
Car Connectivity Consortium

Digital Key Release 3

Technical Specification

**Version 1.2.1
(CCC-TS-101)**

CARCONNECTIVITY
consortium®

Copyright © 2011-2023 Car Connectivity Consortium LLC

All rights reserved

Confidential

1 VERSION HISTORY

1 **LEGAL NOTICE**

2 The copyright in this Specification is owned by the Car Connectivity Consortium LLC ("CCC LLC"). Use of
3 this Specification and any related intellectual property (collectively, the "Specification") is governed by; (a)
4 the license terms contained in this Legal Notice and the Car Connectivity Consortium Specification
5 License Agreement (the "License Agreement") for anyone who is not a Member of CCC LLC (a "non-
6 Member"); or (b) the terms contained in this Legal Notice and the CCC LLC Limited Liability Company
7 Agreement (the "LLC Agreement") for any party who is a Member of CCC LLC (a "Member").

8 Use of the Specification by a non-Member is prohibited unless such non-Member has entered into the
9 License Agreement and downloaded this Specification in accordance with the requirements of CCC LLC.
10 The legal rights and obligations of each Member are governed by the LLC Agreement and their applicable
11 Membership Agreement, including without limitation those contained in Article 10 of the LLC Agreement.

12 In addition to the terms of the LLC Agreement and its exhibits and appendices, CCC LLC hereby grants
13 each Member a right to use and to make verbatim copies of the Specification for the purposes of
14 implementing the technologies specified in the Specification to their products ("Implementing Products")
15 under the terms of the LLC Agreement (the "Purpose"). Members are not permitted to make available or
16 distribute this Specification or any copies thereof to non-Members other than to their Affiliates (as defined
17 in the LLC Agreement) and subcontractors but only to the extent that such Affiliates and subcontractors
18 have a need to know for carrying out the Purpose and provided that such Affiliates and subcontractors
19 accept confidentiality obligations similar to those contained in the LLC Agreement. Each Member shall be
20 responsible for the observance and proper performance by such of its Affiliates and subcontractors of the
21 terms and conditions of this Legal Notice and the Agreement. No other license, express or implied, by
22 estoppel or otherwise, to any intellectual property rights are granted to Members herein.

23 **THE SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES, EXPRESS OR IMPLIED,
24 INCLUDING WITHOUT LIMITATION ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A
25 PARTICULAR PURPOSE, NONINFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY
26 RIGHTS, AND COMPLIANCE WITH APPLICABLE LAWS.** Notice for Members: Any use of the
27 Specification not in compliance with the applicable terms of this Legal Notice, the LLC Agreement, and
28 Membership Agreement is prohibited and any such prohibited use may result in termination of the
29 applicable Membership Agreement and other liability permitted by the applicable agreement or by
30 applicable law to CCC LLC or any of its members for patent, copyright, and/or trademark infringement.

31 Notice for Non-Members: Any use of the Specification not in compliance with the applicable terms of this
32 Legal Notice or the License Agreement for non-Members is prohibited and any such prohibited use may
33 result in termination of the non-Member's License Agreement and other liability permitted by the License
34 Agreement or by applicable law to CCC LLC or any of its members for patent, copyright, and/or trademark
35 infringement.

36 **NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED,
37 REGARDING SUCH LAWS OR REGULATIONS. ALL LIABILITY, INCLUDING LIABILITY FOR
38 INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH
39 LAWS RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE
40 SPECIFICATION, EACH MEMBER AND NON-MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST
41 CCC LLC AND ITS MEMBERS RELATED TO USE OF THE SPECIFICATION.**

43 Disclaimers for Members: Each Member hereby acknowledges that its Implementing Products may be
44 subject to various regulatory controls under the laws and regulations of various jurisdictions worldwide.
45 Such laws and regulatory controls may govern, among other things, the combination, operation, use,
46 implementation, and distribution of Implementing Products. Examples of such laws and regulatory

1 controls include, but are not limited to, road safety regulations, telecommunications regulations,
2 technology transfer controls, and health and safety regulations. Each Member is solely responsible for the
3 compliance by their Implementing Products with any such laws and regulations and for obtaining any and
4 all required authorizations, permits, or licenses for their Implementing Products related to such
5 regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the
6 Specification provides any information or assistance in connection with securing such compliance,
7 authorizations, or licenses.

8 Disclaimers for non-Members: Each non-Member that downloads or otherwise obtains access to the
9 Specification acknowledges that its limited copyright license is for the sole purpose of evaluation and
10 such license does not extend to implementations. Products that are the result of implementing the
11 Specification may be subject to various regulatory controls under the laws and regulations of various
12 jurisdictions worldwide. Such laws and regulatory controls may govern, among other things, the
13 combination, operation, use, implementation, and distribution of products. Examples of such laws and
14 regulatory controls include, but are not limited to, road safety regulations, telecommunications
15 regulations, technology transfer controls, and health and safety regulations. Each non-Member that
16 downloads or otherwise obtains access to the Specification is solely responsible for decisions whether to
17 become a Member of CCC LLC and whether to implement products based upon the Specification, and
18 non-Member acknowledges that if such non-Member implements products then it is solely responsible for
19 the compliance with any such laws and regulations and for obtaining any and all required authorizations,
20 permits, or licenses for their products related to such regulations within the applicable jurisdictions. Each
21 non-Member that downloads or otherwise obtains access to the Specification acknowledges that nothing
22 in the Specification provides any information or assistance in connection with securing such compliance,
23 authorizations, or licenses for products that may implement the Specification.

24
25 CCC LLC reserves the right to adopt any changes or alterations to the Specification as it deems necessary or
26 appropriate.
27 **COPYRIGHT © 2011-2022. CCC LLC.**

28

1 TABLE OF CONTENTS

2	VERSION HISTORY.....	2
3	LEGAL NOTICE	3
4	TABLE OF CONTENTS	5
5	LIST OF FIGURES.....	17
6	LIST OF TABLES.....	20
7	LIST OF LISTINGS.....	26
8	ABBREVIATIONS AND ACRONYMS.....	29
9	DEFINITIONS.....	32
10	NOTATIONS	33
11	CODE LISTINGS.....	35
12	1 INTRODUCTION AND SCOPE.....	36
13	2 SYSTEM ARCHITECTURE.....	37
14	2.1 OVERVIEW	37
15	2.2 HIGH-LEVEL FEATURES	37
16	2.3 HIGH LEVEL ARCHITECTURE	37
17	2.4 ACTORS	39
18	2.4.1 <i>Vehicle</i>	39
19	2.4.2 <i>Vehicle NFC Readers [WCC1]</i>	39
20	2.4.3 <i>Vehicle Bluetooth LE Module [WCC2/WCC3]</i>	40
21	2.4.4 <i>Vehicle UWB Module [WCC3]</i>	40
22	2.4.5 <i>Vehicle OEM Server</i>	40
23	2.4.6 <i>Key Tracking Server (KTS)</i>	40
24	2.4.7 <i>Devices</i>	41
25	2.4.8 <i>Device OEM Server</i>	41
26	2.4.9 <i>Relay Server</i>	41
27	2.5 RELATIONSHIPS.....	42
28	2.5.1 <i>Door NFC Reader (3) [WCC1]</i>	42
29	2.5.2 <i>Console NFC Reader (4) [WCC1]</i>	42
30	2.5.3 <i>Bluetooth LE Interface (11) [WCC2/WCC3]</i>	42
31	2.5.4 <i>UWB Interface (12) [WCC3]</i>	43
32	2.5.5 <i>Owner-to-Friend Device Link (via (2), (6), (8), and (7))</i>	43
33	2.5.6 <i>Owner or Friend Device to Vehicle OEM Server (10, 9)</i>	43
34	2.5.7 <i>Telematics Link (1)</i>	43
35	2.5.8 <i>Owner/Friend Device OEM Server Link (2,7)</i>	43
36	2.5.9 <i>Vehicle OEM Server to KTS (5)</i>	43
37	2.5.10 <i>Owner/Friend Device OEM Server to Vehicle OEM Server (6,8)</i>	44
38	2.6 DEVICE STRUCTURE.....	45

1	2.6.1	<i>NFC Component [WCC1]</i>	45
2	2.6.2	<i>Bluetooth Module [WCC2/WCC3]</i>	46
3	2.6.3	<i>UWB Module [WCC3]</i>	46
4	2.6.4	<i>Secure Element (or equivalent)</i>	46
5	2.6.5	<i>Digital Key Applet</i>	46
6	2.6.6	<i>Digital Key Framework</i>	47
7	2.6.7	<i>Vehicle OEM App</i>	47
8	2.6.8	<i>Native App</i>	47
9	2.7	VEHICLE STATES.....	47
10	2.8	DIGITAL KEY USER CLASSES	47
11	2.8.1	<i>Owner</i>	47
12	2.8.2	<i>Friend</i>	48
13	2.9	ACCESS PROFILES	48
14	2.9.1	<i>Owner</i>	48
15	2.9.2	<i>Friend</i>	48
16	2.10	VERSIONING.....	48
17	2.10.1	<i>General</i>	48
18	2.10.2	<i>Domain Versions</i>	48
19	2.10.3	<i>Domain Wise Version Agreement</i>	50
20	2.10.4	<i>Software Updates</i>	53
21	2.10.5	<i>Version Numbers</i>	57
22	2.10.6	<i>Version Introduction</i>	57
23	2.10.7	<i>Version Support</i>	58
24	2.10.8	<i>Version Table</i>	59
25	2.10.9	<i>Version Deprecation</i>	60
26	3	NFC INTERFACE [WCC1].....	61
27	3.1	NFC FUNCTIONAL REQUIREMENTS	61
28	3.1.1	<i>Vehicle</i>	61
29	3.1.2	<i>Device</i>	61
30	3.2	NFC PROCEDURES	62
31	3.2.1	<i>NFC Polling and Link Setup Procedure</i>	62
32	3.2.2	<i>NFC Data Transfer Procedure</i>	63
33	3.2.3	<i>NFC Link Teardown Procedure</i>	63
34	3.2.4	<i>NFC Reset Procedure</i>	64
35	4	DATA STRUCTURE.....	65
36	4.1	APPLET INSTANCE LAYOUT	65
37	4.2	DIGITAL KEY STRUCTURE	66
38	4.3	MAILBOXES	68
39	4.3.1	<i>Private Mailbox</i>	69
40	4.3.2	<i>Confidential Mailbox</i>	74
41	4.3.3	<i>Immobilizer Token</i>	75

1	4.3.4	<i>Mailbox Access Rights</i>	76
2	5	OWNER PAIRING COMMANDS [WCC1/WCC2/WCC3]	77
3	5.1	SUPPORTED COMMANDS FOR OWNER PAIRING	77
4	5.1.1	<i>SELECT Command</i>	78
5	5.1.2	<i>SPAKE2+ REQUEST Command</i>	79
6	5.1.3	<i>SPAKE2+ VERIFY Command</i>	81
7	5.1.4	<i>WRITE DATA Command</i>	83
8	5.1.5	<i>GET DATA Command</i>	89
9	5.1.6	<i>GET RESPONSE Command</i>	90
10	5.1.7	<i>OP CONTROL FLOW Command</i>	91
11	5.2	DATA ELEMENTS	93
12	5.2.1	<i>Certificates</i>	93
13	6	OWNER PAIRING	94
14	6.1	OVERVIEW	94
15	6.2	KEYS AND DATA	94
16	6.3	OWNER PAIRING IMPLEMENTATION	94
17	6.3.1	<i>Phase 0: Preparation</i>	96
18	6.3.2	<i>Phase 1: Initiate Pairing Procedure</i>	96
19	6.3.3	<i>Phase 2: First Session with NFC Reader</i>	97
20	6.3.4	<i>Phase 3: Second Session with NFC Reader</i>	109
21	6.3.5	<i>Phase 4: Finalization of Pairing Procedure (optional)</i>	115
22	6.3.6	<i>Timers</i>	118
23	6.3.7	<i>Pairing Password URL</i>	119
24	7	STANDARD TRANSACTION	121
25	8	FAST TRANSACTION	123
26	9	USER AUTHENTICATION	124
27	9.1	EXPLICIT USER AUTHENTICATION POLICY	125
28	9.2	IMPLICIT USER AUTHENTICATION POLICY	125
29	10	CHECK PRESENCE TRANSACTION	127
30	11	DIGITAL KEY SHARING [WCC1/WCC2]	128
31	11.1	ENCODING	128
32	11.2	SHARING PRINCIPLES	128
33	11.2.1	<i>Definitions</i>	128
34	11.2.2	<i>Principles</i>	129
35	11.3	COMMUNICATION CHANNEL	130
36	11.3.1	<i>Notifications</i>	132
37	11.3.2	<i>Cross-Platform Sharing Invitation</i>	132
38	11.3.3	<i>General API Parameter Definitions</i>	132
39	11.3.4	<i>API Parameter Usage</i>	136
40	11.3.5	<i>Security Considerations</i>	137

