leveraging Thread MAC security.
 - 20 ▪ `coaps` means secured at transport layer using DTLS as defined in Chapter 7,
21 Security.
- 22 • Use the destination IPv6 address as the URI authority.
- 23 • Use a Thread-specific destination UDP port or the Commissioning Session source port
24 (`<CSSP>`). A Thread-specific destination UDP port is abbreviated using two capital
25 letters. The abbreviation indicates either the default number, or a non-default port
26 number that has been advertised. Both unsecured (`coap`) and secured (`coaps`)
27 CoAP messages may be received by a CoAP server at the same port. The port can be
28 one of:
 - 29 ▪ Management Port: `:MM`
 - 30 ▪ Commissioner Port: `:MC`
 - 31 ▪ Joiner Port: `:MJ`
- 32 • The Commissioning Session source port is the ephemeral source port used when
33 setting up the Commissioning Session using DTLS. The destination port for the
34 Commissioning Session is always `:MC`.

- 1 • Have a URI path of the format /*x*/*y*, where '*x*' is a single character denoting one of
2 the three TMF APIs. Each TMF API defines its own distinct Type-Length-Value (TLV)
3 schema. The three TMF APIs are:
 - 4 ■ Address resolution: /a/...
 - 5 ■ MeshCoP: /c/...
 - 6 ■ Diagnostic: /d/...
- 7 • Indicate the message type (as a string '*y*' in the above URI path format /*x*/*y*).

8 As text, these look like:

9 coap://[<IPv6 address>]:<port>/<TMF API>/<message type>
10 coaps://[<IPv6 address>]:<port>/<TMF API>/<message type>

11 For example the following URI definition in this specification:

12 coap://[<Leader-IP-address>]:MM/c/ms

13 could translate to an actual URI as follows:

14 coap://[fd00:ff1:ce0b:a5e0:0:ff:fe00:0]:61631/c/ms

15 where

- 16 • ':MM' indicates that the destination port is the Thread management port
- 17 • '"c"' indicates the MeshCoP API is used
- 18 • '"ms"' indicates the MeshCoP message is a MGMT_SET.req TMF message

19 **10.3.2 CoAP Request Notation**

20 In this specification a TMF message that uses a CoAP request is defined using the following
21 template format:

22 CoAP Request URI

23 <Request URI in the format defined above>

24 Transaction Pattern

25 <Name(s) of transaction pattern(s) that may be used with the message>

26 CoAP Payload

27 <List of TLV(s) sent as payload within the request or 'none' if none>

28 <Payload Parameter 1>

29 <Description of first parameter in payload. Optional; present if there is a payload.>

30 <Payload Parameter *n*>

31 <Description of *n*th parameter in payload TLVs. Optionally present.>

32 **Note: A TLV that is optional in the payload is notated with square brackets around
33 the TLV name.**

34 **Note: The CoAP method used in the request is not indicated in this notation as only
35 POST is used currently.**

10.3.3 CoAP Response Notation

2 A TMF message that uses a CoAP response is defined using the following template format:

3 CoAP Response Code

4 <Default CoAP response code for the TMF message> <Name of response code>

5 [<Optional other response code(s) for non-default cases can be listed.>]

6 Transaction Pattern

7 <Name(s) of transaction pattern(s) that may be used with the message>

8 CoAP Payload

9 <List of TLV(s) sent as payload within the response or 'none' if none>

10 <Payload parameter 1>

11 <Description of first parameter in payload. Optional; present if there is a payload.>

12 <Payload parameter n>

13 <Description of nth parameter in payload TLVs. Optionally present>

14 **Note: A TLV that is optional in the payload is notated with square brackets around the TLV name.**

10.4 CoAP Message Format

10.4.1 Header Format

18 The CoAP message header format is as shown in Figure 7 in [[RFC 7252](#)].

19 All CoAP requests used in a TMF message MUST use the POST method. Token usage is
20 defined in Section 10.8.2, **Multicast TMF Messages**.

10.4.2 CoAP Options

22 The CoAP header options and their formats are defined in [[RFC 7252](#)] Section 3 and Section
23 5.4. Any CoAP Option not listed in this section MUST NOT be used in TMF messages.

10.4.2.1 Mandatory Options

25 The Uri-Path Option (11) MUST BE used to construct the URI Path of a CoAP request. This
26 option appears twice to encode the URI path of the form /x/y.

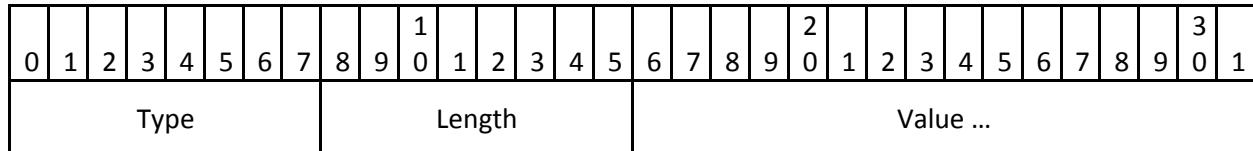
10.4.2.2 Options Not Recommended

28 The Content-Format Option (12) SHOULD NOT by default be included in a TMF message that
29 carries a payload. The reason is that the payload of a TMF Message consists of TLVs,
30 therefore it can always be inferred from context. An exception to the previous rule is the
31 specific case of a CoAP 4.xx/5.xx error response containing a TLV payload, where the
32 Content-Format Option is used to distinguish the response payload from a plain-text CoAP
33 diagnostic payload (Section 5.5.2 of [[RFC 7252](#)]). If a Content-Format Option is

1 nevertheless included, its value MUST be set to application/octet-stream (42). Future
2 versions of this specification may use other content formats.

3 **10.4.3 Payload Format**

4 Payload formatting is a simple TLV (Type-Length-Value) format. There are no padding or
5 alignment requirements for TLVs. The fields in the TLV are sent in network byte order.
6 There are two TLV formats, a base format and an extended format. The base TLV format is
7 shown in Figure 10-1.



8 **Figure 10-1. Base TLV Format**

9 Type

10 Identifies the attribute the TLV is associated with.

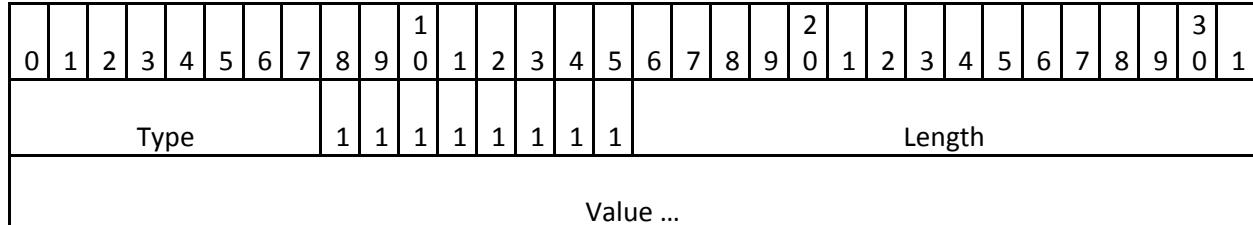
11 Length

12 This field is one octet and indicates the number of octets in the Value field. The length of
13 the Type and Length fields are not counted in the Length value.

14 Value

15 This field is zero or more octets and contains the value of the attribute indicated by the
16 Type. The format of the Value field is determined by the Type field. The length of the
17 Value field is contained in the TLV Length field.

18 The extended format allows for longer TLVs and is shown in Figure 10-2.



19 **Figure 10-2. Extended TLV Format**

20 Type

21 Identifies the attribute the TLV is associated with.

22 Length

23 This field is two octets and indicates the number of octets in the Value field. The length
24 of the Type, the escape (0xFF) and Length fields are not counted in the Length value.

25 Value

26 This field is zero or more octets and contains the value of the attribute indicated by the
27 Type. The length of the Value field is contained in the TLV Length field.

28 To allow for future extensions, when processing incoming TLVs defined as having a fixed
29 Length in this version of the specification, then the TLV MUST NOT be discarded and the

1 Value field(s) spanning the specified fixed Length MUST still be processed as valid even
2 when the actual incoming TLV Length is larger than the specified fixed Length.

3 **10.4.4 CoAP Response Codes**

4 In generating a TMF Response, only the CoAP response codes listed below MAY be used by a
5 CoAP server. Other response codes than those listed in this section SHOULD NOT be
6 returned by a CoAP server.

7 2.xx success codes:

- 8 • 2.04 Changed – this is the default success response.

9 4.xx client error codes:

- 10 • 4.00 Bad Request – returned in the case where the request is malformed or contains
11 an error other than those covered by the below 4.xx error codes.

12 • 4.02 Bad Option – returned in the case where a CoAP Option of class “critical” is not
13 recognized by the server. See [[RFC 7252](#)] for more information.

14 • 4.04 Not Found – MUST be returned in the case where the request refers to a non-
15 existent URI path, for example, if the CoAP server does not recognize an incoming
16 TMF request.

17 • 4.05 Method Not Allowed – returned in the case where an invalid CoAP method is
18 used, for example if a method other than POST is invoked using a TMF API, or an
19 unrecognized method code that is not in the CoAP specification is used.

20 • 4.13 Request Entity Too Large – returned in the case where the server could not
21 wholly process the CoAP request because the message was too large, for example,
22 the server’s buffer size is not large enough to hold the entire request.

23 5.xx server error codes:

24 • 5.00 Internal Server Error – returned in the case where a server error occurred
25 during processing of the CoAP request that caused the request processing to be
26 aborted.

27 • 5.03 Service Unavailable – returned in the case where a server does not have
28 resources currently to process the request.

29 **10.5 CoAP Parameters**

30 Thread uses the default CoAP parameters listed in Section 4.8 of [[RFC 7252](#)], except for
31 those listed below where different values are used.

- 32 • PROBING_RATE: 100 byte/second

33 **10.6 CoAP Message Delivery**

34 **10.6.1 Reliability**

35 TMF messages can be delivered reliably (confirmable, or CON) or unreliably (non-
36 confirmable, or NON). The CoAP messaging layer specifies the use of a CON/ACK message

- 1 pair for confirmable delivery and the use of a single NON message for non-confirmable
2 delivery. The use of either confirmable or non-confirmable delivery is defined for each TMF
3 transaction pattern.
- 4 For confirmable delivery, duplicate detection as described in [[RFC 7252](#)] MUST be applied at
5 the CoAP server side. The reason is that many TMF messages create significant state
6 change or trigger significant resource usage at the receiving Thread device. Therefore, it is
7 undesirable for a CoAP client to time out and retransmit its request to the CoAP server one
8 or more times in the case of an activity such as Router ID allocation. Duplicate detection is
9 described in more detail in [[RFC 7252](#)] Section 4.5.

10.6.2 TMF IPv6/UDP Addressing Rules

11 A TMF CoAP message sent to the Thread Management UDP port (:MM) MUST comply with at
12 least one of these addressing rules:

- 13 • The IPv6 source address is an RLOC or ALOC.
14 • The IPv6 destination address is an RLOC or ALOC.
15 • The IPv6 destination address is a Link-Local address.

16 Some of the individual TMF Message definitions will also impose specific, stricter rules for
17 addressing.

10.6.3 TMF Received Message Processing

19 An implementation SHOULD ensure that the default UDP port number for Thread
20 Management (:MM) is exclusively available to Thread Management functions as defined in
21 this specification. However, in rare cases, this port is already bound to an application
22 process or an application process needs to bind to the :MM port number in addition to
23 Thread Management.

24 For any cases where the port :MM is shared with an application process, the following
25 processing rule MUST be followed for any UDP packet towards the port :MM received on a
26 Thread interface:

27 IF: [(IPv6 source address is one of RLOC/ALOC) **OR**
28 (IPv6 destination address is one of RLOC/ALOC) **OR**
29 (IPv6 destination address is Link-Local Scope)]
30 THEN: process UDP packet by Thread Management
31 ELSE: dispatch UDP packet to Application process

10.7 TMF Message Types

33 There are five types of TMF message:

- 34 • TMF Request (.req)
35 • TMF Response (.rsp)
36 • TMF Query (.qry)
37 • TMF Answer (.ans)
38 • TMF Notification (.ntf)

10.7.1 TMF Request

2 A TMF Request is sent by a requester. A TMF Request is a unicast message from a requester
3 addressed specifically to a target responder for the purpose of soliciting exactly one TMF
4 Response from the target responder.

10.7.2 TMF Response

6 A TMF Response is sent by a responder. A TMF Response is a unicast message from a responder
7 constituting a solicited response to a TMF Request.

10.7.3 TMF Query

9 A TMF Query is sent by an inquirer. A TMF Query is a unicast or multicast message from an
10 inquirer for the purpose of eliciting TMF Answers from one or more answerers. A TMF Query
11 may result in no elicited TMF Answers.

10.7.4 TMF Answer

13 A TMF Answer is sent by an answerer. A TMF Answer is a unicast message from an answerer
14 for the purpose of answering a TMF Query from an inquirer.

10.7.5 TMF Notification

16 A TMF Notification is sent by a notifier. A TMF Notification is a unicast or multicast message
17 from a notifier which does not solicit any response or elicit any answer and thus can be
18 considered a one-way transaction.

10.7.6 CoAP Mapping

20 TMF messages are mapped to CoAP messages according to Table 10-1.

21 **Table 10-1. TMF to CoAP Message Mapping**

TMF message type	Initiator	CoAP message type	Role of Initiator
TMF Request	Requester	CoAP request	CoAP client
TMF Response	Responder	CoAP response	CoAP server
TMF Query	Inquirer	CoAP request	CoAP client
TMF Answer	Answerer	CoAP request	CoAP client
TMF Notification	Notifier	CoAP request	CoAP client

22 All TMF Requests MUST use the POST method. The use of other methods is not required as
23 the RESTful architectural style is not being used.

10.8 TMF Message Transaction Patterns

2 There are four CoAP-based transaction patterns used in conjunction with TMF messages.
3 These are shown in Table 10-2.

4 **Table 10-2. TMF Transaction Patterns**

Transaction Pattern	Abbreviation
TMF Request, piggybacked TMF Response	Req+Rsp_Piggybacked
TMF Request, separate TMF Response	Req+Rsp_Separate
Confirmable TMF Notification, Query or Answer, piggybacked dummy response	Ntf/Qry/Ans_CON
Non-confirmable TMF Notification, Query or Answer	Ntf/Qry/Ans_NON

5 A TMF Request is always sent as a CoAP request using a CON CoAP message.

10.8.1 Unicast TMF Messages

7 The transaction patterns used in conjunction with unicast TMF messages are shown in Table
8 10-3.

9 **Table 10-3. Unicast TMF Message Transaction Patterns**

TMF Message	Transaction pattern(s)
TMF Request	Req+Rsp_Piggybacked Req+Rsp_Separate (request part)
TMF Response	Req+Rsp_Piggybacked Req+Rsp_Separate (response part)
TMF Query	Ntf/Qry/Ans_CON
TMF Answer	Ntf/Qry/Ans_CON
TMF Notification	Ntf/Qry/Ans_CON Ntf/Qry/Ans_NON

10.8.2 Multicast TMF Messages

11 The transaction patterns used in conjunction with multicast TMF messages are shown in
12 Table 10-4.

1 **Table 10-4. Multicast TMF Message Transaction Patterns**

TMF Message	Transaction pattern
TMF Query	Ntf/Qry/Ans_NON
TMF Notification	Ntf/Qry/Ans_NON

2

10.8.3 Token Usage

3 A zero-length CoAP Token MAY be used for Ntf/Qry/Ans_CON and Ntf/Qry/Ans_NON. A non-
4 zero length CoAP Token MUST be used for Req+Rsp_Piggybacked and Req+Rsp_Separate.

5

10.8.4 CoAP Request/Response Pairing

6 CoAP generally specifies the use of a CoAP request/response pair for all transactions due to
7 the RESTful architectural style recommended for CoAP.

8 Req+Rsp_Piggybacked and Req+Rsp_Separate map directly to a CoAP request/response
9 pair.

10 In the case of Ntf/Qry/Ans_CON, the ACK sent in reply has a piggybacked dummy CoAP
11 response with code 2.04 ('Changed') instead of being empty (code 0.00). This has three
12 purposes:

- 13 • To comply with the general specification to use CoAP request/response pairs
- 14 • To use the dummy CoAP response to correspond to and thus close the CoAP request,
15 indicating that no CoAP level error occurred, and implying that no further CoAP
16 response is to be expected. A further CoAP response is usually expected if an empty
17 ACK is sent in reply.
- 18 • To enable addition or removal of TMF commands in future versions of the Thread
19 specification, by providing a means to detect that a specific TMF command is
20 unavailable on the recipient. In that case, the recipient will return code 4.04 ('Not
21 Found'), which signals the sender of the request that the command is not available.

22

10.8.5 TMF Request, Piggybacked TMF Response

23 The TMF Request, piggybacked TMF Response (Req+Rsp_Piggybacked) transaction pattern
24 is a CoAP request using a CoAP CON message, which is replied to with a CoAP ACK message
25 on which the CoAP response is piggybacked. Req+Rsp_Piggybacked SHALL be used when
26 the responder can determine with certainty that it is able to respond within the CoAP
27 ACK_TIMEOUT timeout period.

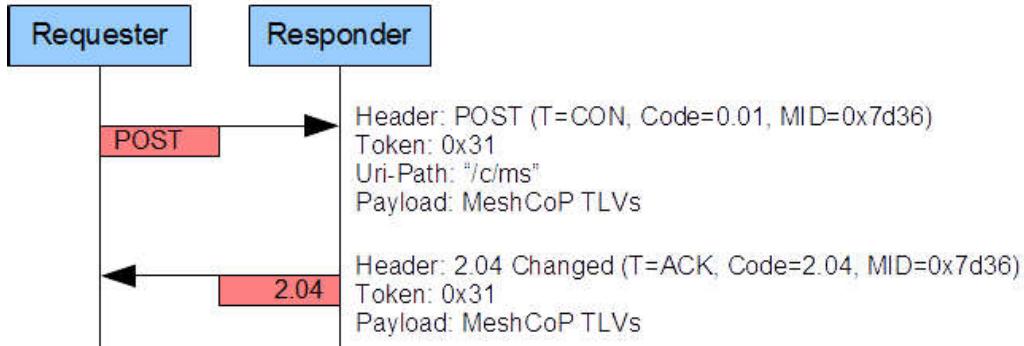


Figure 10-3. Req+Rsp_Piggybacked Example Transaction

10.8.6 TMF Request, Separate TMF Response

The TMF Request, separate TMF Response (Req+Rsp_Separate) transaction pattern is a CoAP request using a CoAP CON message, which is replied to with an empty CoAP ACK message. The CoAP response is sent separately using a further CoAP CON message, which in turn is replied to with an empty CoAP ACK message. Req+Rsp_Separate SHALL be used when the responder cannot determine beforehand that it is able to respond within the CoAP ACK_TIMEOUT period.