1	11.3.6	<i>Device PIN</i>	137
2	11.4	CERTIFICATE CHAIN	143
3	11.5	PREREQUISITES	143
4	11.6	KEY CONFIGURATION	144
5	11.7	ERROR CODES.....	144
6	11.8	KEY SHARING FLOW: STEPS	145
7	11.8.1	<i>Steps 1 and 2 (Owner): Sharing Invitation</i>	145
8	11.8.2	<i>Steps 3 and 4 (Friend): Key Signing Request</i>	147
9	11.8.3	<i>Steps 5 and 6 (Owner): Generate Import Request</i>	149
10	11.8.4	<i>Step 7 (Friend): Endpoint Data Import</i>	152
11	11.8.5	<i>Steps 8 and 9 (Friend): Track Key</i>	154
12	11.8.6	<i>Step 10 (Vehicle OEM Server): Track Key Response</i>	154
13	11.8.7	<i>Step 14 (Friend): First Transaction</i>	154
14	11.9	OWNER DEVICE OEM SERVER NOTIFICATION	155
15	11.9.1	<i>Step 12 (Vehicle OEM Server): eventNotification</i>	155
16	11.9.2	<i>Step 13 (Owner Device OEM Server): eventNotificationResponse</i>	155
17	11.10	KEY TRACKING AND ONLINE ATTESTATION DELIVERY.....	156
18	11.10.1	<i>Online immobilizer token and slot identifier retrieval</i>	157
19	11.11	VEHICLE ATTESTATION	158
20	11.12	ALGORITHMS	158
21	11.13	VERSIONING.....	160
22	12	DELEGATED KEY SHARING	161
23	13	KEY TERMINATION AND DELETION [WCC1/WCC2]	162
24	13.1	KEY TERMINATION SCENARIOS	163
25	13.2	FRIEND KEY REMOTE TERMINATION	166
26	13.2.1	<i>REV_100: Friend key termination in vehicle</i>	167
27	13.2.2	<i>REV_110: Friend key termination in owner Vehicle OEM account</i>	168
28	13.2.3	<i>REV_120: Friend key termination in friend Vehicle OEM account</i>	168
29	13.2.4	<i>REV_130: Friend key termination on owner device natively</i>	169
30	13.2.5	<i>REV_140: Friend key termination on owner device in Vehicle OEM app</i>	169
31	13.2.6	<i>REV_150: Friend key termination based on expiry date of the key</i>	169
32	13.2.7	<i>REV_160: Friend key termination by Device OEM (device security issue)</i>	171
33	13.2.8	<i>REV_170: Friend key termination due to remote wipe of device</i>	172
34	13.2.9	<i>REV_180: Friend key termination due to deletion of personal data in friend Vehicle OEM account</i>	173
35	13.3	FRIEND KEY LOCAL TERMINATION.....	173
36	13.3.1	<i>REV_200: Friend key termination on friend device natively</i>	173
37	13.3.2	<i>REV_210: Friend key termination on friend device in Vehicle OEM app</i>	173
38	13.3.3	<i>REV_220: Friend key termination due to local wipe of device</i>	174
39	13.4	FRIEND KEY REMOTE SUSPEND/RESUME	174
40	13.4.1	<i>REV_300: Friend key temporary suspension in Device OEM account</i>	174

1	13.4.2	<i>REV_310: Friend key resume in Device OEM account.....</i>	175
2	13.4.3	<i>Resume attestation</i>	176
3	13.5	OWNER KEY REMOTE TERMINATION.....	177
4	13.5.1	<i>REV_400: Owner key deletion in vehicle UI (change device).....</i>	177
5	13.5.2	<i>REV_410: Owner key termination by Device OEM (security issue)</i>	179
6	13.5.3	<i>REV_420: Owner key termination due to remote wipe of device</i>	179
7	13.5.4	<i>REV_500: Owner key termination on owner device natively (delete pass for Digital Key).....</i>	179
9	13.5.5	<i>REV_510: Owner key termination on owner device in Vehicle OEM app</i>	179
10	13.5.6	<i>REV_520: Owner key termination due to local wipe of device</i>	179
11	13.5.7	<i>REV_600/610: Owner key temporary suspension in device lost mode in Device OEM account</i>	180
13	13.6	OWNER DEVICE UNPAIRING.....	181
14	13.6.1	<i>REV_700: Unpairing in vehicle UI (sell vehicle).....</i>	181
15	13.6.2	<i>REV_710: Unpairing on owner device in Vehicle OEM app</i>	182
16	13.6.3	<i>REV_720: Unpairing in owner Vehicle OEM account (sale of vehicle)</i>	183
17	13.6.4	<i>REV_730: Unpairing based on cancellation or expiry date of the Digital Key service</i>	183
19	13.6.5	<i>REV_740: Unpairing by Vehicle OEM (vehicle stolen)</i>	183
20	13.6.6	<i>REV_750: Unpairing by Vehicle OEM (security breach in vehicle).....</i>	183
21	13.6.7	<i>REV_760: Unpairing by Vehicle OEM (garage service process)</i>	183
22	13.6.8	<i>REV_770: Unpairing due to deletion of personal data in owner Vehicle OEM account</i>	183
24	13.6.9	<i>REV_800: Unbinding of vehicle from owner account in vehicle UI (unpairing, owner sale of vehicle)</i>	184
25	13.6.10	<i>REV_810: Unbinding of vehicle from owner account in owner Vehicle OEM account (unpairing, owner sale of vehicle).....</i>	184
26	13.6.11	<i>REV_820: Unbinding of vehicle from owner account in Vehicle OEM app (unpairing, owner sale of vehicle)</i>	184
30	13.7	TERMINATION IN VEHICLE.....	184
31	13.7.1	<i>Rules.....</i>	184
32	13.7.2	<i>Example.....</i>	185
33	14	AUTHENTICATION AND PRIVACY	189
34	14.1	AUTHENTICATION AND PRIVACY KEYS	189
35	14.1.1	<i>Usage</i>	189
36	14.1.2	<i>Certificate Chains</i>	190
37	14.2	REMOTE TERMINATION REQUESTS	192
38	14.2.1	<i>Example of RTR Signature Verification.....</i>	195
39	14.3	ENCRYPTION AND SIGNATURE VERIFICATION SCHEMES	196
40	14.4	OEM APP DATA ATTESTATION	196
41	15	DIGITAL KEY APPLET	197
42	15.1	INTRODUCTION	197

1	15.2 KEYS AND DATA.....	197
2	15.3 APPLET IMPLEMENTATION	198
3	15.3.1 <i>Introduction</i>	198
4	15.3.2 <i>Commands</i>	210
5	15.3.3 <i>Security</i>	253
6	15.3.4 <i>Optimization</i>	256
7	15.4 HELPER METHOD	256
8	15.4.1 <i>Hierarchy</i>	257
9	15.5 VEHICLE IMPLEMENTATION	268
10	15.5.1 <i>Security Guidelines</i>	268
11	15.5.2 <i>Optimizations</i>	268
12	16 CERTIFICATE.....	269
13	16.1 GENERAL	269
14	16.2 CERTIFICATES AND RELATIONSHIPS.....	269
15	16.2.1 <i>[A] – SE Root CA Certificate</i>	272
16	16.2.2 <i>[B] – SE Root Certificate</i>	272
17	16.2.3 <i>[C] – Instance CA Attestation (signed by SE Root) per Vehicle OEM</i>	272
18	16.2.4 <i>[D] – Device OEM CA Certificate</i>	272
19	16.2.5 <i>[E] – Instance CA Certificate (Signed by Device OEM) per Vehicle OEM</i>	273
20	16.2.6 <i>[F] – Device OEM CA Certificate (Signed by Vehicle OEM)</i>	273
21	16.2.7 <i>[G] – Digital Key per vehicle</i>	273
22	16.2.8 <i>[H] – Digital Key Certificate</i>	273
23	16.2.9 <i>[J] – Vehicle OEM CA Certificate</i>	273
24	16.2.10 <i>[K] – Vehicle Public Key Certificate</i>	274
25	16.2.11 <i>[L] – Digital Key Creation Data</i>	274
26	16.2.12 <i>[M] – Vehicle OEM CA Certificate (signed by Device OEM)</i>	274
27	16.2.13 <i>[N] – Instance CA Attestation (self-signed) per Vehicle OEM</i>	274
28	16.3 CERTIFICATE SIZE RESTRICTIONS	275
29	16.4 DEVICE CERTIFICATE CHAIN	276
30	16.5 VEHICLE CERTIFICATE CHAIN	276
31	16.6 SUPPORTED VERIFICATION CHAIN	276
32	16.7 OWNER PAIRING CERTIFICATE CHAIN	276
33	16.7.1 <i>Device and Vehicle</i>	276
34	16.8 KEY SHARING CERTIFICATE CHAIN	277
35	17 SERVER-TO-SERVER COMMUNICATIONS AND API.....	278
36	17.1 INTRODUCTION	278
37	17.2 API DESIGN	278
38	17.3 SECURITY	278
39	17.4 URL SCHEME	278
40	17.5 HEALTHCHECK.....	279

1	17.6	USER INTERFACE	279
2	17.6.1	Description.....	279
3	17.7	SERVER APIs	279
4	17.7.1	API - Track Key	279
5	17.7.2	Manage Key	286
6	17.7.3	API – eventNotification.....	290
7	17.7.4	API – Healthcheck	294
8	17.7.5	API – versionUpdate.....	295
9	17.8	STRUCTURE DEFINITIONS	298
10	17.8.1	Request Headers	298
11	17.8.2	Response Headers.....	298
12	17.8.3	Key Type.....	299
13	17.8.4	EncryptedDataContainer.....	299
14	17.8.5	Unencrypted uiBundle	300
15	17.8.6	Digital Key Status	301
16	17.8.7	Action for manageKey.....	302
17	17.8.8	Key Actions	302
18	17.8.9	Event Type of Notification	303
19	17.8.10	Event Data of Event Notification	305
20	17.8.11	Unencrypted Event Notification Data.....	306
21	17.8.12	Shared Key.....	307
22	17.8.13	Key Configuration.....	307
23	17.8.14	Entitlement	307
24	17.8.15	Unencrypted vehicleMobilizationData Fields	308
25	17.8.16	Supported Entitlements	308
26	17.8.17	Device Type.....	309
27	17.8.18	UI Elements.....	309
28	17.8.19	Activation Options	309
29	17.8.20	Supported approved sharing methods	310
30	17.9	HTTP STATUS CODES.....	310
31	17.10	SUB STATUS CODES	311
32	17.11	ECC ENCRYPTION FOR DEVICE AND SERVER	313
33	17.11.1	Frame/Packet Contents.....	313
34	17.11.2	Encryption Process	314
35	17.11.3	Decryption Process.....	314
36	17.11.4	Key Derivation Function.....	315
37	17.11.5	Symmetric Encryption Key.....	315
38	17.11.6	Initialization Vector Derivation.....	316
39	17.11.7	EC Public Key Point Encoding in Uncompressed Form	316
40	17.12	VERSIONING.....	316
41	17.12.1	General	316
42	17.13	EXAMPLE	316

1	<i>17.13.1 Keys</i>	316
2	<i>17.13.2 Message to encrypt</i>	317
3	<i>17.13.3 Encryption process</i>	317
4	<i>17.13.4 Decryption process</i>	317
5	18 SECURITY	319
6	18.1 SPAKE2+ PROTOCOL DESCRIPTION.....	319
7	18.1.1 General	319
8	18.1.2 Execution.....	319
9	18.1.3 Verification	321
10	18.1.4 Summary	321
11	18.1.5 SPAKE2+ Constant Definitions.....	322
12	18.2 PRIVACY PROPERTIES	323
13	18.2.1 NFC Transaction [WCC1/WCC2].....	323
14	18.2.2 Vehicle OEM Server	323
15	18.2.3 User Privacy	323
16	18.2.4 Multi-Vehicle OEM Deployment.....	323
17	18.2.5 Error Handling	324
18	18.3 CRYPTOGRAPHIC ALGORITHMS	324
19	18.4 CRYPTOGRAPHIC PROTOCOLS.....	324
20	18.4.1 Server Password Generation	324
21	18.4.2 Vehicle-side public EC Point Generation.....	324
22	18.4.3 Device-side public EC Point Generation.....	324
23	18.4.4 Vehicle-side computation of Shared Secret	325
24	18.4.5 Device-side computation of Shared Secret	325
25	18.4.6 Derivation of Evidence Keys.....	325
26	18.4.7 Vehicle-side computation of Evidence	325
27	18.4.8 Device-side computation of Evidence.....	326
28	18.4.9 Derivation of System Keys [WCC1/WCC2/WCC3].....	326
29	18.4.10 Generate Attestation Signature.....	326
30	18.4.11 Validate Attestation or Certificate	326
31	18.4.12 Secure Channel Command Encryption and Authentication	326
32	18.4.13 Secure Channel Response Encryption and Authentication.....	327
33	18.5 ERROR HANDLING [WCC1/WCC2/WCC3]	327
34	18.6 SECURE ELEMENT (SE).....	327
35	19 BLUETOOTH LOW ENERGY (LE) INTERFACE [WCC2/WCC3]	329
36	19.1 BLUETOOTH LE FUNCTIONAL REQUIREMENTS	329
37	19.1.1 Vehicle.....	329
38	19.1.2 Device	329
39	19.2 BLUETOOTH LE PROCEDURES	330
40	19.2.1 Owner Pairing Connection Establishment	330
41	19.2.2 Bluetooth Encryption	342