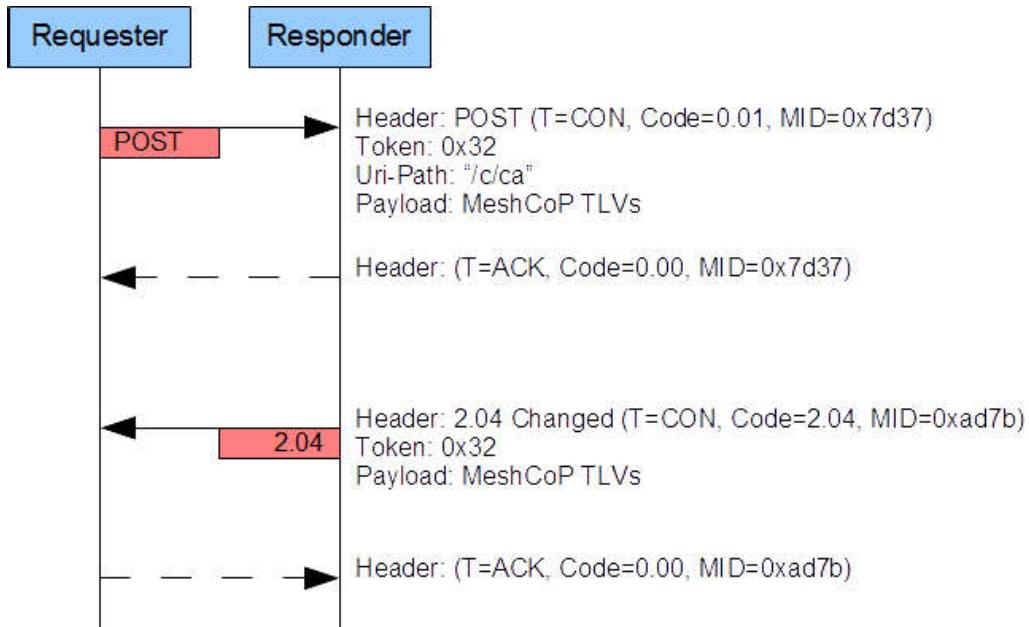


Figure 10-4. Req+Rsp_Separate Example Transaction

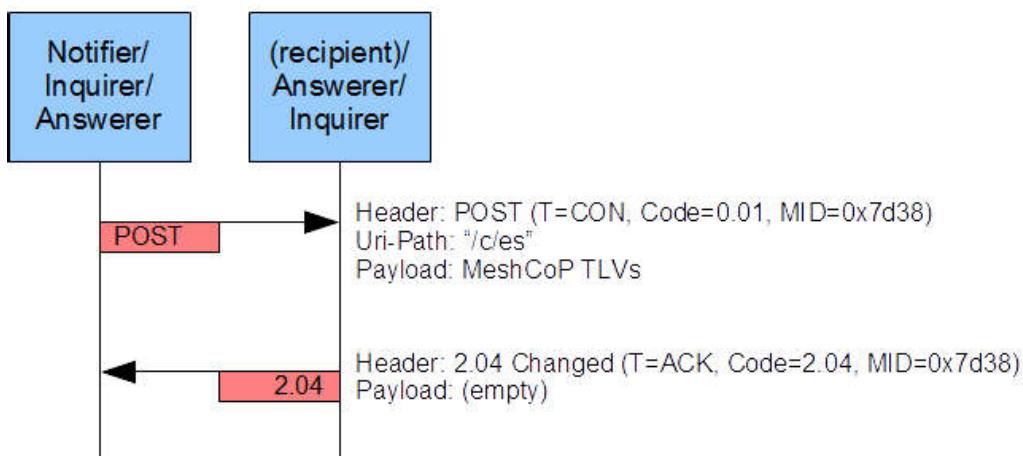
10.8.6.1 Example Scenario

An example of this is when the Commissioner sends COMM_KA.req. When the Border Agent as responder receives the COMM_KA.req, it sends a further LEAD_KA.req to the Leader and waits for LEAD_KA.rsp. The Leader as responder to the LEAD_KA.req is likely to be able to determine that it can respond within ACK_TIMEOUT period and therefore SHOULD use the Req+Rsp_Piggybacked transaction pattern. However, the Border Agent as responder to the

1 COMM_KA.req is unlikely to be able to determine that it can respond within the
2 ACK_TIMEOUT period and therefore SHOULD use the Req+Rsp_Separate transaction
3 pattern, otherwise it may cause the requester to retransmit the COMM_KA.req.

4 **10.8.7 Confirmable TMF Notification, Query or 5 Answer, Piggybacked Dummy Response**

6 The Confirmable TMF Notification, Query or Answer, piggybacked dummy response
7 (Ntf/Qry/Ans_CON) transaction pattern is a CoAP request using a CoAP CON message,
8 which is replied to with a CoAP ACK message on which a dummy CoAP response is
9 piggybacked (see Section 10.8.4, [CoAP Request/Response Pairing](#)). Ntf/Qry/Ans_CON is
10 used for unicast TMF Queries and TMF Answers and for unicast TMF Notifications where
11 reliability is required per transaction.



12
13 **Figure 10-5. Ntf/Qry/Ans_CON Example Transaction**

14 **10.8.8 Non-confirmable TMF Notification, Query or 15 Answer**

16 The Non-confirmable TMF Notification, Query or Answer (Ntf/Qry/Ans_NON) transaction
17 pattern is a CoAP request using a CoAP NON message. Ntf/Qry/Ans_NON is used for
18 multicast TMF Notifications and TMF Queries and for unicast TMF Notifications where
19 reliability is not required per transaction as it may be implied at a higher layer. For example,
20 DTLS handshake records have their own reliability mechanism, therefore a TMF Notification
21 carrying a DTLS handshake record does not need to be delivered reliably at the CoAP
22 messaging layer. Note that a CoAP response is not generated by the recipient: it is
23 suppressed. This deviates from the unicast RESTful case described in [[RFC 7252](#)] Section
24 2.2, where a request is defined to always solicit a response.

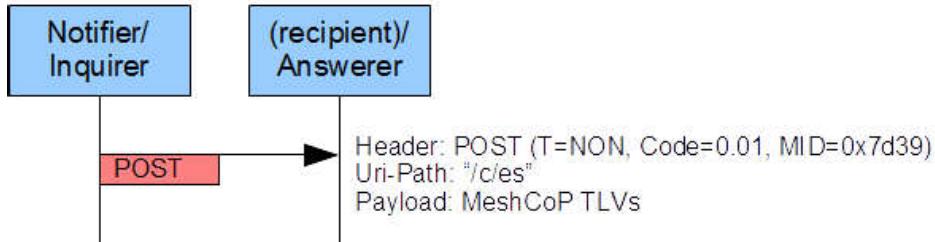


Figure 10-6. Ntf/Qry/Ans_NON Example Transaction

10.9 Commissioner Transactions

There are three categories of transaction involving a Commissioner, depending on the type of Commissioner:

- Native or External Commissioner with Thread device
 - Native or External Commissioner destined for Border Agent
 - On-mesh Commissioner destined for Thread device

10.9.1 Native or External Commissioner with Thread Device

A transaction involving a Native or External Commissioner with a Thread device uses the existing secure Commissioning Session to encapsulate a TMF message, which is then relayed by the Border Agent. This is illustrated in Figure 10-7.

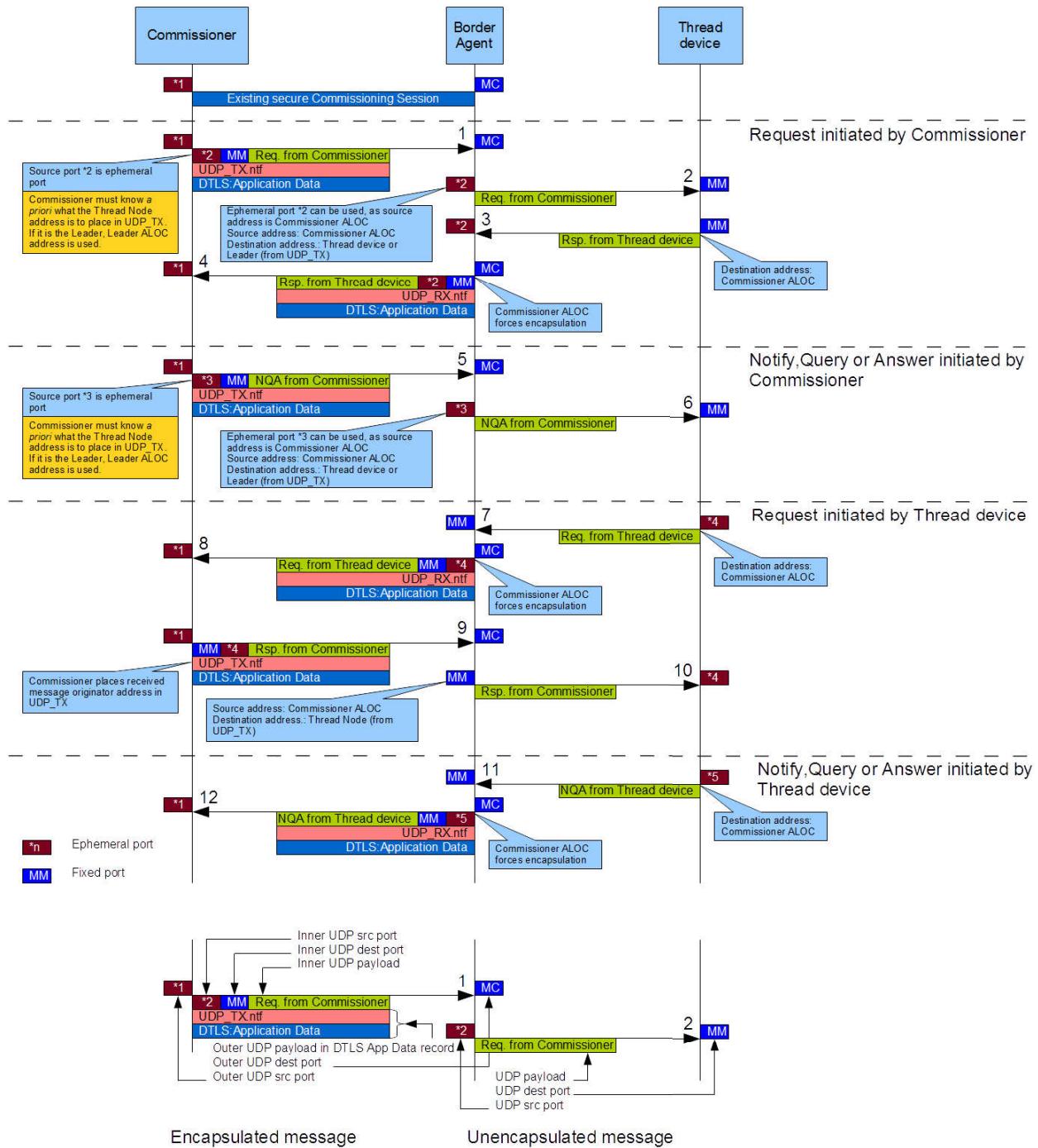


Figure 10-7. Native or External Commissioner Transactions with Thread Device

10.9.1.1 Destined for Thread Device

A TMF message originating from a Native or External Commissioner destined for a Thread device is encapsulated using a UDP_TX.ntf message.

10.9.1.1.1 Commissioner Behavior

2 The Commissioner SHALL form an IPv6 Address TLV using the address of the destination
3 Thread device. If the destination Thread device is the Leader, the IPv6 Address TLV SHALL
4 contain the Leader ALOC. The Commissioner forms a UDP Encapsulation TLV using:

- 5 • Source port set to a random 16-bit number (ephemeral source port)
- 6 • Destination port set to the Thread management port
- 7 • Payload: MeshCoP TLVs set to the TMF message

8 The Commissioner then forms a UDP_TX.ntf TMF message and sends the UDP_TX.ntf to the
9 Border Agent through the secure Commissioning Session.

10.9.1.1.2 Border Agent Behavior

11 The Border Agent SHALL decapsulate the UDP_TX.ntf message and form the UDP datagram
12 from the UDP Encapsulation TLV. The UDP datagram source port, destination port, and
13 payload SHALL be obtained from the UDP Encapsulation TLV. The destination address of the
14 IPv6 packet containing the UDP datagram SHALL be obtained from the IPv6 Address TLV.
15 The source address of the IPv6 packet SHALL be the Commissioner ALOC corresponding to
16 the Native or External Commissioner.

17 The Border Agent then sends the resulting IPv6 packet to the corresponding Thread device.

10.9.1.2 Originating from Thread Device

19 A TMF message originating from a Thread device destined for a Native or External
20 Commissioner is encapsulated using a UDP_RX.ntf message.

10.9.1.2.1 Thread Device Behavior

22 The Thread device SHALL form an IPv6 packet destined for the Commissioner ALOC
23 corresponding to the Native or External Commissioner. The source address of the IPv6
24 packet SHALL be the address of the Thread device or, if the Thread device is the Leader, the
25 Leader ALOC. The payload of the IPv6 packet SHALL be the TMF message destined for the
26 Commissioner.

27 The Thread device then sends the resulting IPv6 packet to the Border Agent.

10.9.1.2.2 Border Agent Behavior

29 On reception of an IPv6 packet containing a TMF message destined for the Commissioner
30 ALOC corresponding to the Native or External Commissioner, the Border Agent SHALL form
31 an IPv6 Address TLV using the address of the originating Thread device. If the originating
32 Thread device is the Leader, the IPv6 Address TLV SHALL contain the Leader ALOC. The
33 Border Agent forms a UDP Encapsulation TLV using:

- 34 • Source port set to the source port obtained from the incoming UDP datagram
35 containing the TMF message
- 36 • Destination port set to the destination port obtained from the incoming UDP
37 datagram containing the TMF message
- 38 • Payload set to the payload obtained from the incoming UDP datagram containing the
39 TMF message

40 The Border Agent then forms a UDP_RX.ntf TMF message and sends the UDP_RX.ntf to the
41 Commissioner through the secure Commissioning Session.

10.9.2 Native or External Commissioner with Border Agent

A transaction involving a Native or External Commissioner with the Border Agent uses the existing secure Commissioning Session to carry a TMF message with no encapsulation. This is illustrated in Figure 10-8.

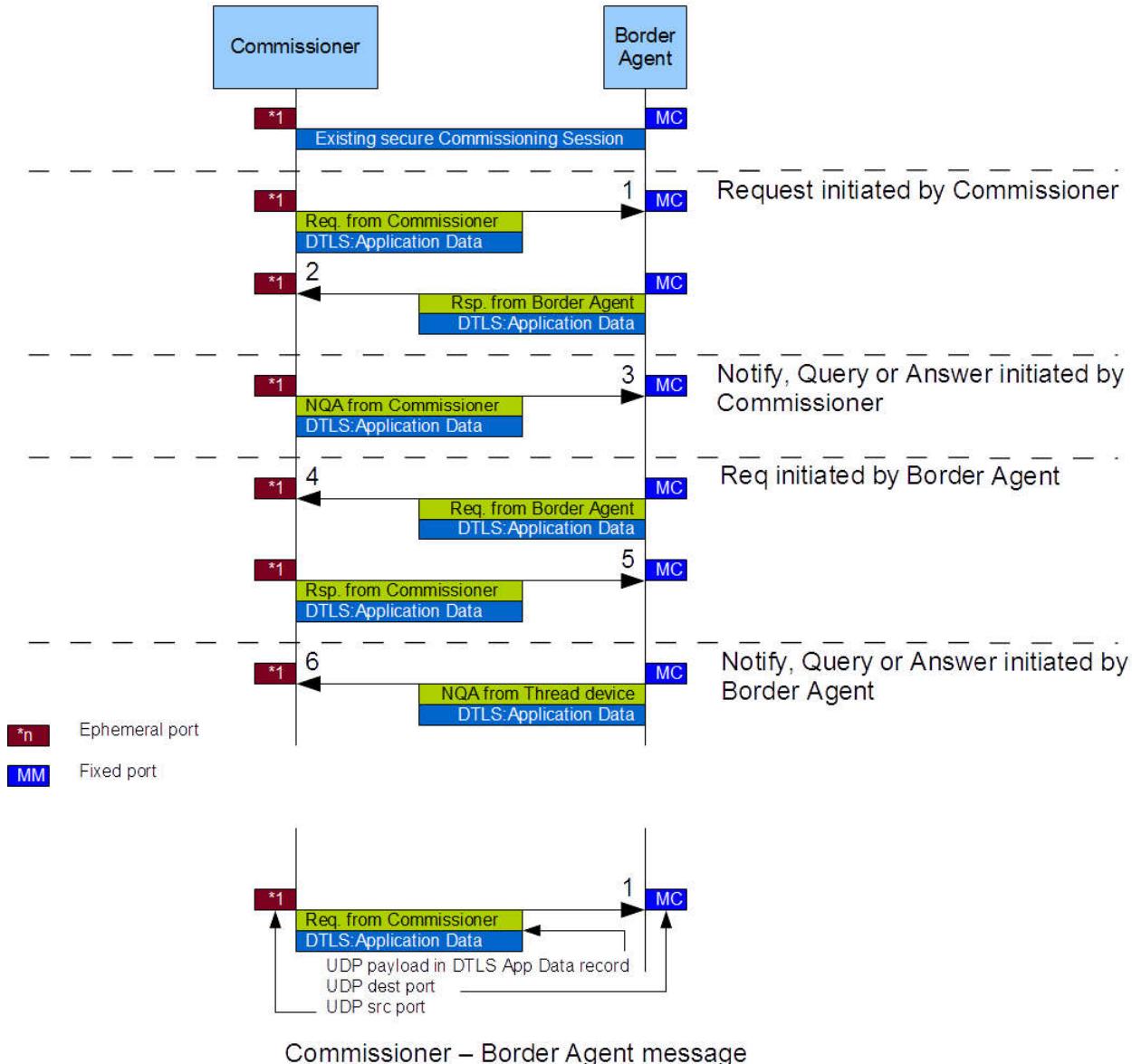
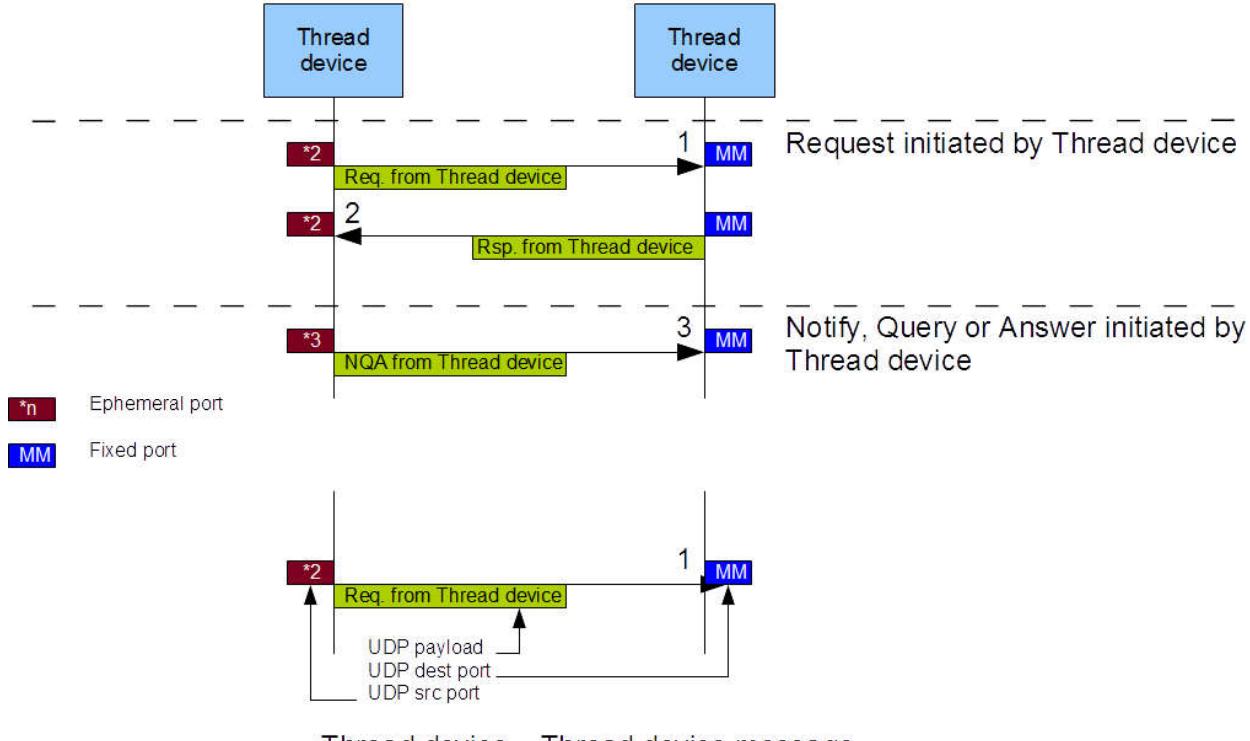