1	19.2.3	<i>Passive Entry: Bluetooth LE Setup:</i>	342
2	19.3	DK MESSAGE FORMAT	346
3	19.3.1	<i>UWB Ranging Service Message.....</i>	348
4	19.3.2	<i>SE Message</i>	356
5	19.3.3	<i>Supplementary Service Message - Time Sync.....</i>	357
6	19.3.4	<i>Supplementary Service Message - First Approach.....</i>	359
7	19.3.5	<i>Supplementary Service Message - RKE.....</i>	360
8	19.3.6	<i>Vehicle OEM App Message</i>	361
9	19.3.7	<i>Head Unit Pairing Message.....</i>	361
10	19.3.8	<i>DK Event Notification.....</i>	363
11	19.4	TIME SYNCHRONIZATION.....	375
12	19.4.1	<i>Definitions.....</i>	375
13	19.4.2	<i>General Description.....</i>	378
14	19.4.3	<i>Device Uncertainty</i>	381
15	19.4.4	<i>“Bluetooth LE Timesync” Procedures.</i>	382
16	19.4.5	<i>“Bluetooth LE Timesync” message flow examples</i>	385
17	19.4.6	<i>Conditions for a “Bluetooth LE Timesync” procedure</i>	387
18	19.5	DIGITAL KEY – FLOWS	388
19	19.5.1	<i>Owner Pairing Flow</i>	388
20	19.5.2	<i>Head Unit Pairing Flow</i>	396
21	19.5.3	<i>Passive Entry</i>	398
22	19.5.4	<i>URSK Management.....</i>	398
23	19.5.5	<i>Flow Selection - Establish Secure Ranging.....</i>	400
24	19.5.6	<i>Standard Transaction over Bluetooth LE</i>	406
25	19.5.7	<i>Engine Start</i>	408
26	19.5.8	<i>Friend - Sharing & First Approach.....</i>	408
27	19.5.9	<i>RKE Function Flow</i>	411
28	19.5.10	<i>Bluetooth LE Activation Flow.....</i>	416
29	19.6	DIGITAL KEY - PREFERENCE MANAGEMENT.....	417
30	19.6.1	<i>Preference Management: Connected Vehicles < Connection Limit</i>	417
31	19.6.2	<i>Preference Management: Connected Vehicles >= Connection Limit</i>	419
32	19.7	SUBEVENT HANDLING	419
33	19.7.1	<i>Command Complete SubEvent Code Handling</i>	419
34	19.7.2	<i>Ranging Session Status Changed SubEvent.....</i>	420
35	19.8	ERROR HANDLING	423
36	19.8.1	<i>SE Transaction Recovery.....</i>	423
37	20	UWB MAC AND CHANNEL ACCESS [WCC3].....	425
38	20.1	UWB MAC ARCHITECTURE.....	425
39	20.1.1	<i>Overview</i>	425
40	20.2	MAC TIME GRID	427
41	20.3	MAC TIME GRID SYNCHRONIZATION	432

1	20.4	HOPPING FLAG AND ROUND INDEX DETERMINATION	433
2	20.5	MAC PROTOCOL	435
3	20.5.1	<i>Ranging Exchange Sequence</i>	435
4	20.5.2	<i>Ranging Session Setup</i>	440
5	20.5.3	<i>MAC Control Channel (Over Bluetooth LE)</i>	443
6	20.5.4	<i>UWB MAC Configuration</i>	444
7	20.6	STS INDEX INCREMENTATION	446
8	20.6.1	<i>Rules for STS Index Incrementation</i>	447
9	20.6.2	<i>STS Incrementation and Packet Mapping within a Ranging Round</i>	447
10	20.6.3	<i>Calculation of STS Index</i>	448
11	21	UWB PHY [WCC3]	449
12	21.1	INTRODUCTION	449
13	21.2	UWB PHY BLOCK DIAGRAM	449
14	21.3	UWB PACKET FORMAT	449
15	21.4	UWB FRAME ELEMENTS	450
16	21.4.1	<i>BPRF SYNC</i>	451
17	21.4.2	<i>BPRF SFD</i>	452
18	21.4.3	<i>Scrambled Timestamp Sequence (STS)</i>	452
19	21.4.4	<i>BPRF PHR</i>	452
20	21.4.5	<i>Standard Compliant Data Field</i>	453
21	21.5	PULSE SHAPE COMBINATIONS	453
22	21.5.1	<i>Symmetrical Root-Raised-Cosine Pulse – PulseShape 0x0</i>	453
23	21.5.2	<i>Precursor-Free Pulse – PulseShape 0x1</i>	454
24	21.5.3	<i>PulseShape Combinations</i>	454
25	21.6	RANGING MARKER	455
26	21.7	UWB RF	455
27	21.7.1	<i>Operating frequency bands and Channel assignments</i>	455
28	21.7.2	<i>Frequency Source Requirements</i>	455
29	22	UWB SECURITY [WCC3]	456
30	22.1	CRYPTOGRAPHY	456
31	22.1.1	<i>Deriving the STS Sequence from the DRBG</i>	456
32	22.1.2	<i>SP0 Frames Encryption</i>	456
33	22.1.3	<i>Key Derivation Functions (KDFs)</i>	457
34	22.1.4	<i>UWB Tracking Prevention Between Ranging Recoveries</i>	460
35	22.2	UWB RANGING	461
36	22.2.1	<i>STS Index Management</i>	461
37	22.3	UWB MODULE	462
38	23	REFERENCES	463
39	APPENDIX A.	465
40	A.1.	<i>Standard Transaction Cryptography Flow</i>	465

1	A.2.	<i>Fast Transaction Cryptography Flow</i>	468
2	A.3.	<i>Standard Transaction Cryptography Flow (with Reader Intent for Fast</i> <i>Transaction)</i>	470
3	A.4.	<i>X.509 Schemes</i>	474
4	APPENDIX B.....		479
5	B.1.	<i>Instance Configurations</i>	479
6	B.2.	<i>Certificate Field Formats</i>	480
7	APPENDIX C.....		484
8	C.1.	<i>External CA Certificate</i>	484
9	C.2.	<i>Vehicle OEM Intermediate Certificate</i>	485
10	C.3.	<i>Vehicle OEM Key/Leaf Certificate</i>	486
11	C.4.	<i>Vehicle OEM Privacy Encryption Certificate</i>	487
12	C.5.	<i>Vehicle OEM Signature Verification Certificate</i>	488
13	APPENDIX D.	OWNER PAIRING TEST VECTORS [TO BE UPDATED]	490
14	D.1.	<i>Derivation of z0, z1 with Scrypt</i>	490
15	D.2.	<i>Computation of w0, w1</i>	490
16	D.3.	<i>Computation of L</i>	491
17	D.4.	<i>Computation of X</i>	491
18	D.5.	<i>Computation of Y</i>	492
19	D.6.	<i>Computation of Z, V (by device)</i>	493
20	D.7.	<i>Computation of Z, V (by vehicle)</i>	494
21	D.8.	<i>Derivation of K, CK, SK</i>	496
22	D.9.	<i>Derivation of Evidence Keys K1, K2</i>	497
23	D.10.	<i>Computation of Evidences M1, M2</i>	497
24	D.11.	<i>Derivation of System Keys</i>	498
25	APPENDIX E.	NFC-F SUPPORT [WCC1]	498
26	E.1.	<i>Introduction</i>	498
27	E.2.	<i>Device Requirement</i>	499
28	E.3.	<i>Preconditions</i>	500
29	E.4.	<i>Commands and responses</i>	500
30	E.5.	<i>Protocol Operation</i>	502
31	APPENDIX F.	DIGITAL KEY FRAMEWORK API	505
32	F.1.	<i>Overview</i>	505
33	F.2.	<i>Functional Requirements</i>	505
34	APPENDIX G.	506
35	G.1.	<i>Ranging Session Parameter Tables [WCC3]</i>	506
36	G.2.	<i>Hopping Sequence</i>	507
37	G.3.	<i>Default Hopping Sequence:</i>	507
38	G.4.	<i>AES-Based Hopping Sequence:</i>	507
39	APPENDIX H.	[WCC3].....	507
40	APPENDIX I.	[WCC3].....	508
41	I.1.	<i>PulseShape 0x0</i>	508

1	<i>I.2.</i>	<i>PulseShape 0x2</i>	509
2	APPENDIX J.	[WCC3].....	511
3	<i>J.1.</i>	<i>UWB Test Vector</i>	511
4	<i>J.2.</i>	<i>UWB Test Vector without hopping</i>	512
5	<i>J.3.</i>	<i>UWB Test Vector with continuous hopping</i>	516
6			

1 LIST OF FIGURES

2	Figure 2-1: Digital Key Architecture with Actors and Their Relationships	38
3	Figure 2-2: Device Functional Elements.	45
4	Figure 2-3: Domain Versions	49
5	Figure 2-4: D-VS Agreement for Key Tracking	52
6	Figure 2-5: Version Agreement after Device Software Update	54
7	Figure 2-6: Example – manageKey() usage based on D-VS agreed version	55
8	Figure 2-7: V-OD-FW Agreement after Vehicle SW Update	56
9	Figure 4-1: Applet Instance Layout	65
10	Figure 4-2: Friend Digital Key Structure and Entitlements Attestation	66
11	Figure 4-3: Mailboxes Read/Write Permissions	69
12	Figure 4-4: Owner Device Private Mailbox Signaling	70
13	Figure 4-5: Slot Identifier Bitmap Assignment	71
14	Figure 4-6: Owner Device Immobilizer Token Signaling	72
15	Figure 4-7: Slot Identifier List to Confidential Mailbox mapping	73
16	Figure 6-1: Owner Pairing NFC Exchanges	95
17	Figure 6-2: Owner Pairing Flow - Phase 0/1: Preparation/Initiation	96
18	Figure 6-3: Owner Pairing Flow - Phase 2: First NFC Session	98
19	Figure 6-4: SPAKE2+ Flow	100
20	Figure 6-5: Key Creation Data Transfer to Device	101
21	Figure 6-6: Key Creation Data Transfer to Device	102
22	Figure 6-7: Key Creation Info Retrieval by Vehicle	104
23	Figure 6-8: Owner Pairing Flow – Phase 3: Second NFC Session	111
24	Figure 6-9: Error Management of Different Phases in Owner Pairing	115
25	Figure 6-10: Owner Pairing Flow – Phase 4: Finalization	116
26	Figure 6-11: Vehicle and Device Transaction Timeout	119
27	Figure 7-1: Standard Transaction Flow	122
28	Figure 8-1: Fast Transaction Flow	123
29	Figure 10-1: Check Presence Transaction	127
30	Figure 13-1: REV_100: Friend Key Termination in Vehicle	167
31	Figure 13-2: REV_100a: Friend Key Termination in Vehicle (Vehicle Required to be Online)	167
32	Figure 13-3: REV_110/120: Friend Key Termination in Owner/Friend Vehicle OEM Account	168
33	Figure 13-4: REV_130/140: Friend Key Termination on Owner Device Natively/in Vehicle OEM App	169
34	Figure 13-5: REV_150: Friend Key Termination Based on Expiry Date of the Key	170
35	Figure 13-7: REV_160: Friend Key Termination by Device OEM (Device Security Issue)	171
36	Figure 13-8: REV_160a: Friend Key termination by Device OEM and Friend Device is Offline	171
37	Figure 13-9: REV_170: Friend Key Termination Due to Remote Wipe of Device	172
38	Figure 13-10: REV_200/210: Friend Key Termination on Friend Device Natively/in Vehicle OEM App	173
39	Figure 13-11: REV_220: Friend Key Termination Due to Local Wipe of Device	174
40	Figure 13-12: REV_300: Friend Key Suspension by Device OEM Account	175
41	Figure 13-13: REV_310: Friend Key Resume in Device OEM Account or on Device	176
42	Figure 13-14: REV_310: Friend Key Resume Using ResumeAttestation	176

1	Figure 13-15: REV_400: Owner Key Deletion in Vehicle UI (Change of Device)	178
2	Figure 13-16: REV_400a: Owner Key Deletion in Vehicle UI (Change of Device)	179
3	Figure 13-17: REV_600: Owner key suspension due to device reported lost/stolen	180
4	Figure 13-18: REV_610: Owner key resumption after device reported lost/stolen	181
5	Figure 13-19: REV_700: Unpairing in Vehicle UI (Sale of Device)	182
6	Figure 13-20: REV_700a: Unpairing in Vehicle UI (Vehicle Required to be Online)	182
7	Figure 13-21: REV_710: Unpairing in Vehicle OEM App on Owner Device	183
8	Figure 13-22: Example Step 1: After Owner Pairing	185
9	Figure 13-23: Example Step 2: Device Fully Refilled	186
10	Figure 13-24: Example Step 3: Key Shared with Friend	186
11	Figure 13-25: Example Step 4: Refill of Shared Slot Identifier	187
12	Figure 13-26: Example Step 5a: Key Termination in Vehicle	187
13	Figure 13-27: Example Step 5b: Key Termination in Device	188
14	Figure 14-1: Message Authentication and Privacy Encryption	189
15	Figure 14-2: Authentication and Privacy Encryption Certificate Chain	191
16	Figure 15-1: Authentication Command Flows Over Contactless Interface	201
17	Figure 15-2: Authentication Command Flows Over Wired Interface	201
18	Figure 16-1: Variant 1 Certification Chain Model	270
19	Figure 16-2: Variant 2 Certification Chain Model	271
20	Figure 19-1: Bluetooth LE Link Layer Connection Establishment.	331
21	Figure 19-2: L2CAP Connection-Oriented Channel.	331
22	Figure 19-3: Bluetooth LE Pairing and Encryption Setup.	333
23	Figure 19-4: Passive Entry Flow Diagram.	343
24	Figure 19-5: Connection performance in relationship with device, transmitter, and receiver requirements.	345
25	Figure 19-7: Synchronization Methods.	379
26	Figure 19-8: Synchronization state between device and anchor.	380
27	Figure 19-9: Time Synchronization Uncertainty Flow Diagram.	382
28	Figure 19-10: Successful Bluetooth LE timeSync at BT connection (Procedure 0).	383
29	Figure 19-11: Successful Bluetooth LE timeSync triggered by vehicle (Procedure 1).	384
30	Figure 19-12: Unsuccessful Timesync message.	384
31	Figure 19-13: Bluetooth LE Timesync message flow example 1.	385
32	Figure 19-14: Time Synchronization flow diagram example 2.	386
33	Figure 19-15: Bluetooth LE Time Sync message flow example 3.	387
34	Figure 19-16: Owner Pairing Flow for Bluetooth LE/UWB.	389
35	Figure 19-17: Bluetooth LE Secure OOB Pairing Prep.	392
36	Figure 19-18: Capability Exchange.	394
37	Figure 19-20: Head Unit Pairing Flow Diagram.	397
38	Figure 19-21: URSK Derivation Flow in a dedicated Standard Transaction	399
39	Figure 19-22: Flow Selection for Establishing Secure Ranging	401
40	Figure 19-23: Secure Ranging Setup Flow.	402
41	Figure 19-24: Sub-Optimal Flow.	403
42	Figure 19-25: Ranging Session State Machine.	404
43	Figure 19-26: Ranging Suspend Accepted Flow.	404
44	Figure 19-27: Ranging Suspend Delayed Flow.	405

1	Figure 19-28: Ranging Recovery Flow.	406
2	Figure 19-29: Standard transaction over Bluetooth LE.	407
3	Figure 19-31: Friend First Approach Flow.	410
4	Figure 19-32: RKE Flow for an Event-based RKE Action.	412
5	Figure 19-33: RKE Flow for an Enduring RKE action.	414
6	Figure 19-34: Vehicle Function Status Retrieval and Event-based Update.	416
7	Figure 19-35: First Approach Activation.	417
8	Figure 19-36: Device Preference Management.	418
9	Figure 19-37: Required Capability Exchanged.	420
10	Figure 19-38: URSK Not Found.	421
11	Figure 19-40: Recovery Failed Flow.	422
12	Figure 19-41: URSK Refresh due to Status Word 6484 _h in CREATE RANGING KEY Response.	423
13	Figure 20-1: Digital Key RAN Concept.	426
14	Figure 20-2: Digital Key One-to-Many Ranging Protocol.	428
15	Figure 20-3: Overview of MAC Grid (See Table G- 1).	431
16	Figure 20-4: Overview of Block Synchronization Approach.	433
17	Figure 20-5: Example of UWB Message Flow for MAC Protocol.	435
18	Figure 20-6: Example of MAC Parameter Negotiation and Setting.	443
19	Figure 20-7: SP0 Packet Content.	444
20	Figure 20-8: Basic Principles of STS Incrementation.	447
21	Figure 21-1: Block diagram for IEEE 802.15.4z HRP UWB PHY.	449
22	Figure 21-2: SP3	450
23	Figure 21-3: SP0	450
24	Figure 21-4: PHR bit assignment.	452
25	Figure 21-5: Data Field Encoding for Standard PHY.	453
26	Figure 22-1: Overview of the KDF used and its relationship.	457

1 LIST OF TABLES

2	Table 4-1: Signaling Bitmap Decoding	71
3	Table 4-2: Private Mailbox Content	73
4	Table 4-3: Confidential Mailbox Content.....	75
5	Table 4-4: Mailbox Access Rights.....	76
6	Table 5-1: Owner Pairing Command Set.....	77
7	Table 5-2: Generic Status Words.....	77
8	Table 5-3: Response to SELECT Command	78
9	Table 5-4: SPAKE2+ REQUEST Command Fields.....	79
10	Table 5-5: SPAKE2+ REQUEST Response Fields	79
11	Table 5-6: SPAKE2+_REQUEST Response Error Status Words	81
12	Table 5-7: SPAKE2+ VERIFY Command Fields	81
13	Table 5-8: SPAKE2+ VERIFY Response Fields.....	82
14	Table 5-9: SPAKE2+ VERIFY Response Error Status Words.....	82
15	Table 5-10: WRITE DATA Response Error Status Words	83
16	Table 5-11: Objects for Digital Key Creation.....	83
17	Table 5-12: Objects for Device Digital Key Certificate	84
18	Table 5-13: Mailbox Mapping	86
19	Table 5-14: Device Configuration Data.....	87
20	Table 5-15: GET DATA Response Error Status Words	89
21	Table 5-16: GET DATA Response Decrypted Payload for tag 7F20 _h	90
22	Table 5-17: GET DATA Response Decrypted Payload for tag 7F22 _h	90
23	Table 5-18: GET DATA Response Decrypted Payload for tag 7F24 _h	90
24	Table 5-19: GET DATA Response Decrypted Payload for Tag D3 _h	90
25	Table 5-20: GET RESPONSE Response Error Status Words	91
26	Table 5-21: OP CONTROL FLOW P1 Parameters.....	92
27	Table 5-22: OP CONTROL FLOW P2 Parameters for P1=10 _h (continue)	92
28	Table 5-23: OP CONTROL FLOW P2 Parameters for P1=11 (end with success)	92
29	Table 5-24: OP CONTROL FLOW P2 Parameters for P1=12 (end with failure)	92
30	Table 5-25: Certificates	93
31	Table 6-1: Error SW Condition List 1.	109
32	Table 6-2: Owner Key Tracking Request	112
33	Table 6-3: Owner Key Tracking Response Parameters	112
34	Table 6-4: Recommended Minimum Vehicle and Device Timeout Values	119
35	Table 6-5: Pairing Password URL Syntax	120
36	Table 9-1: AUTH0 Command Transaction Type Coding	124
37	Table 9-2: User Authentication Policies	125
38	Table 11-4: Key Configuration.....	138
39	Table 11-6: Key Creation Request.....	139
40	Table 11-7: Key Signing Request.....	142
41	Table 11-9: Error Code Message	144
42	Table 11-10: Error Codes	145

1	Table 11-12: External CA Certificate Container	148
2	Table 11-13: Instance CA Certificate Container	148
3	Table 11-14: Endpoint Certificate Container.....	148
4	Table 11-16: Import Request	150
5	Table 11-18: Attestation Package.....	153
6	Table 11-19: Key Tracking and Online Attestation Delivery Request	156
7	Table 11-20: Key Tracking Response	157
8	Table 11-21: Vehicle Attestation Data Fields.....	158
9	Table 11-22: Vehicle Attestation.....	158
10	Table 13-1: Vehicle-triggered Key Termination	163
11	Table 13-2: Device-triggered Key Termination	163
12	Table 13-3: Vehicle OEM-triggered Key Termination.....	163
13	Table 13-4: Device OEM-triggered Key Termination.....	164
14	Table 13-6:Detailed Key Termination Use Cases	165
15	Table 13-7: Resume Attestation	177
16	Table 14-1: Authentication and Privacy Keys	190
17	Table 14-2: Remote Termination Request Data Fields.....	193
18	Table 14-3: Remote Termination Request Owner Device.....	193
19	Table 14-4: Remote Termination Request from Vehicle OEM Server.....	194
20	Table 14-5: OEM App Data Attestation Input.....	196
21	Table 15-1: Command Availability on Interfaces	199
22	Table 15-2: Generic Status Words.....	203
23	Table 15-3: Coding of the Different Ranges of Class Byte Values	204
24	Table 15-4: INSTALL for INSTALL Content of Tag C9	204
25	Table 15-5: Applet Implementation Options	205
26	Table 15-6: Values for Protocol Data Type A (tag ‘86’)	206
27	Table 15-7: Values for Protocol Data Type B (tag 87 _h)	207
28	Table 15-9: Coding of the notification	208
29	Table 15-10: Notification Context.....	208
30	Table 15-11: Value and Presence of the Different Fields in the HCI Notification Event.....	208
31	Table 15-12: SELECT Response Fields	211
32	Table 15-13: Endpoint Configuration.....	211
33	Table 15-14: Setting of Tag 46 _h and Tag 47 _h by vehicle for various wireless capability combinations.....	213
34	Table 15-15: Setting of Tag 46 _h and Tag 47 _h by the owner device for the key sharing.....	214
35	Table 15-16: Endpoint verification information	215
36	Table 15-17: Endpoint Termination Request.....	219
37	Table 15-18: Endpoint Termination Attestation Data Fields	220
38	Table 15-19: Endpoint Termination Attestation	220
39	Table 15-20: Deletion Request	221
40	Table 15-21: AUTHORIZE ENDPOINT Command Payload.....	222
41	Table 15-22: AUTHORIZE ENDPOINT Internal Buffer Content Before Processing (offset 0)	226
42	Table 15-23: AUTHORIZE ENDPOINT Attestation Data Fields	226
43	Table 15-24: AUTHORIZE ENDPOINT Internal Buffer Content After Processing (offset 0).....	226
44	Table 15-25: View Parameters	228

1	Table 15-26: View Endpoint Identifier Response in Internal Buffer.....	229
2	Table 15-27: View Instance CA Response in Internal Buffer	229
3	Table 15-28: AUTH0 P1, P2 Parameters.....	230
4	Table 15-29: AUTH0 Command Payload	230
5	Table 15-30: AUTH0 Response Payload.....	231
6	Table 15-31: AUTH1 Vehicle Authentication Data Fields	232
7	Table 15-32: AUTH1 Command Payload	233
8	Table 15-33: AUTH1 Response Payload Before Encryption	233
9	Table 15-35: PRESENCE0 Response Payload.....	234
10	Table 15-36: PRESENCE1 Command Payload	235
11	Table 15-37: PRESENCE1 Vehicle Authentication Data Fields.....	235
12	Table 15-38: PRESENCE1 Response Payload Before Encryption	235
13	Table 15-39: READ BUFFER Command Payload	236
14	Table 15-40: Exchange Command Decrypted Payload	238
15	Table 15-41: Exchange Response Decrypted Payload	239
16	Table 15-42: CONTROL FLOW P1 Parameters.....	240
17	Table 15-43: CONTROL FLOW P2 Parameters.....	240
18	Table 15-44: CONTROL FLOW P1/P2 Values for Applet.....	241
19	Table 15-45: CREATE ENCRYPTION KEY Command Payload.....	242
20	Table 15-46: Encryption Key Attestation Data Fields.....	243
21	Table 15-47: Encryption Key Attestation	243
22	Table 15-48: GET PRIVATE DATA Command Payload.....	244
23	Table 15-49: SET PRIVATE DATA Command Payload	245
24	Table 15-50: SET CONFIDENTIAL DATA Command Payload	245
25	Table 15-51: SET CONFIDENTIAL DATA Internal Buffer Content Before Processing	245
26	Table 15-52: SETUP ENDPOINT Command Payload	246
27	Table 15-53: SETUP INSTANCE Command Payload.....	248
28	Table 15-54: SIGN Command Payload	249
29	Table 15-55: SIGN Command Payload	249
30	Table 15-56: SIGN Data Fields	249
31	Table 15-57: SIGN Response	250
32	Table 15-58: MANAGE UA Command Payload	250
33	Table 15-59: Delete Ranging Keys Request.....	252
34	Table 15-60: Receiver Endpoint Key Attestation Signed by Sender	263
35	Table 15-61: Arbitrary Data Attestation	265
36	Table 19-1: AdvA field of ADV_IND.....	335
37	Table 19-2: AdvData field of ADV_IND.....	335
38	Table 19-3: Definition of IntentConfiguration byte.....	335
39	Table 19-4: Mandatory fields in Pairing Request	335
40	Table 19-5: Mandatory fields in Pairing Response.....	336
41	Table 19-6: DK Service UUID.....	336
42	Table 19-7: SPSM Characteristic declaration.....	336
43	Table 19-8: SPSM Characteristic value declaration.....	337
44	Table 19-19: DK Message Format.....	346