Figure 10-8. Native or External Commissioner Transactions with Border Agent

10.9.3 On-mesh Commissioner

A transaction involving an On-mesh Commissioner with any other Thread device uses the Thread network to carry a TMF message with no encapsulation. A TMF message originating from the On-mesh Commissioner destined for a Thread device and a TMF message

1 originating from a Thread device destined for the On-mesh Commissioner are sent directly
2 through the Thread network. This is illustrated in .



4 **Figure 10-9. Thread Device Transactions with Other Thread Device**

5 **10.10 Message Mapping**

6 If a responder knows it can respond within ACK_TIMEOUT to a TMF Request, it piggybacks
7 the TMF Response in the ACK (Req+Rsp_Piggybacked), otherwise it replies with an empty
8 ACK and responds later (Req+Rsp_Separate).

9 In the following tables:

- 10 UTP: Unicast Transaction Pattern
- 11 MTP: Multicast Transaction Pattern
- 12 Schm: URI scheme (coap/coaps)
- 13 RRP: Req+Rsp_Piggybacked
- 14 RRS: Req+Rsp_Separate
- 15 NQAC: Ntf/Qry/Ans_CON
- 16 NQAN: Ntf/Qry/Ans_NON

17 **10.10.1 Address Resolution API (/a/...)**

18 The TLVs used in this namespace are defined in Section 5.19, Network Layer TLVs in
19 Chapter 5, Network Layer.

1

Table 10-5. Address Resolution API

Message	URI	UTP	MTP	Port	Schm
ADDR_QRY.qry	/a/aq	-	NQAN	:MM	coap
ADDR_NTF.ans	/a/an	NQAC	-	:MM	coap
ADDR_NTF.ntf	/a/an	NQAC	-	:MM	coap
ADDR_ERR.ntf	/a/ae	NQAC	NQAN	:MM	coap
ADDR_SOL.req/rsp	/a/as	RRP RRS	-	:MM	coap
ADDR_REL.ntf	/a/ar	NQAC	-	:MM	coap
SVR_DATA.ntf	/a/sd	NQAC	-	:MM	coap
ND_DATA.req/rsp	/a/nd	RRP RRS	-	:MM	coap

2

10.10.2 MeshCop API (/c/...)

3
4

The TLVs used in this namespace are defined in Section 8.10, MeshCoP TLV Formats, in Chapter 8, Mesh Commissioning Protocol.

5

Table 10-6. MeshCoP API

Message	URI	UTP	MTP	Port	Schm
COMM_PET.req/rsp	/c/cp	RRS	-	:MC	coaps
COMM_KA.req/rsp	/c/ca	RRS	-	:MC	coaps
UDP_RX.ntf	/c/ur	NQAN	-	:MC	coaps
UDP_TX.ntf	/c/ut	NQAN	-	:MC	coaps
RLY_RX.ntf	/c/rx	NQAN	-	:MM :MC	coap coaps
RLY_TX.ntf	/c/tx	NQAN	-	:MM :MC	coap coaps
MGMT_GET.req/rsp	/c/mg	RRP RRS	-	:MM :MC	coap coaps

Message	URI	UTP	MTP	Port	Schm
MGMT_SET.req/rsp	/c/ms	RRP RRS	-	:MM :MC	coap coaps
LEAD_PET.req/rsp	/c/lp	RRP RRS	-	:MM	coap
LEAD_KA.req/rsp	/c/la	RRP RRS	-	:MM	coap
JOIN_ENT.ntf	/c/je	NQAC	-	:MM	coap
JOIN_FIN.req/rsp	/c/jf	RRP	-	:MJ	coaps
MGMT_COMMISISONER_GET.req/rsp	/c/cg	RRP RRS	-	:MM :MC	coap coaps
MGMT_COMMISISONER_SET.req/rsp	/c/cs	RRP RRS	-	:MM :MC	coap coaps
MGMT_ACTIVE_GET.req/rsp	/c/ag	RRP RRS	-	:MM :MC	coap coaps
MGMT_ACTIVE_SET.req/rsp	/c/as	RRP RRS	-	:MM :MC	coap coaps
MGMT_PENDING_GET.req/rsp	/c/pg	RRP RRS	-	:MM :MC	coap coaps
MGMT_PENDING_SET.req/rsp	/c/ps	RRP RRS	-	:MM :MC	coap coaps
MGMT_DATASET_CHANGED.ntf	/c/dc	NQAC	-	:MM	coap
MGMT_ANNOUNCE_BEGIN.ntf	/c/ab	NQAC	NQAN	:MM MC	coap coaps
MGMT_PANID_QUERY.qry	/c/pq	NQAC	NQAN	:MM :MC	coap coaps
MGMT_PANID_CONFLICT.ans	/c/pc	NQAC	-	:MM	Coap
MGMT_ED_SCAN.qry	/c/es	NQAC	NQAN	:MM :MC	coap coaps
MGMT_ED_REPORT.ans	/c/er	NQAC	-	:MM	coap

10.10.3 Diagnostic API (/d/...)

2 The TLVs used in this namespace are defined in Section 10.11.4, [Diagnostic TLVs](#).

3 **Table 10-7. Diagnostic API**

Message	URI	UTP	MTP	Port	Schm
DIAG_GET.req/rsp	/d/dg	RRP RRS	-	:MM :MC	coap coaps
DIAG_GET.qry	/d/dg	NQAC	NQAN	:MM :MC	coap coaps
DIAG_GET.ans	/d/dg	NQAC	-	:MM :MC	coap coaps
DIAG_RST.ntf	/d/dr	NQAC	-	:MM :MC	coap coaps

10.11 Network Diagnostics

10.11.1 Overview

6 This section defines the methods for observing network health statistics and topology data,
7 such as routing tables, link costs and packet error rates on a Thread Network. This data can
8 be retrieved from a Thread device by requesting diagnostic TLVs. New TLV formats are
9 defined for this purpose, but also existing TLV formats from this specification are re-used
10 whenever possible.

10.11.1.1 Device Diagnostics

12 The following device data can be retrieved from a device:

- MAC Extended Address (64-bit)
- MAC Address (16-bit), also referred to as IEEE 802.15.4 Short Address or RLOC16
- Mode (Capability information)
- Timeout (Sleepy polling period)
- Link quality information
- Routing Table
- Leader Data
- Network Data

10.11.1.2 Network Diagnostics

22 It is a good practice to make the IPv6 address list available to an upper layer network
23 application to determine the overall connection status of a Thread device.

10.11.1.3 Link Diagnostics

2 Each device maintains packet counters for number of transmitted and received 802.15.4
3 packets, packet errors, discarded packets and unrecognized packets. These counters can be
4 reset using the DIAG_RST.ntf command.