1	Table 19-20: Message Header definition.....	346
2	Table 19-21: Message Type definition.....	346
3	Table 19-22: Payload Header definition.....	347
4	Table 19-23: Message Type and its associated messages.....	347
5	Table 19-24: Ranging_Capability_RQ message and its parameters.....	349
6	Table 19-25: Definition of the parameters for Ranging_Capability_RQ.....	349
7	Table 19-26: Ranging_Capability_RS message and its parameters	349
8	Table 19-27: Definition of the parameters for Ranging_Capability_RS	350
9	Table 19-28: Ranging_Session_RQ message and its parameters	350
10	Table 19-29: Definition of the parameters for Ranging_Session_RQ	350
11	Table 19-30: Ranging_Session_RS message and its parameters.....	351
12	Table 19-31: Definition of the parameters for Ranging_Session_RS.....	351
13	Table 19-32: Ranging_Session_Setup_RQ message and its parameters	352
14	Table 19-33: Definition of the parameters for Ranging_Session_Setup_RQ	353
15	Table 19-34: Ranging_Session_Setup_RS message and its parameters	354
16	Table 19-35: Definition of the parameters for Ranging_Session_Setup_RS.....	354
17	Table 19-36: Ranging_Suspend_RQ message and its parameter	354
18	Table 19-37: Definition of the parameter for Ranging_Suspend_RQ	354
19	Table 19-38: Ranging_Suspend_RS message and its parameter	354
20	Table 19-39: Definition of the parameter for Ranging_Suspend_RS	355
21	Table 19-40: Ranging_Recovery_RQ message and its parameter.....	355
22	Table 19-41: Definition of the parameter for Ranging_Recovery_RQ.....	355
23	Table 19-42:Ranging_Recovery_RS message and its parameter	355
24	Table 19-43: Definition of the parameter for Ranging_Recovery_RS	355
25	Table 19-44: Configurable_Ranging_Recovery_RQ message and its parameter.....	356
26	Table 19-45: Definition of the parameter for Configurable_Ranging_Recovery_RQ.....	356
27	Table 19-46: Configurable_Ranging_Recovery_Response message and its parameter	356
28	Table 19-47: Definition of the parameter for Configurable_Ranging_Recovery_Response	356
29	Table 19-48: DK_APDU_RQ message and its parameter.....	357
30	Table 19-49: Definition of the parameter for DK_APDU_RQ.....	357
31	Table 19-50: DK_APDU_RS message and its parameter	357
32	Table 19-51: Definition of the parameter for DK_APDU_RS	357
33	Table 19-52: Time_Sync message and its parameters	357
34	Table 19-53: Definition of the parameter for Time_Sync	358
35	Table 19-54: First_Approach_RQ message and its parameters.....	359
36	Table 19-55: Definition of the parameter for First_Approach_RQ	359
37	Table 19-56: First_Approach_RS message and its parameters.....	360
38	Table 19-57: Definition of the parameter for First_Approach_RS	360
39	Table 19-58: RKE_Auth_RQ message and its parameter	360
40	Table 19-59: Definition of the parameter for RKE_Auth_RQ	360
41	Table 19-60: RKE_Auth_RS message and its parameter	360
42	Table 19-61: Pass_through message and its parameter	361
43	Table 19-62: Definition of the parameter for Pass_through	361
44	Table 19-63: HU_PP message and its parameters	361

1	Table 19-64: Definition of the parameter for HU_PP.....	361
2	Table 19-65: HUP_RQ message and its parameters.....	362
3	Table 19-66: Definition of the parameter for HUP_RQ.....	362
4	Table 19-67: HUP_RS message and its parameters.	363
5	Table 19-68: Definition of the parameter for HUP_RS.....	363
6	Table 19-69: DK Event Notification message and its parameters.	363
7	Table 19-70: Definition of the parameters for DK Event Notification.	363
8	Table 19-71: DK SubEvents Category and its parameters.	364
9	Table 19-72: Definition of Command_Status and its Command_Status codes.....	364
10	Table 19-73: List of Command_Status for Command Complete SubEvent.....	364
11	Table 19-74: Definition of Session_Status and its Session_Status codes.....	366
12	Table 19-75: List of Session_Status for Ranging Session Status Changed SubEvent.....	366
13	Table 19-76: Definition of DR_Intent and DR_Intent codes.....	366
14	Table 19-77: List of DR_Intent for Device Ranging Intent SubEvent.	366
15	Table 19-78: List of Vehicle Status Changed SubEvent Tags.....	368
16	Table 19-79: Definition of Function ids and their corresponding Action ids.	370
17	Table 19-80: Definition of Standardized Function/Execution Status Values to Indicate the (Temporary) Unavailability of Function/Execution or Errors.	373
19	Table 19-81: Definition of RKE Request SubEvent Templates.	374
20	Table 19-82: Definition of Headunit_Pairing_Status and its Session_Status.	375
21	Table 19-83: List of Session_Status for Head Unit SubEvent.	375
22	Table 19-84: 7F49 Template.	391
23	Table 19-85: URSK storage requirements per Digital Key endpoint.	399
24	Table 19-86: RKE allowed user authentication configuration.....	411
25	Table 20-1: Example MAC Configuration.	432
26	Table 20-2: Mapping of Slots to UWB Ranging Packets.....	436
27	Table 20-3: Pre-Poll Request Message and its parameters.	437
28	Table 20-4: The definition of parameters in the Pre-Poll Message.	437
29	Table 20-5: Final_Data Message and its parameters.	438
30	Table 20-6: The definition of parameters in the Final_Data Message.	439
31	Table 20-7: Ranging Status of a Responder.	440
32	Table 20-8: The Contents of the MHR Field.	444
33	Table 20-9: The Contents of the Frame Control Field.	444
34	Table 20-10: The Contents of the Auxiliary Security Header.	445
35	Table 20-11: The Contents of the Vendor Specific Header IE.	446
36	Table 21-1: Supported UWB Configurations.	451
37	Table 21-2: The Mandatory Length-127 Ternary Code Sequences.	451
38	Table 21-3: Summary of pulse shapes.	454
39	Table 21-4: Overview of PulseShape_Combo values and the associated transmit pulse shapes.	454
40	Table A-1: Issuer and Verifier Rules Common to External CA, Instance CA and Endpoint	477
41	Table A-2: Issuer and Verifier Rules For Extensions Common to External CA, Instance CA and Endpoint	478
42	Table A-3: Issuer and Verifier Rules For Extensions Specific to External CA	478
43	Table A-4: Issuer and Verifier Rules For Extensions Specific to Instance CA	478
44	Table A-5: Issuer and Verifier Rules For Extensions Specific to Endpoint	479

1	Table A-6: Issuer and Verifier Rules for Vehicle OEM Root CA certificate	479
2	Table A-7: Issuer and Verifier Rules for Vehicle Public Key Certificate	479
3	Table E-1: ENCAPSULATION_CMD format.....	501
4	Table E-2: Format of the P1 field	501
5	Table E-3: ENCAPSULATION_RSP format.....	501
6	Table E-4: Format of the P1 field	502
7	Table E-5: SENSF_RES format	503
8	Table E-6: Format of RWTI	504
9	Table G-1: Number of Valid Ranging Rounds per 96 ms.	506
10		
11		

1 LIST OF LISTINGS

2	Listing 5-1: Vehicle Certificate Extension Schema.....	84
3	Listing 5-2: Vehicle Certificate Extension Data.....	85
4	Listing 5-3: Vehicle Public Key Certificate Data	85
5	Listing 11-3: Sharing Invitation.....	146
6	Listing 11-4: Key Creation Processing	148
7	Listing 11-6: Import Request.....	151
8	Listing 11-7: Friend, Endpoint Data Import Processing	153
9	Listing 11-8: First Friend Transaction	154
10	Listing 11-9: Key Tracking Receipt	157
11	Listing 11-10: setPrivateMailboxBit	158
12	Listing 11-11: clearPrivateMailboxBit	159
13	Listing 11-12: clearPrivateMailboxBytes	159
14	Listing 11-13:Generate Attestation Signature	159
15	Listing 11-14: Generate Sharing Password	159
16	Listing 14-1: Vehicle OEM Privacy Encryption Certificate Extension Schema	192
17	Listing 14-2: Vehicle OEM Privacy Encryption Certificate Extension Data	192
18	Listing 14-3: Vehicle OEM Signature Certificate Extension Schema	192
19	Listing 14-4: Vehicle OEM Signature Certificate Extension Data	192
20	Listing 14-5: OEM App Data Attestation Processing.....	196
21	Listing 15-1: INSTALL for INSTALL Processing	206
22	Listing 15-2: SELECT Processing	211
23	Listing 15-3: Endpoint Certificate Extension Schema.....	215
24	Listing 15-4: Endpoint Certificate Extension Data.....	216
25	Listing 15-5: Endpoint Certificate Data.....	217
26	Listing 15-6: CREATE ENDPOINT Processing	218
27	Listing 15-7: TERMINATE ENDPOINT Processing for Command Format 1	220
28	Listing 15-8: TERMINATE ENDPOINT Processing for Command Format 2	220
29	Listing 15-9: DELETE ENDPOINT Processing	221
30	Listing 15-10: External CA Certificate Extension Schema	222
31	Listing 15-11: External CA Certificate Extension Data	222
32	Listing 15-12: External CA Certificate Basic Constraints Extension Data	222
33	Listing 15-13: External CA Certificate Data	222
34	Listing 15-14: Instance CA Certificate Extension Schema	224
35	Listing 15-15: Instance CA Certificate Extension Data.....	224
36	Listing 15-16: Instance CA Certificate Data	224
37	Listing 15-17: AUTHORIZE ENDPOINT Processing.....	227
38	Listing 15-18: VIEW Processing.....	229
39	Listing 15-19: AUTH0 Processing	231
40	Listing 15-20: AUTH1 Processing	233
41	Listing 15-21: PRESENCE0 Processing	234
42	Listing 15-22: PRESENCE1 Processing	235

1	Listing 15-23: READ BUFFER Processing for Command Format 1	236
2	Listing 15-24: READ BUFFER Processing for Command Format 2	237
3	Listing 15-25: WRITE BUFFER Processing	237
4	Listing 15-26: EXCHANGE Processing	239
5	Listing 15-27: CONTROL FLOW Processing	242
6	Listing 15-28: CREATE ENCRYPTION KEY Processing	243
7	Listing 15-29: GET PRIVATE DATA Processing for Command Format 1	244
8	Listing 15-30: GET PRIVATE DATA Processing for Command Format 2	244
9	Listing 15-31: SET PRIVATE DATA Processing	245
10	Listing 15-32: SET CONFIDENTIAL DATA Processing	245
11	Listing 15-33: SETUP ENDPOINT Processing	247
12	Listing 15-34: SETUP INSTANCE Processing	248
13	Listing 15-35: SIGN Processing	250
14	Listing 15-36: onDeselect Event Processing	251
15	Listing 15-37: CREATE RANGING KEY processing	251
16	Listing 15-38: DELETE RANGING KEYS Processing	252
17	Listing 15-39: Generate Random	253
18	Listing 15-40: Generate Key Pair	253
19	Listing 15-41: Generate Identifier	253
20	Listing 15-42: Generate Attestation Signature	253
21	Listing 15-43: Verify Attestation Signature	253
22	Listing 15-44: Compute Shared Key with Diffie-Hellman	254
23	Listing 15-45: Key Derivation	254
24	Listing 15-46: Secure Channel Command Decryption and Authentication	254
25	Listing 15-47: Secure Channel Response Encryption and Authentication	255
26	Listing 15-48: Derive KEenc, KEmac	255
27	Listing 15-49: Confidential Mailbox Encryption and Authentication	255
28	Listing 15-50: Compute DeviceCryptogram	255
29	Listing 15-51: framework.createInstance Processing	257
30	Listing 15-52: framework.getInstance Processing	258
31	Listing 15-53: framework.view Processing	258
32	Listing 15-54: framework.deleteInstance Processing	258
33	Listing 15-55: instance.view Processing	259
34	Listing 15-56: instance.createEndpoint Processing	260
35	Listing 15-57: instance.deleteEndpoint Processing	260
36	Listing 15-58: instance.getEndpoint Processing	261
37	Listing 15-59: instance.setParameters Processing	261
38	Listing 15-60: endpoint.setParameters Processing	262
39	Listing 15-61: endpoint.getCertificate Processing	262
40	Listing 15-62: endpoint.terminate Processing	263
41	Listing 15-63: endpoint.authorize Processing	264
42	Listing 15-64: endpoint.createEncryptionKey Processing	264
43	Listing 15-65: endpoint.setConfidentialData Processing	265
44	Listing 15-66: endpoint.getPrivateData Processing	265

1	Listing 15-67: endpoint.setPrivateData Processing	265
2	Listing 15-68: endpoint.sign Processing	266
3	Listing 15-69: endpoint.auth0 Processing	266
4	Listing 15-70: endpoint.auth1 Processing	267
5	Listing 15-71: endpoint.exchange Processing	267
6	Listing 15-72: endpoint.createRangingKey processing	267
7	Listing 17-1: Content of versionUpdate API on Vehicle Software Update	295
8	Listing 17-2: Content of versionUpdate API after Device Software Update	296
9	Listing 18-1: Server Password Generation	324
10	Listing 18-2: Vehicle-side Public Point Generation	324
11	Listing 18-3: Device-side Public Point Generation	324
12	Listing 18-4: Vehicle-side Computation of Shared Secret	325
13	Listing 18-5: Device-side Computation of Shared Secret	325
14	Listing 18-6: Derivation of Evidence Keys	325
15	Listing 18-7: Vehicle-side Computation of Evidence	325
16	Listing 18-8: Device-side Computation of Evidence	326
17	Listing 18-9: Derivation of System Keys	326
18	Listing 18-10: Secure Channel Command Encryption and Authentication	326
19	Listing 18-11: Secure Channel Response Encryption and Authentication	327
20	Listing A-1: Standard Transaction Cryptography Flow	465
21	Listing A-2: Fast Transaction Cryptography Flow	468
22	Listing A-3: Standard Transaction Cryptography Flow (with Reader Intent for Fast Transaction)	470
23	Listing A-4: X.509 v3 schema	474
24	Listing A-5: X.509 Key Usage extension	475
25	Listing A-6: X.509 Basic Constraints extension	476
26	Listing A-7: X.509 Authority Key Identifier extension	476
27	Listing A-8: X.509 Subject Key Identifier extension	476
28	Listing A-9: X.509 ECDSA Signature Format	476
29		
30		