10.11.2 Diagnostic Commands

6 The diagnostic commands provide a mechanism to obtain connectivity, topology and link
7 quality data from all the Thread devices in a Thread Network. The cornerstone of the initial
8 Thread diagnostics protocol is to simply provide a multi-hop query mechanism for TLVs over
9 CoAP. Thread Diagnostic commands MUST be supported by FTDs. Thread Diagnostic
10 commands are optional for MTDs and MAY be supported.

10.11.2.1 DIAG_GET.req – Get Diagnostic Request

12 Request a set of diagnostic data from a Thread device.

13 CoAP Request URI

14 `coap://[<destination>]:MM/d/dg`

15 Transaction Pattern

16 Req+Rsp_Piggybacked or Req+Rsp_Separate

17 CoAP Payload

18 Type List TLV

19 Type List TLV

20 Contains a list of 8-bit Diagnostic TLV type identifiers to request from the recipient.

10.11.2.2 DIAG_GET.rsp – Get Diagnostic Response

22 Response to DIAG_GET.req with a set of diagnostic data from a Thread device. This
23 response is not sent by an MTD that does not support Thread Diagnostic commands.

24 CoAP Response Code

25 2.04 Changed

26 Transaction pattern

27 Req+Rsp_Piggybacked or Req+Rsp_Separate

28 CoAP Payload

29 Diagnostic TLVs

30 Diagnostic TLVs

31 A series of zero or more Diagnostic TLVs containing parameters and values that were
32 requested by the DIAG_GET.req message. If a Thread device is unable to supply a
33 specific Diagnostic TLV, that TLV is omitted. The list will be empty if a Thread device is
34 unable to supply any TLVs.

10.11.2.3 DIAG_GET.qry – Get Diagnostic Query

2 Query a set of diagnostic data from more than one Thread device. This is typically used with
3 a multicast address; DIAG_GET.req SHOULD be used for a unicast request of diagnostic
4 data from a specific Thread device.

5 CoAP Request URI
6 `coap://[<destination>]:MM/d/dq`

7 Transaction Pattern
8 Ntf/Qry/Ans_CON (unicast) or Ntf/Qry/Ans_NON (multicast)

9 CoAP Payload
10 Type List TLV
11 Type List TLV
12 Contains a list of 8-bit Diagnostic TLV type identifiers to request from the recipient.

10.11.2.4 DIAG_GET.ans – Get Diagnostic Answer

14 Answer to DIAG_GET.qry with a set of diagnostic data from a Thread device as follows. This
15 answer is not sent by an MTD that does not support Thread Diagnostic commands.

16 CoAP Request URI
17 `coap://[<destination>]:MM/d/da`

18 Transaction Pattern
19 Ntf/Qry/Ans_CON

20 CoAP Payload
21 Diagnostic TLVs
22 Diagnostic TLVs
23 A series of zero or more Diagnostic TLVs containing parameters and values that were
24 queried by the DIAG_GET.qry message. If a Thread device is unable to supply a specific
25 Diagnostic TLV, that TLV is omitted. The list will be empty if a Thread device is unable to
26 supply any TLVs.

10.11.2.5 DIAG_RST.ntf – Reset Diagnostic Notification

28 Notification to reset a set of diagnostic data on a Thread device. Only diagnostic items
29 marked with 'Can Reset' equal to 'Y' in Table 10-8 can be reset using this command.

30 CoAP Request URI
31 `coap://[<destination>]:MM/d/dr`

32 Transaction Pattern
33 Ntf/Qry/Ans_CON

34 CoAP Payload
35 Type List TLV
36 Type List TLV
37 A list of 8-bit Diagnostic TLV type identifiers to reset on the recipient.

10.11.3 Diagnostic TLVs Overview

Table 10-8 lists the Diagnostic TLVs along with the information these provide. The last column marks whether the diagnostic counter value(s) in the TLV are reset to 0 by the DIAG_RST.ntf command (if 'Y') or not (if 'N'). The diagnostic TLVs that can reset ('Y') are reset back to 0 also when the Thread device reboots.

Note: All diagnostic information in these items pertain to the specific Thread interface over which the information is requested. If a device has multiple Thread interfaces, the diagnostic information is maintained separately per interface.

Table 10-8. Diagnostic TLVs

TLV Type	Name	TLV Length, format of Value and description	Can Reset?
0	MAC Extended Address (64-bit)	Same format and semantics as MAC Extended Address TLV (Network Layer TLV Type 1, Section 5.18.2)	N
1	MAC Address (16-bit)	Same format and semantics as Address16 TLV (MLE TLV Type 10, Section 4.5.11)	N
2	Mode (Capability Information)	Same format and semantic as Mode TLV (MLE TLV Type 1, Section 4.5.2)	N
3	Timeout	For SEDs, indicates the maximum polling time period. A non-SED Thread device MUST omit this TLV in a DIAG_GET.rsp. Same format and semantics as Timeout TLV (MLE TLV Type 2, Section 4.5.3)	N
4	Connectivity	Same format and semantics as the Connectivity TLV (MLE TLV Type 15, Section 4.5.16)	N
5	Route64	Same format and semantics as the Route64 TLV (MLE Type 9, Section 4.5.10)	N
6	Leader Data	Same format and semantics as the Leader Data TLV (MLE TLV Type 11, Section 4.5.12)	N
7	Network Data	Same format and semantics as the Network Data TLV (MLE TLV Type 12, Section 4.5.13)	N
8	IPv6 address list	List of all link-local and higher scoped IPv6 (unicast) addresses registered by the Thread device on this Thread interface. With N addresses in the list the length is N*16 bytes. All 16-byte addresses are concatenated.	N
9	MAC Counters	TLV that contains packet/event counters for the MAC 802.15.4 interface. Defined in Section 10.11.4.1, MAC Counters TLV (9) .	Y

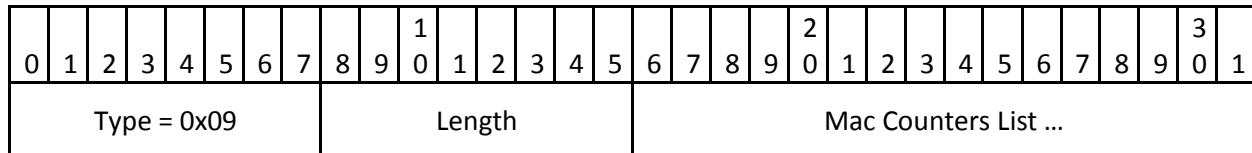
TLV Type	Name	TLV Length, format of Value and description	Can Reset?
14	Battery Level	Indication of remaining battery energy. Defined in Section 10.11.4.2, <u>Battery Level TLV (14)</u> .	N
15	Supply Voltage	Indication of the current supply voltage. Defined in Section 10.11.4.3, <u>Supply Voltage TLV (15)</u> .	N
16	Child Table	List of all children of a Router. Defined in Section 10.11.4.4, <u>Child Table TLV (16)</u> .	N
17	Channel Pages	List of supported frequency bands. Defined in Section 10.11.4.5, <u>Channel Pages TLV (17)</u> .	N
18	Type List	List of type identifiers used to request or reset multiple diagnostic values.	N
19	Max Child Timeout	Same format and semantics as Timeout TLV (MLE TLV Type 2, Section 4.5.3)	N

10.11.4 Diagnostic TLVs

2 This section provides details for the format of selected Diagnostic TLVs.

3 10.11.4.1 MAC Counters TLV (9)

4 The MAC Counters TLV is defined as follows:



5 **Figure 10-10. MAC Counters TLV (9)**

6 Mac Counters List

7 A list as defined below of values according to the semantics and the syntax of selected
8 MIB objects of the 'ifPacketGroup' MIB object group from [\[RFC 2863\]](#).

9 The specific values returned are a useful subset of the 'ifPacketGroup' MIB object. This
10 includes counters for transmitted/received 802.15.4 packets, as well as dropped packet
11 counters and separate unicast/broadcast statistics.

12 The following elements from [\[RFC 2863\]](#) are included in this order:

13 ifInUnknownProtos, ifInErrors, ifOutErrors, ifInUcastPkts, ifInBroadcastPkts, ifInDiscards,
14 ifOutUcastPkts, ifOutBroadcastPkts, ifOutDiscards

15 Per element the format as defined by [\[RFC 2863\]](#) is used including its referenced
16 [\[RFC 1902\]](#) for the Counter32 and Integer32 definitions. Although the MIB descriptions in
17 these RFCs are described using the ASN.1 notation, no ASN.1-specific encoding like DER,
18 XER or PER is applied to the list defined above.

10.11.4.2 Battery Level TLV (14)

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3
Type = 0x0E								Length = 1								Battery Level									

Figure 10-11. Battery Level TLV (14)

Battery Level

An 8-bit unsigned integer that indicates remaining battery energy in the Thread device as an integer percentage value 0-100 (0x00-0x64).

If the battery level is not measured, is unknown or the device does not operate on battery power this TLV is omitted in a DIAG_GET.rsp or DIAG_GET.ans.

10.11.4.3 Supply Voltage TLV (15)

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x0F								Length = 2								Supply Voltage																		

Figure 10-12. Supply Voltage TLV (15)

Supply Voltage

A 16-bit unsigned integer, in millivolt [mV] units, indicating the current supply voltage to the Thread radio interface subsystem of the Thread device.

If the supply voltage is not measured or is unknown this TLV is omitted in a DIAG_GET.rsp or DIAG_GET.ans.

10.11.4.4 Child Table TLV (16)

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x10								Length								List of Child Entries																		

Figure 10-13. Child Table TLV (16)

List of Child Entries

A list of zero or more Child Entry data structures as defined below.