1 ABBREVIATIONS AND ACRONYMS

2	AES	Advanced Encryption Standard
3	AGC	Automatic Gain Control
4	AID	Application Identifier
5	APDU	Application Protocol Data Unit
6	Bluetooth LE	Bluetooth Low Energy
7	BPM	Burst Position Modulation
8	BPSK	Binary Phase Shift Keying
9	CA	Certificate Authority
10	CASD	Controlling Authority Security Domain
11	CMAC	Cipher-based Message Authentication Code
12	CoC	Connection Oriented Channel over Bluetooth LE
13	DK	Digital Key
14	DRBG	Deterministic Random Bit Generator
15	dUDSK	Derived UWB Data Secret Key
16	dURSK	Derived URSK
17	ECC	Elliptic Curve Cryptography
18	ECDSA	Elliptic Curve Digital Signature Algorithm
19	ECIES	Elliptic Curve Integrated Encryption Scheme
20	ERDEV	Enhanced Ranging Device
21	GP	GlobalPlatform
22	HCE	Host Card Emulation; the NFC communication is routed to the framework instead of the SE
23		
24	HCI	Host Controller Interface
25	HRP	High Rate Pulse Repetition Frequency
26	HTTPS	Hypertext Transfer Protocol Secure
27	ID&V	Identification and Verification (of User Identity)
28	KDF	Key Derivation Function
29	KML	Key Management Logic
30	KTS	Key Tracking Server, operated by Vehicle OEM, privacy protected
31	L2CAP	Logical Link Control and Adaptation Protocol
32	Lc	Length Command
33	Le	Length Expected
34	LL	Link Layer
35	LSB	Least Significant Byte
36	MAC	Message Authentication Code

1	MFR	MAC Footer
2	MHR	MAC Header
3	MIC	Message Integrity Check
4	MITM	Man-in-the-Middle
5	MSB	Most Significant Byte
6	MSD	Message Sequence Diagram
7	mUPSK	Master UWB Privacy Secret Key
8	mURSK	Master URSK
9	NFC	Near Field Communication
10	NVM	Non-Volatile Memory
11	O2M	One-to-Many
12	OCSP	Online Certificate Status Protocol
13	OEM	Original Equipment Manufacturer
14	OOB	Out-of-Band
15	PAKE	Password Authenticated Key Exchange
16	PHR	Physical Header
17	PHY	Physical Layer
18	PRF	Pulse Repetition Frequency
19	PSDU	Physical Layer Service Data Unit
20	PSM	Protocol/Service Multiplexer
21	QoS	Quality of Service
22	RAN	Ranging Area Network
23	RDEV	Ranging-capable device
24	RFU	Reserved for Future Use
25	RKE	Remote Keyless Entry
26	RS	Reed Solomon
27	RTR	Remote Termination Request
28	TA	Termination Attestation
29	SCA	Side Channel Attack
30	SDS-TWR	Symmetric Double-Sided Two-Way Ranging
31	SE	Secure Element
32	SECDED	Single Error Correct Dual Error Detect
33	SFD	Start of Frame Delimiter
34	SHA	Secure Hash Algorithm
35	SHR	Synchronization Header
36	SP0	STS Packets type 0 (packets with payload and no STS)
37	SP3	STS Packets type 3 (packets without PHR, MHR, or payload)
38	SPAKE2+	Simple Password Authenticated Key Exchange

1	SPSM	Simplified Protocol/Service Multiplexer
2	STS	Scrambled Timestamp Sequence
3	SW	Status Word
4	SYNC	Synchronization Header
5	TLV	Tag Length Value
6	ToF	Time of Flight
7	TTL	Time to Live
8	URL	Universal Resource Locator
9	URSK	UWB Ranging Secret Key
10	UWB	Ultra Wide Band

1 **DEFINITIONS**

2	Bluetooth Module	An entity managing the Bluetooth communication between device and vehicle as described in this specification. This entity may consist of one or more integrated circuits or may be part of a combo chip solution
5	Central	The master of the Bluetooth LE connection
6	Digital Key	Digital credentials used to authenticate access to the vehicle
7	Endpoint	Digital Key object in the applet
8	Initiator	An entity that starts the UWB ranging packet exchange by sending a first UWB POLL packet
10	Owner Pairing	Pairing of an Owner Device with a Vehicle
11	Peripheral	The slave in the Bluetooth LE connection.
12	Responder	An entity that responds to a UWB POLL packet
13	Shared Key	Digital Key shared with a friend device. It is used interchangeably with friend Digital Key
15	Slot identifier	Value stored in private mailbox to reference a key slot of a Digital Key
16	Vehicle	A vehicle implementing the Digital Key service
17	UWB Module	An entity managing the UWB ranging session as described in this specification. This entity may consist of one or more integrated circuits or may be part of a combo chip solution
20		

1 NOTATIONS

- 2 Values like A1_h or FEED_h are hexadecimal values using the Most Significant Byte first
3 convention (big-endian).
- 4 Values like 1101_b are binary values using Most Significant Bit first convention.
- 5 Command and response APDU bytes and buffers are implicitly represented in hexadecimal
6 notation.
- 7 Other numerical values like 1234 are decimal representations.
- 8 Fields in parentheses (..) are optional or conditional.
- 9 The concatenation operation is represented by ||.
- 10 ASCII values are represented by “quotes.”
- 11 String values shall be UTF-8 encoded unless specified differently.
- 12 Strings containing URLs/URIs shall be encoded as per RFC 3986 [27] unless specified
13 differently.
- 14 Strings containing date values shall follow ISO-8601[2] definition using date format yyyy-MM-
15 dd'T'HH:mm:ss.SSSZ unless specified differently
- 16 Binary data in JSON shall be encoded in hex string format (e.g., “deadbeef”) unless specified
17 differently.
- 18 In fields described as a sequence of bits, bit7 is the most significant bit, bit0 is the least
19 significant bit.
- 20 The following table shows a list of the parameters/variables used in this specification for the
21 purpose of the UWB MAC layer definition.
- 22

Parameter/Variable	Definition	Context
i	Ranging Block index	Ranging session
s	Ranging Round index	Ranging session
m	Slot index	Ranging session
k	Ranging Session index	Ranging session
N_{Round}^k	Number of ranging rounds in a ranging block	Ranging session
$\text{Round_Idx}^k(i)$	Ranging round index in ranging block i in the k -th ranging session that is used for the ranging exchange.	Ranging session
$N_{\text{Responder}}^k$	Number of responders	Ranging session
l	Index of responder	Ranging session

Parameter/Variable	Definition	Context
R_l	Responder with index $l=0,1,\dots, N_{\text{Responder}}^k - 1$	Ranging session
UWB_{time0}	Initiator time reference	RAN Global
UWB_{time0}^k	Initiator time reference for the k -th ranging session (by default, this defines the beginning of ranging block 1)	Ranging session
$UWB_{time0}^k(i)$	Initiator time reference for ranging block i for the k -th ranging session	Ranging session
$UWB_{time0}^k(i, Round_Idx^k(i))$	Initiator time reference for ranging exchanges in ranging block i , round $Round_Idx^k(i)$ for the k -th ranging session	Ranging session
STS_Index0^k	First STS index of the k -th ranging session, as negotiated during ranging session setup	Ranging session
$STS_Index^k(i)$	First STS index of ranging block i for the k -th ranging session	Ranging session
$STS_Index^k(i,s)$	First STS index of ranging round s of ranging block i for the k -th ranging session	Ranging session
$STS_Index^k(i,s,TYPE)$	STS index of the slot whose type is $TYPE$ in ranging round s of ranging block i for the k -th ranging session	Ranging session
T_{Round}^k	Ranging round duration	Ranging session
T_{Block}^k	Ranging block duration	Ranging session
$N_{\text{Slot_per_Round}}^k$	Number of slots in a ranging round	Ranging session

1
2
3

1 **CODE LISTINGS**

2 The pseudo-code listings presented in this document highlight the important processing steps
3 from a functional perspective. These pseudo-code listings do not represent any specific
4 implementation, but actual implementations shall be functionally equivalent. Operations like
5 basic input validation, length checking, input checking, option checking, or security counter-
6 measures may not be described. Memory or speed optimizations are not described in these
7 listings unless explicitly mentioned.

1 1 INTRODUCTION AND SCOPE

2 The Digital Key Technical Specification Release 3 specifies a Digital Key ecosystem using a
3 standardized Digital Key applet, standardized vehicle access protocol, and a scalable architecture
4 to support wide-scale deployment of the Digital Key services across different Vehicle OEMs and
5 Device OEMs. Release 3 is backward compatible with Release 2. Release 3 is not backward
6 compatible with Release 1. Release 1 can be deployed independently of Release 2 or 3.

7 The Digital Key Technical Specification Release 3 is based on the use of BLE/UWB or NFC as
8 an underlying radio technology to enable Digital Key services. The Digital Key management
9 framework is designed to be radio technologies agnostic, enabling the framework to be extended
10 to support other technologies.

11

12

1 2 SYSTEM ARCHITECTURE

2 With the exception of Section 2.10, which is normative, this is an informational section, which
3 provides an overview of the features, components, architecture, and operations of the Digital Key
4 system.

5 2.1 Overview

6 The proposed system uses asymmetric cryptography to mutually authenticate vehicle and device.
7 The device reveals its identity only to known vehicles.

8 Public keys are mutually exchanged through pairing of the owner device to the vehicle. The
9 owner can then authorize the use of Digital Keys by friends and family members by signing their
10 public keys.

11 The system is designed to work fully offline (i.e., no server connection is needed for a vehicle or
12 a device) for all relevant features at the time of execution, such as the owner pairing or
13 lock/unlock vehicle using the Digital Key. Where regulatory or business constraints require
14 online connections, these can be added to the system.

15 2.2 High-Level Features

16 The described system complies with the following high-level requirements:

- 17 • Security and privacy equivalent to, or better than, physical keys
- 18 • Pairing of owner device and vehicle
- 19 • Key sharing between owner and friends
- 20 • Interoperability across devices and Vehicle OEMs
- 21 • Support of multiple Digital Keys from different Vehicle OEMs on a single device
- 22 • Enabling of Vehicle OEM to control the issuance of Digital Key and its policy
- 23 • Privacy protection against active/passive eavesdroppers

24 2.3 High level architecture

25 The Digital Key ecosystem consists of multiple actors that are connected to each other using a
26 combination of standard and proprietary links as shown in [Figure 2-1](#). The standardized links are
27 fully specified in this specification to enable implementation and interoperability. The roles and
28 responsibilities of each entity and their relationships are described in Sections [2.4](#) and [2.5](#).

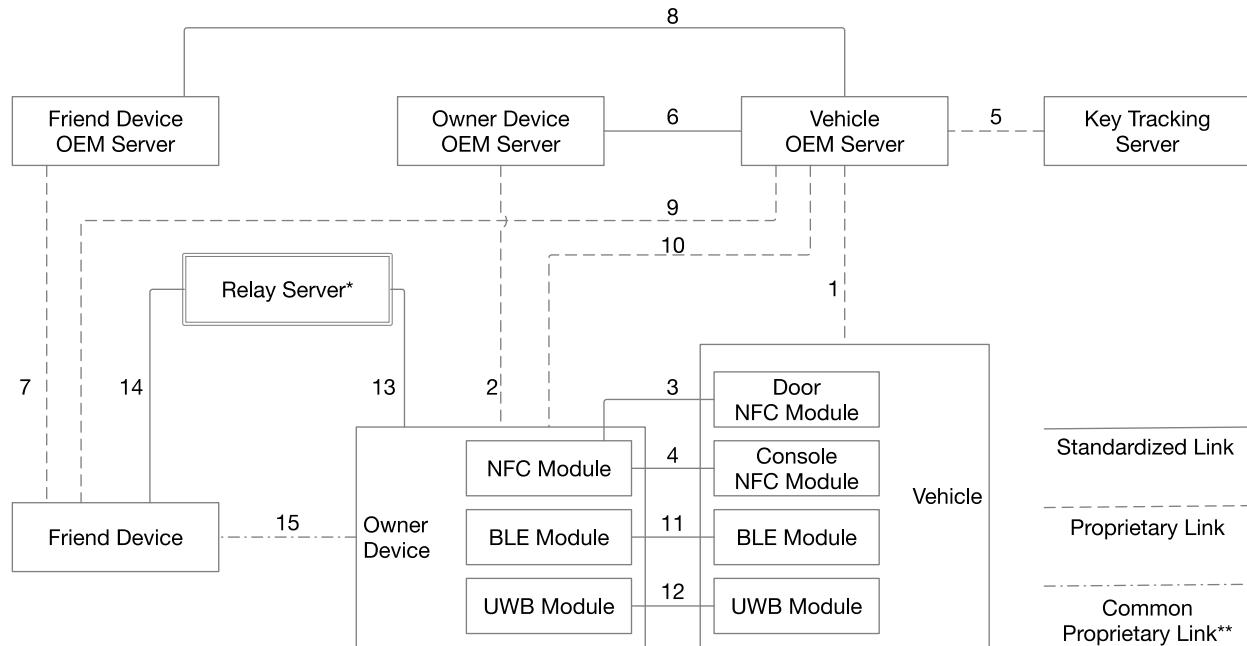
29 Note: The entities and the proprietary links (dashed lines in [Figure 2-1](#)) that are out of scope of
30 this specification are also described to provide an overview of the high-level functionalities of
31 the system.

32
33
34

1

2

Figure 2-1: Digital Key Architecture with Actors and Their Relationships



* Relay Server Selected by Owner Device OEM to Share keys with Friend Device from a different Device OEM.
Relay Server may be implemented by Device OEM, Vehicle OEM or by a Third Party

** Common Proprietary link is a link that both Owner and Friend device can use for communication.
E.g.: iMessage, WhatsApp or Other

3

4

- 5 In the system, the **vehicle** is linked to the **Vehicle OEM Server** per Telematics Link (1). This link provides a secure communication channel and is fully controlled by the Vehicle OEM.
- 6 The **vehicle** is equipped with **NFC Readers** and optional **Bluetooth/UWB modules** to communicate with the devices for owner pairing, vehicle locking/unlocking, and engine start via links (3), (4) in the case of NFC connectivity, or links (11), (12) in the case of BLE/UWB connectivity.
- 7 The **owner device** communicates with the **owner Device OEM Server** using a proprietary method (2). The **owner device** can also communicate directly with the **Vehicle OEM Server** using a proprietary Vehicle OEM app interface (10). Similarly, the **friend device** can communicate directly with the **Vehicle OEM Server** using a proprietary Vehicle OEM app interface (9).
- 8 The (owner/friend) **Device OEM Server** is responsible for managing the life cycle of the Digital Key applet and to update necessary certificates in the owner/friend device via (2)/ (7). It can provide services to suspend, restore, and wipe Digital Keys when a device is lost or stolen.
- 9 The **Vehicle OEM Server** hosts user accounts and manages ID&V. It also connects to an optional **Key Tracking Server** (5) to register all issued Digital Keys for a vehicle in such a way that privacy of the stored information is preserved.
- 10 The **owner Device OEM Server** is not directly connected to the other Device OEM Servers.

1 The **owner device** can share a Digital Key with the **friend device** (via (2), (6), (8) and (7)),
2 defining the appropriate access profile and can terminate the shared Digital Key when it is no
3 longer needed. Key sharing can be accomplished either via (2) and (7), (most commonly when
4 the Owner Device and Friend device are manufactured by the same OEM), or through a Relay
5 Server (most commonly when the Owner Device and Friend Device are manufactured by
6 different OEMs: via (13), (14)). The invitation to obtain a shared key from the relay server shall
7 be sent from the owner device to the friend device via (15). The invitation may use a second
8 factor authentication scheme where the friend proves its identity using schemes defined and
9 required by either Device OEMs (such as Device PIN) or Vehicle OEMs (such as sharing
10 password) as outlined in Section [11.2.2](#). The **friend device** can host Digital Keys shared by the
11 owner but cannot share the same Digital Key with any other device.
12 The **friend device** and its **friend Device OEM Server** connection (7) support necessary
13 certificate services, as in link (2). Device OEM or Vehicle OEM can facilitate device change.
14 However, this is out of scope.
15 The interface between the **Vehicle OEM Server** and (owner/friend) **Device OEM Server** (6)/
16 (8) is used to exchange the servers' certificates, key tracking, key termination, and notifications.
17 All eligible devices contain a certified SE as well as NFC capability to enable them to
18 communicate with the vehicle.

19 **2.4 Actors**

20 In this section, the roles and responsibilities of the entities involved in the Digital Key ecosystem
21 are described. The subsections within 2.4 describe the primary functions of the various actors.

22 **2.4.1 Vehicle**

- 23 • Determine if the owner/friend device is eligible for the Digital Key service before allowing
24 owner pairing or accepting a friend key shared by the owner device.
- 25 • If key tracking is required, may provide owner pairing information (owner public key, device
26 information, etc.) to the KTS or verify that owner pairing information was received by the
27 KTS.
- 28 • Verify authenticity of the device.
- 29 • Authorize any device that proves that it has a valid Digital Key to access the vehicle, and if
30 required by the vehicle, an immobilizer token to start the engine.
- 31 • If required, provide a user interface to delete the owner and friend Digital Keys.
- 32 • Provide secure processing and storage environment.

33 **2.4.2 Vehicle NFC Readers [WCC1]**

- 34 • Communicate with the owner device for owner pairing and Digital Key transactions
35 (lock/unlock, engine start, etc.).
- 36 • Communicate with the friend device for Digital Key transactions.

1 2.4.3 *Vehicle Bluetooth LE Module [WCC2/WCC3]*

- 2 • Communicate with the owner device for owner pairing and Digital Key transactions
3 (lock/unlock, engine start, RKE etc.).
4 • Communicate with the friend device for first friend transaction and Digital Key transactions
5 • Communicate with owner or friend device for setup of a secure ranging over UWB
6 • Communicate with owner or friend device for Remote transactions to allow the device to
7 initiate on-demand features (e.g. Lock/Unlock etc.)
8 • Communicate with owner or friend device for transmission of Notifications to notify and
9 signal change of state information
10 • Communicate with owner or friend device for transmission of data of 3rd party vehicle OEM
11 application.

12 2.4.4 *Vehicle UWB Module [WCC3]*

- 13 • Communicate with owner or friend device for secure ranging to securely determine the
14 distance between device and vehicle to support a secure distance measurement for passive
15 entry and passive engine start functionality.

16 2.4.5 *Vehicle OEM Server*

- 17 • Host owner account that links to the owner's vehicle(s).
18 • Manage Digital Key service subscriptions.
19 • Sign a shared Digital Key structure for acceptance by vehicle, assuring that business policies
20 are checked and Digital Keys are tracked.
21 • Provide necessary attestations to the vehicle (when online) so that shared friend Digital Keys
22 are accepted by the vehicle in the first friend transaction.
23 • Terminate Digital Keys in the vehicle when they have been deleted on a device.
24 • Sync with the owner device for termination of Digital Keys that is done offline.
25 • Manage a secure channel to the vehicle.
26 • Create pairing passwords and provide them to owner device and vehicle.
27 • Sign vehicle public key.
28 • Provide necessary certificates to Device OEMs.
29 • Provide Vehicle OEM and (optionally) Device OEM public keys to vehicle for owner pairing
30 and friend sharing.

31 2.4.6 *Key Tracking Server (KTS)*

- 32 • Record relevant data to be able to assign a tracked Digital Key for a vehicle to a device. The
33 KTS is likely to be managed by the Vehicle OEM.
34 The following are the key properties of the KTS:
35 ○ Only accessed for data query when required for legal or insurance reasons or for
36 transmission of friend key information during owner device change (see Section
37 [13.5.1.1](#))

- 1 ○ Data separation from the Vehicle OEM Server to fulfill privacy requirements of
2 tracked data

3 **2.4.7 Devices**

- 4 • Contain a secure processing and storage environment (SE or equivalent) running a Digital
5 Key applet
6 • Can take on the role of an owner device and friend device
7 • Support contactless transactions to lock/unlock vehicle and start the engine
8 • Support configurable user authentication (e.g., passcode)
9 • Check service eligibility before allowing owner pairing or acceptance of a friend Digital Key

10 **2.4.7.1 Owner Device**

- 11 • Implement main features: transaction, owner pairing, Digital Key sharing (sender), and
12 Digital Key termination
13 • Store necessary certificates for owner pairing and Digital Key sharing
14 • Terminate Shared Keys by sending termination request to vehicle (via Vehicle OEM Server)
15 and to friend device via Device OEM Server and Vehicle OEM Server
16 • Provide a common proprietary link (15) from [Figure 2-1](#) for Key Sharing

17 **2.4.7.2 Friend Device**

- 18 • Implement main features: transaction, Digital Key sharing (receiver), key termination
19 • Store necessary certificates for Digital Key sharing
20 • Send termination attestation to Vehicle OEM Server
21 • Provide a common proprietary link (15) from [Figure 2-1](#) to accept shared keys

22 **2.4.8 Device OEM Server**

- 23 • Load and install the Digital Key instance of the Digital Key applet (if necessary)
24 • Provide and update necessary certificates in the device
25 • Allow Digital Key functionality to be temporarily disabled on a lost or stolen device (if
26 online)
27 • Allow wipe of Digital Keys on a lost or stolen device (if online)

28 **2.4.9 Relay Server**

- 29 • Provides a standardized communication channel to support key sharing between two
30 devices from different (or the same) OEMs. The standardized communication channel is
31 outlined in the Secure Credential Transfer RFC [\[38\]](#).
32 • Implements push notifications and supports polling mechanisms to keep devices
33 informed during the key sharing process.
34 • May be implemented by any entity and must be included in the CCC-approved Relay
35 Server providers listed in [\[35\]](#)

2.5 Relationships

In this section, the relationships between different entities are described. Links referred to in this section (in parentheses) are shown in [Figure 2-1](#).

The following links are standardized according to the protocol, and relevant message formats are defined in this specification:

- **Owner/Friend Device:** Door NFC reader (link 3) as defined in Section [2.5.1](#)
- **Owner/Friend Device:** Console NFC reader (link 4) as defined in Section [2.5.2](#)
- **Owner/Friend Device OEM Server:** Vehicle OEM Server (links 6 and 8) as defined in Section [2.5.10](#).
- **Owner/Friend Device:** Bluetooth LE Interface as defined in Section [2.5.3](#).
- **Owner/Friend Device:** UWB Interface as defined in Section [2.5.4](#).
- **Relay server:** Owner and friend device (links 13 and 14) as defined in Section [11.3](#)

All other links are out of scope of this specification and are described at a high level to provide an overview of the underlying links and their functionality. Standardized information may be transported across out-of-scope links.

2.5.1 Door NFC Reader (3) [WCC1]

- Conducts regular transactions (fast or standard) with all devices that have a valid and registered Digital Key
- Conducts first transaction for a friend device during Key Sharing, transmitting necessary attestations so that the vehicle can verify the Digital Key of the friend device
- Implements specific polling to allow automatic selection of the appropriate Digital Key

2.5.2 Console NFC Reader (4) [WCC1]

- Conducts (standard or fast) engine start transactions with all devices that have a valid and registered Digital Key
- Authorizes engine start
- Conducts owner pairing transaction with a device to enable it to become the owner device
- Conducts first transaction for a friend device in cases when the vehicle was offline during key sharing, transmitting necessary attestations so that the vehicle can verify the Digital Key of the friend device
- Implements specific polling to allow automatic selection of the appropriate Digital Key (see Section [2.10](#), NFC Interface)

2.5.3 Bluetooth LE Interface (11) [WCC2/WCC3]

- Conducts standard transactions with all devices that have a valid and registered Digital Key
- Conducts owner pairing transaction with a device to enable it to become the owner device
- Conducts first transaction for a friend device
- Conducts setup of the secure UWB ranging session

- 1 • Conducts Remote transactions to allow the device to initiate on-demand features (e.g.
2 Lock/Unlock etc.)
3 • Transmit Notifications to notify and signal change of state information
4 • Transmit data of 3rd party vehicle OEM application.

5 **2.5.4 UWB Interface (12) [WCC3]**

- 6 • Conducts secure ranging to securely determine the distance between device and vehicle to
7 support a secure distance measurement for passive entry and passive engine start
8 functionality.