Child Entry

Structure data about a Child as follows:

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3
Timeout						Rsv	Child ID						Mode												

Figure 10-14. Child Entry Data Format

- 1 Timeout
2 Expected poll time expressed as $2^{(\text{Timeout}-4)}$ seconds. This provides an exponential
3 range between 62ms and 4.25 years. The parent shall fill this field with the closest
4 ceiling value based on the MLE Timeout TLV.
- 5 Rsv
6 Reserved for future use.
- 7 Child ID
8 The 9-bit Child ID from which an RLOC can be generated.
- 9 Mode
10 The mode bits for the child as sent in the MLE Mode TLV.

11 **10.11.4.5 Channel Pages TLV (17)**

12 When selecting a new Channel for a Thread Network, it may be desirable to know what
13 frequency bands each Thread device supports.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x11	Length										ChannelPages ...																							

14 **Figure 10-15. Channel Pages TLV (17)**

15 ChannelPages

16 A list of one or more 8-bit integers that indicates the set of supported IEEE 802.15.4
17 ChannelPage values.

18 **10.11.4.5.1 Supported Channel Pages**

19 Table 10-9 lists the supported channel pages.

20 **Table 10-9. Supported Channel Pages**

Channel Page Value	Description
0	2.4 GHz O-QPSK PHY as defined in Section 6.5 of [IEEE802154]
1 – 255	Reserved

21 **10.11.4.6 Type List TLV (18)**

22 Concatenated list of type identifiers of other diagnostics TLVs used to request or reset
23 multiple diagnostic values.

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x12	Length										TypeIdentifiers...																							

24 **Figure 10-16. Type List TLV (18)**

1 TypeIdentifiers

2 A list of one or more 8-bit integers that contain the type identifiers for Diagnostic values
3 being requested or reset (see Table 10-8).

4 **10.11.4.7 Max Child Timeout TLV (19)**

0	1	2	3	4	5	6	7	8	9	1	0	1	2	3	4	5	6	7	8	9	2	0	1	2	3	4	5	6	7	8	9	3	0	1
Type = 0x13										Length										Max Timeout														

5 **Figure 10-17. Max Child Timeout TLV (18)**

6 Max Timeout

7 This field reports the longest MLE Timeout a router has registered for all of its active
8 children.

9

10 **10.12 Persistent Data**

11 Devices operating in the field may be reset accidentally or on purpose for a variety of
12 reasons. Devices that have been reset need to restart network operations without user
13 intervention. For this to be done successfully, non-volatile storage must store the following
14 information:

- 15 • The content of the Active Operational Dataset and the Pending Operational Dataset
16 • Security material (each key used along with corresponding outgoing frame counters)
17 • Addressing information from the network to form the IPv6 addresses for the devices.
18 (Note for Sleepy End Devices (SEDs): Parent information is to be saved and the
19 device attempts to reconnect with its Parent. If the SED cannot connect to the
20 existing Parent, it reconnects to another Parent, and then stores that data.)

21 **Note: Router or REED devices reconnect upon a reset using the synchronization
22 upon reset process and MLE messages described in Chapter 4, Mesh Link
23 Establishment.**

24 Parent Routers MUST store the following information for each MTD child:

- 25 • Extended and short MAC addresses
26 • Child mode (to determine whether it is sleepy)
27 • Child timeout value

10.13 IANA Considerations

Table 10-10 lists the port used in the Thread Management Framework. Because the port number is in the dynamic/private UDP port range, IANA registration is not needed.

Table 10-10. Thread Management Framework Port

Service name	Port Abbreviation	Default port number	Transport	Description
threadmgmt	:MM	61631	udp	Thread Network Management

Note: The port number :MM (0xf0bf) is the last number within the 6LoWPAN-compressible range of UDP ports (Section 4.3.1 of [\[RFC 6282\]](#)).

Note: This specification defines no mechanisms to change the port number :MM to another, non-default, value.

CHAPTER 11 FUNCTIONAL DESCRIPTION

Contents

11.1 Forming a Network	11-2
11.1.1 Types of Devices and Network Structure	11-2
11.1.2 Thread Network Data.....	11-3
11.2 Joining a Thread Network	11-3
11.3 Links and Link Establishment.....	11-4
11.4 Message Forwarding and Routing	11-5
11.5 Router Selection	11-5
11.6 FTD Parents and MTD Children	11-6
11.7 Partitioning	11-6
11.8 Commissioning.....	11-7

11.1 Forming a Network

As with any 802.15.4 network, forming a Thread Network begins with picking a channel and PAN Identifier (PAN ID). Normally the device performs an energy scan to pick a quiet channel and then an active scan to verify which PAN IDs may be already used by other networks in range.

The device MUST choose the following values upon starting a Thread Network.

- Thread Network Short PAN Identifier (PAN ID): 2 bytes in length set as the *macPANId* IEEE 802.15.4 PIB attribute for all nodes; the PAN ID MUST be different from 0xFFFF and SHOULD be a randomly generated value which does not conflict with a PAN ID already used by a device in range as identified by an active scan.
- Master Key: 16 bytes in length, chosen using a cryptographically-sound random number generator.
- Commissioning Credential: This is human-readable key, 8-255 bytes in length, used to form the PSKc used when authenticating a Commissioner.
- Mesh-Local Prefix: an IPv6 Unique Local Address (ULA) prefix used for communication within the Thread Network. The value of the prefix MUST be generated according to IPv6 prefix format described in section 3 of [\[RFC 4193\]](#).
- Extended PAN ID: a randomly chosen 8-byte value, used to uniquely identify Thread Networks in range.
- Network Name: a human-readable name for the network, up to 16 bytes in length. The Network Name also appears in Discovery Response or IEEE 802.15.4 beacon messages sent by members of the Thread Network.

The forming node becomes the first Router in the Thread Network partition and selects a Router ID for itself.

If the forming node is a Border Router, it adds its information about its external connection to the Thread Network Data. At this point the Thread Network has been formed and the forming node becomes an active Router, sending periodic MLE (Mesh Link Establishment) advertisements.

11.1.1 Types of Devices and Network Structure

A Thread Network may contain two different types of Thread devices. Each device acts in a particular role. There are two types of Thread devices:

- Full Thread Device (FTD)
- Minimal Thread Device (MTD)

FTDs can communicate with each other and with their attached MTD Children. MTDs can only communicate with the FTD Parent they are attached to.

There are five roles a Thread device can act as:

- Active Router
- Router-Eligible End Device (REED)
- Full End Device (FED)
- Minimal End Device (MED)
- Sleepy End Device (SED)

An FTD may act as an Active Router, a REED, or a FED. An MTD may act as a MED or a SED.

- 1 In this specification, an FTD acting as an Active Router is referred to simply as a "Router"
2 and an FTD acting as a REED is referred to as a "REED", reflecting their different roles.
- 3 Other than the device that forms the Thread Network, which is automatically a Router,
4 devices initially attach to the Thread Network as an End Device. Those that attach as REEDs
5 may request to become Routers at any time after attaching.
- 6 Routers may also downgrade to REEDs, depending on the state of the Thread Network.
7 There can be a maximum of 32 Routers in a Thread Network partition. Each Router is
8 assigned one of 63 Router IDs, 0...62 (Router ID 63 is reserved).
- 9 There is a designated Router, the Leader, which accumulates and distributes information
10 about the Thread Network Partition. The Leader is chosen autonomously by the Thread
11 Network; any Router is able to function as the Leader if the need arises. The Leader's main
12 functions are to:
- 13 • Assign and manage Router IDs.
14 • Collate and distribute the Thread Network Data.

15 **11.1.2 Thread Network Data**

- 16 The Thread Network Data is a collection of information about the Thread Network that is
17 collected and distributed by the Leader. This includes: Border Router and valid prefixes; and
18 commissioning information, including Thread Network parameters.
- 19 The Thread Network Data is distributed in a Trickle-like fashion: Routers send periodic MLE
20 advertisements that include the sequence number of the latest Thread Network Data they
21 have heard. Nodes hearing of a newer sequence number from a neighbor, request the
22 Thread Network Data from that neighbor.
- 23 MTDs operate differently with respect to the Thread Network Data. MTDs can request that
24 they only receive a relatively stable subset of the Thread Network Data from their FTD
25 Parent. This is to avoid overburdening battery-powered devices with keeping track of
26 ephemeral information.

27 **11.2 Joining a Thread Network**

28 To join a Thread Network a device must first acquire the following commissioning
29 information:

- 30 • Master Key
31 • Commissioning Key (PSKc)
32 • Mesh-Local Prefix
33 • Extended PAN ID
34 • Network Name

35 These may be obtained either through in-band commissioning (as described in Chapter 8,
36 Mesh Commissioning Protocol) or through out-of-band commissioning. Out-of-band
37 commissioning is not defined in this specification, other than a requirement that it be at
38 least as secure as in-band commissioning.

1 The joining device also needs the Thread Network's current channel and PAN ID, which it
2 may receive either out-of-band or by scanning for a Thread Network to join. Once the
3 Thread Network has been located, the joining device attaches as an End Device to a Parent
4 Router already in the Thread Network. When it attaches, the joining device receives an
5 RLOC16 assigned by its Parent; this ID embeds the Parent's Router ID and the Child ID
6 assigned by the Parent to the joining device. The joining device also receives the current
7 Thread Network Data. The device is now on the Thread Network and has the following
8 unicast addresses:

- RLOC - address formed from the Mesh-Local Prefix and an IID derived from its Router ID and Child ID. The RLOC16 is the last 16 bits of the RLOC address.
- ML-EID - Mesh-Local Endpoint Identifier address formed from the Mesh-Local Prefix and an IID chosen by the device.

13 The RLOC address may change when the network topology changes. In particular, if an end
14 device attaches to a new Parent, it will get a new RLOC address.

15 Once on the Thread Network, the device may configure global addresses via any available
16 Border Routers, and, if a REED, may upgrade to becoming an active Router. If the device is
17 itself a Border Router, it informs the Leader and is added to the list of Border Routers in the
18 Thread Network Data.

19 **11.3 Links and Link Establishment**

20 Thread uses 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) for packet
21 compression. The Leader assigns the Context IDs and distributes them in the Thread
22 Network Data. The Mesh-Local Prefix is always assigned context ID zero.

23 With the exception of in-band commissioning, all messages are secured, either by
24 [IEEE802154](#) security (level 5, encryption and 32-bit MIC), or by the MLE protocol (which is
25 based on 802.15.4 AES-CCM frame security, but done above the link layer). The 802.15.4
26 and MLE keys are derived from the master key received during commissioning.

27 MLE is used to configure links. The two main aspects are initializing frame counters and link
28 parameters (rx-on-when-idle, timeouts, etc.). Frame counters are used to detect replayed
29 messages. Thread devices use MLE handshakes to establish a link with a neighbor. MLE
30 introduces a handshake that verifies that a newly-heard neighbor does in fact possess the
31 link-layer key and is not simply replaying old messages.

32 One of the goals of the Thread Network design is to send as few non-application messages
33 as possible while including the greatest amount of information in each message. Because of
34 this, the MLE messages—both the link configuration messages and periodic advertisements—
35 contain a variety of information relating to different Thread Network functions.

36 There are two different MLE handshakes for the two types of links: router-to-router and
37 router-to-end device. The first carry additional information about link quality and
38 connectivity, the second about the end device's properties.

39 When attaching to a Parent, an end device first attempts to find a suitable Router within one
40 hop of itself. If that fails, it then chooses a REED within one hop and requests that the REED
41 become device router.

- 1 Routers also periodically send out MLE advertisements that contain the following:
- 2 • A heartbeat value used to determine connectivity to the Leader.
- 3 • The sequence numbers of the Thread Network Data held by the sender.
- 4 • The most recent set of assigned router IDs heard by the sender.
- 5 • The qualities of the links between the sender and its neighbors.
- 6 • The sender's route cost to all Routers in the Thread Network Partition.
- 7 Neighboring Routers that receive the advertisement use it to update their own local data.

11.4 Message Forwarding and Routing

9 MTDs forward all packets directly to their Parent and only receive unicast packets directly
10 from their FTD Parent. REEDs and FEDs can receive multicast messages from neighboring
11 10 Routers other than their respective Parents.

12 Forwarding on FTDs is done in two parts: selecting the relevant destination within the
13 Thread Network and then selecting the correct next hop to reach the mesh destination. If
14 the IP destination is a link local address, then the destination is a neighboring device and is
15 sent directly to the neighbor. If the IP destination is an RLOC, then the mesh destination
16 can be extracted from the IP destination. If the IP destination has an on-mesh prefix, then
17 Address Query is used to determine the RLOC of the IP destination (the results of Address
18 Query messages are also cached for later use).

19 For all other unicast destinations, the Router uses the Border Router information from the
20 Thread Network Data to choose the most appropriate Border Router, which is then the mesh
21 destination. In choosing a Border Router to forward to, the Router takes in to consideration
22 both the packet's IP destination and IP source addresses.

23 Multicasts are forwarded using proactive MPL (Multicast Protocol for Low Power and Lossy
24 Networks) forwarding [\[RFC 7731\]](#). Each Router retransmits each multicast packet a fixed
25 number of times.

26 Thread Routers use a distance-vector algorithm to determine next hop information for mesh
27 destinations. Each MLE advertisement includes its sender's link quality to each of its
28 neighbors and its path cost to all Routers in the network. The neighbors of the Routers use
29 this information to update their own next hops and path costs. The next hop of a packet
30 toward an end device is either the end device itself (in case its Parent determines the next
31 hop for the packet), or else the next hop is the next-hop Router on the optimal path toward
32 that Parent, as determined by the distance-vector algorithm. The format of mesh addresses
33 allows the Parent's mesh address to be determined from that of its end devices.

11.5 Router Selection

35 To include all routing and link information in a single 802.15.4 packet, Thread Networks are
36 limited to a relatively small number of Routers. The set of Routers must be chosen from the
37 available REEDs so the Thread Network stays connected and routes are not stretched out to
38 more hops than necessary.

39 Router selection is done in a distributed fashion: each Router and REED chooses whether to
40 change state based on its information about the local network topology.

41 The Leader arbitrates the assignment of Router IDs to avoid duplicate assignments. To
42 make this simple, there are more Router IDs than there are available Router slots. This

1 allows the Leader to assign an ID to a new Router without having to wait for information
2 about its predecessor to be removed from the Thread Network Partition's Routers. Router
3 IDs are assigned using Thread Management commands. When a REED wishes to become a
4 Router, it sends an Address Solicit request to the Leader; when a Router wishes to become
5 a REED, it sends an Address Release request to the Leader. The Leader will honor an
6 Address Solicit request as long as there are fewer than 32 routers in the Thread Network
7 Partition.

8 A REED requests to become a Router if either:

- 9 • The number of Routers is below ROUTER_UPGRADE_THRESHOLD.
- 10 • A node that is attempting to attach to the Thread Network attempts to use the REED
11 as its Parent.

12 A Router requests to become a REED if it determines that all of the following conditions are
13 met:

- 14 • The number of Routers in the Thread Network Partition is above
15 ROUTER_DOWNGRADE_THRESHOLD.
- 16 • It has at least one neighbor Router with at least comparable connectivity. (See
17 Section 5.9.9, Router ID Management, in Chapter 5, Network Layer, for details.)
- 18 • It has a relatively small number of MTD Children (the exact number is determined by
19 the number of Routers in the Thread Network partition). (See Section 5.9.9, Router
20 ID Management, in Chapter 5, Network Layer, for details.)

21 **11.6 FTD Parents and MTD Children**

22 Each FTD Parent keeps a table of the MTD Children associated with it. To reclaim unused
23 Child IDs and to avoid routing messages to an end device's old location, each MTD Child has
24 an associated timeout. If the MTD Child does not check in with its FTD Parent within the
25 timeout, the FTD Parent reclaims the MTD Child's ID and no longer acts on its behalf. The
26 timeout is specified by the MTD Child when it attaches. SEDs check in by sending an
27 802.15.4 data request. MEDs check in using the Child Update Request.

28 To reduce the code needed on MTDs, and because communication with SEDs can introduce
29 a great deal of latency, an FTD Parent acts as a proxy for the MTD Child in the following
30 ways:

- 31 • Respond to Address Query messages.
- 32 • Proactively send new Thread Network Data to the MTD Child when this data becomes
33 available.
- 34 • Inform them of changes to the MAC key.

35 This requires that FTD Parents store additional information about their MTD Children:

- 36 • Addresses configured using valid prefixes.
- 37 • The latest Thread Network Data the Child has received.
- 38 • The sequence number of the MAC key in use by the Child.

39 **11.7 Partitioning**

40 While the Thread Network intends that all nodes stay connected with one another, it is
41 possible that a Thread Network may consist of two or more disconnected fragments or

- 1 partitions. Thread is designed so that each partition can operate as a separate network and
2 separate partitions that become connected can merge together into a single connected
3 partition.
- 4 Each Thread Network Partition forms a separate mesh network with its own Leader, Thread
5 Network Data, and Router ID assignments. Messages are not forwarded between Thread
6 Network Partitions. The goal is for every collection of connected nodes in a Thread Network
7 to rapidly assemble into a single Thread Network Partition. A node that loses its connection
8 to the Leader of its Thread Network Partition joins the highest priority Thread Network
9 Partition available among its neighbors; if there is no Thread Network Partition available it
10 starts its own, with itself as the Leader.

11.8 Commissioning

12 The commissioning process begins when an off-network or on-mesh commissioning device
13 becomes an active Commissioner for a Thread Network Partition. There can be only one
14 active Commissioner at a time for a Thread Network Partition, so the commissioning device
15 petitions the Leader to become the active Commissioner.

16 An off-network commissioning device, typically a mobile phone, will generally initiate
17 commissioning by discovering the Thread Network through one of its Border Agents. The
18 device then sets up a DTLS connection with the Border Agent using the network's
19 Commissioning key (PSKc).

20 Once active, the Commissioner can enable joining on the network and optionally provide
21 Steering Data that indicates the EUI-64s of the devices expected to join.

22 A joining device (Joiner) can then communicate with the Commissioner via a relaying
23 protocol involving both a "Joiner Router" that is one hop away from the Joiner, and the
24 Border Agent to which the Commissioner is connected. The Joiner and the Commissioner
25 then perform a DTLS handshake using the Joiner's passphrase, which the Commissioner
26 MUST have received out-of-band, typically by being entered by a user. Once the handshake
27 is complete, the shared secret it produces is used to pass the Thread Network's
28 commissioning material from the Joiner Router to the joining device.

29 The Commissioner may also query and set Thread Network parameters, such as the Thread
30 Network name and security configuration.