9 **2.5.5 Owner-to-Friend Device Link (via (2), (6), (8), and (7))**

10 **2.5.5.1 Owner-to-Friend Device Link (via (2), (6), (8), and (7))**

- 11 • Owner device deletes Digital Keys from the friend device by sending termination and
12 deletion commands
13 • Friend device sends termination attestation when a Digital Key on the device is terminated

14 **2.5.5.2 Owner-to-Friend Device Link (via (13), (14) and (15))**

- 15 • Owner device shares Digital Keys with friend device using sequence of communications
16 links: (15), (13), (14)

17 **2.5.6 Owner or Friend Device to Vehicle OEM Server (10, 9)**

- 18 • Links 9 and 10 are realized through the Vehicle OEM app; see Section [2.6.7](#)

19 **2.5.7 Telematics Link (1)**

- 20 • Vehicle OEM managed, proprietary, trusted, and confidentiality-preserving link. Key
21 functions include:
22 ○ Send owner pairing verifier and additional pairing information to the vehicle
23 ○ Obtain signature of the vehicle public key from the Vehicle OEM CA
24 ○ Send Digital Key termination information to delete a specific Digital Key in the vehicle
25 ○ Notify Vehicle OEM Server about Digital Key deletion in vehicle to remove Digital Key
26 from the friend device
27 ○ Register owner Digital Keys with Vehicle OEM Server (KTS)

28 **2.5.8 Owner/Friend Device OEM Server Link (2,7)**

- 29 • Owner/Friend Device authenticates Device OEM Server in a device-specific way
30 • Device loads/install Digital Key applet (if not done in factory)

31 **2.5.9 Vehicle OEM Server to KTS (5)**

- 32 • Provide key tracking data to KTS, such as
33 ○ Public key of owner and friend devices

- 1 ○ Instance CA identifier of hosting SE (owner or friend)
- 2 ○ Anonymized vehicle identifier

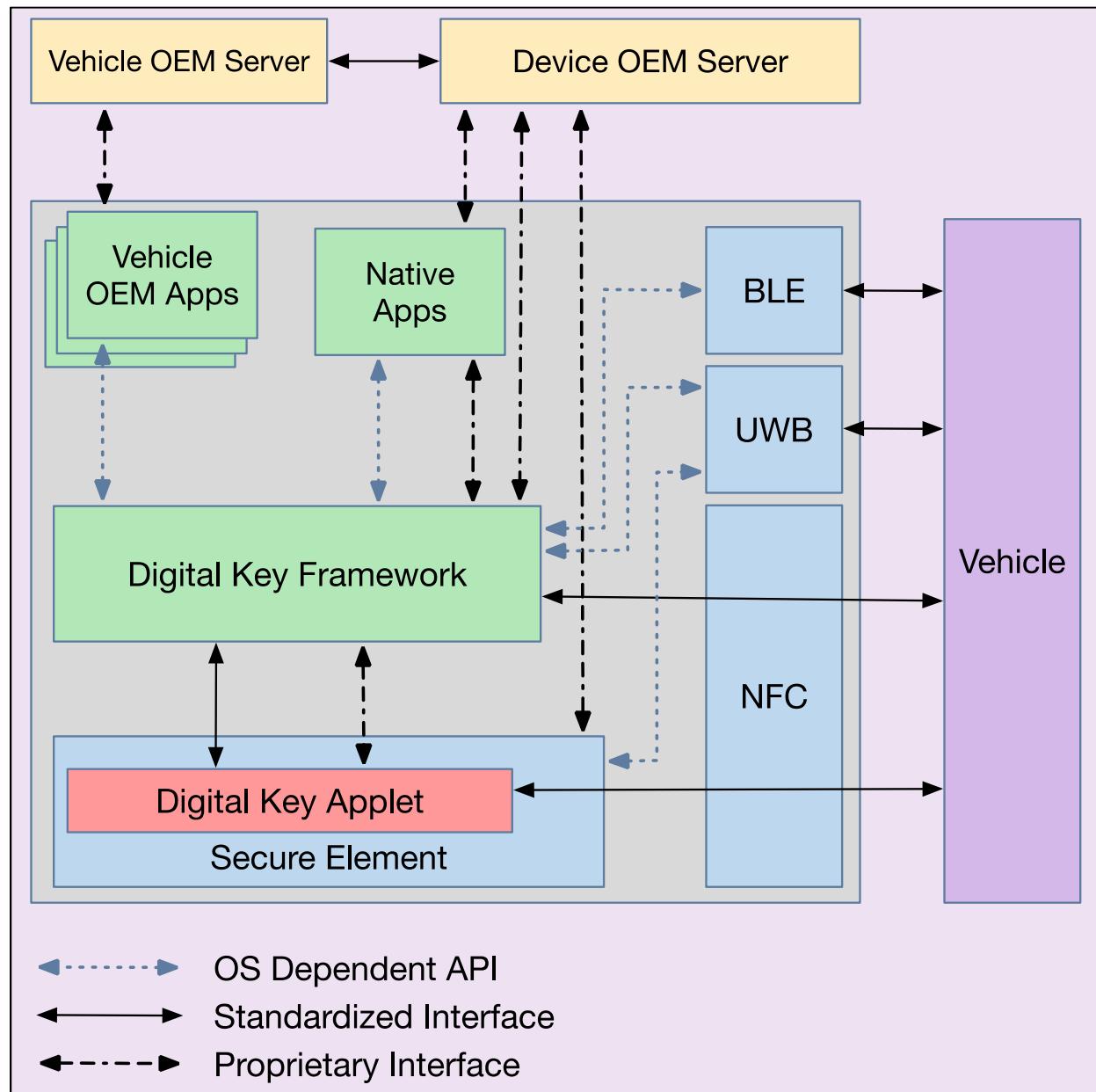
3 *2.5.10 Owner/Friend Device OEM Server to Vehicle OEM Server (6,8)*

- 4 • Establish trust relationship
- 5 • Exchange and sign the necessary certificates (see Section [16](#))
- 6 • Support Digital Key sharing
- 7 • Support Digital Key tracking
- 8 • Support Digital Key termination
- 9 • Support notifications

1 2.6 Device Structure

2 On the device side, the following functional elements participate in the system:

3 *Figure 2-2: Device Functional Elements.*



4
5 Secure Element and NFC controller are required components of eligible devices. The SE
6 provides the root of trust, which is the starting point of the trust chain (see Section [16](#)).

7 2.6.1 NFC Component [WCC1]

- 8
 - Card emulation mode required for contactless transactions
 - Host card emulation mode required for owner pairing

1 2.6.2 *Bluetooth Module [WCC2/WCC3]*

- 2 • Communicate with the vehicle for owner pairing, first friend transaction and Digital Key
3 transactions (lock/unlock, engine start, RKE etc.).
4 • Communicate with vehicle for setup of a secure ranging over UWB
5 • Communicate with vehicle for Remote transactions to allow the device to initiate on-
6 demand features (e.g. Lock/Unlock etc.)
7 • Communicate with vehicle for transmission of Notifications to notify and signal change
8 of state information
9 • Communicate with vehicle for transmission of data of 3rd party vehicle OEM
10 application.

11 2.6.3 *UWB Module [WCC3]*

- 12 • Communicate with vehicle for secure ranging to securely determine the distance between
13 device and vehicle to support a secure distance measurement for passive entry and
14 passive engine start functionality.

15 2.6.4 *Secure Element (or equivalent)*

- 16 • Java Card (example) and GlobalPlatform [\[20\]](#) support, SE Root based on GP-defined
17 Controlling Authority Security Domain available
18 • GlobalPlatform Card Specification Amendment C [\[20\]](#) and GlobalPlatform Contactless
19 Extension [\[21\]](#)
20 • Standard symmetric and asymmetric cryptographic support
21 • Hosts Digital Key applet
22 • Ability to distinguish communication between wired interface and contactless interface

23 2.6.5 *Digital Key Applet*

24 The Digital Key applet provides the following services:

- 25 • Hosts Digital Keys (one applet instance hosts Digital Keys of all Vehicle OEMs)
26 • Implements relevant (fast and standard) transactions
27 • Implements Instance CA (see Section [4.1](#) and Section [16.2.3](#)) to support offline use cases
28 and privacy protection
29 • Stores immobilizer tokens when required by the vehicle, offline attestations, access
30 profiles, and other data associated with a Digital Key
31 • Verifies authenticity of the vehicle
32 • Verifies certificate chain of friend public key

33 If the Digital Key applet is the SE-centric applet model as defined in Section [15.1](#), the applet also
34 provides the following service:

- 35 • Verifies the Vehicle Public Key Certificate [K]

1 2.6.6 *Digital Key Framework*

- 2 • Implements main features: owner pairing, Digital Key sharing, and management
3 • Provides common Digital Key service functionality via a set of OS-specific APIs for
4 Vehicle OEM apps. The APIs are described in documentation of the respective OS
5 platform developer. The functional requirements are defined in [Appendix F](#).

6 2.6.7 *Vehicle OEM App*

- 7 • The Vehicle OEM app is optional. The main features of the app are supported natively by
8 the device.
9 • May support the same features as the native app plus Vehicle OEM-specific features
10 • Provides ID&V with Vehicle OEM Server
11 • Retrieves owner pairing password
12 • Manages keys with non-standard access profiles

13 2.6.8 *Native App*

- 14 • Provides device-native UI such as Digital Key creation, Digital Key termination and
15 deletion, Digital Key enable/disable, etc.
16 • Displays a list of all issued owner/friend Digital Keys

17 **2.7 Vehicle States**

18 The vehicle has the following possible internal states:

- 19 • **Unpaired:** State in which no owner device is associated with the vehicle. This occurs
20 either at first purchase or when the owner chooses in the vehicle menu to unpair, which
21 leads to the deletion of all Digital Keys (owner and friend).
22 • **Paired:** State in which an owner device is associated with the vehicle. This state is
23 maintained when the owner deletes the associated owner device (e.g., to change the
24 owner device) without deleting the friend devices (see Section [13.5](#)).
25 • **Pairing:** The vehicle is in this state when waiting for an owner device to be associated.
26 • **Blocked:** The vehicle is in this state after a configurable number of failed pairing
27 attempts (Vehicle OEM policy)

28 **2.8 Digital Key User Classes**

29 This section describes the different roles associated with Digital Keys. A single device can hold
30 Digital Keys with different roles for multiple vehicles.

31 2.8.1 *Owner*

32 The vehicle accepts only one owner device. When an owner device is associated, the vehicle
33 switches from the unpaired to paired state. The owner has all access rights to the vehicle. The
34 owner's Digital Key may need to be registered in the KTS in order to be accepted by the vehicle.

1 **2.8.2 *Friend***

2 The vehicle accepts several friend devices. Friend devices may have restricted access rights to
3 the vehicle. These access rights are assigned by the owner using an access profile (see Section
4 [2.9.2](#)) when issuing the Digital Key and are checked by the vehicle and/or Vehicle OEM Server
5 according to the Vehicle OEM policy. The friend Digital Key may need to be registered in the
6 KTS in order to be accepted by the vehicle.

7 **2.9 Access Profiles**

8 For each device associated with the vehicle (owner device and one or more friend devices), the
9 vehicle stores the access profile and corresponding public key.

10 **2.9.1 Owner**

11 The owner Digital Key has all access rights with no restrictions.

12 **2.9.2 *Friend***

13 The owner may choose which profile to grant to the friend during key sharing.

14 The list of supported access profiles is defined in Section [11](#). As not all vehicles support all
15 profiles, the vehicle indicates to the owner device which profiles are supported during owner
16 pairing (see Section [6](#)).

17 **2.10 Versioning**

18 **2.10.1 General**

19 Versioning is required so that devices, vehicles, and servers with different Digital Key
20 specification releases, can still work together.

21 Version negotiation is used to agree on the highest version supported by all participants of a
22 specific feature or relationship.

23 **2.10.2 Domain Versions**

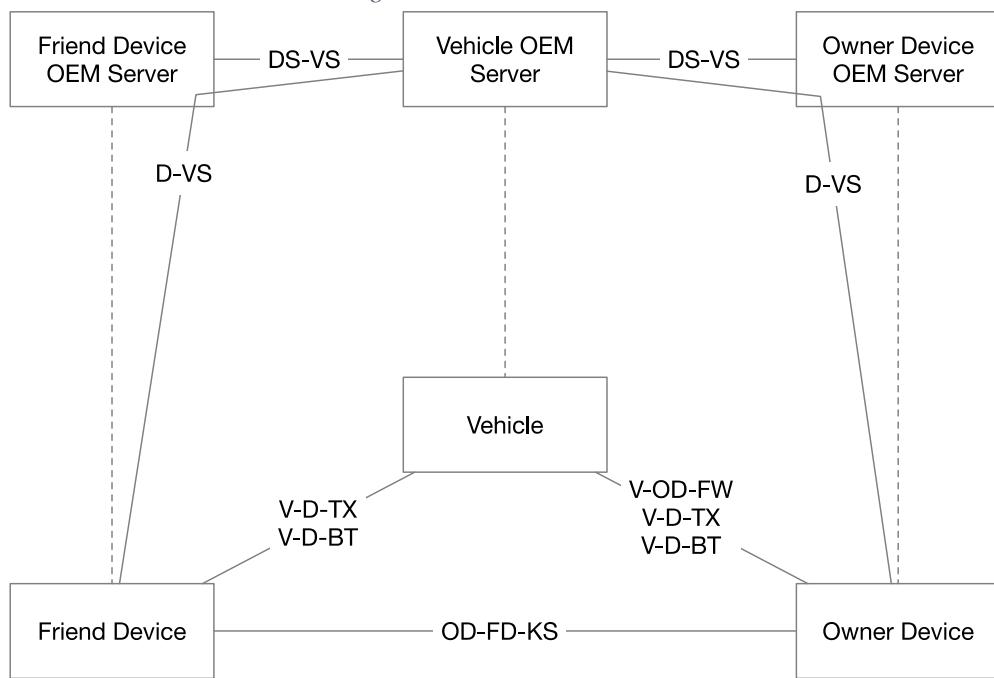
24 Figure 2-3 identifies the domain versions that are relevant for the system.

25 Domain versions describe logical relationships, not physical APIs, between actors in the system.
26 All APIs are associated to a particular domain version. Data structures might be associated with
27 one or multiple domain versions, depending on how many entities receive and process the data
28 structure. Data structures can contain elements that are determined by a different domain version
29 than the data structure layout itself. This specification documents those associations where
30 reasonable, so as to allow quick identification of the impact that an API or data structure change
31 has on the relevant domain versions. Some data structures do not list domain version
32 information. In case of a change in those data structures, the relevant domain version(s) need(s)
33 to be identified depending on the nature of the change.

- 1 The domain version determining the data structure is provided in the first line (e.g., highest level
2 TLV). Domain versions for data elements are provided with each element in the structure
3 description.
- 4 Note that the APIs and data structures do not need to (but can) carry explicit version information,
5 the knowledge of the API and data structure semantics and syntax are implicitly known by
6 sender and receiver based on the associated domain version.
- 7 Note also that the relationships described by dashed lines in Figure 2-3 are out of scope of this
8 specification and the representation of these relationships in any subsequent MSDs in this
9 specification is for illustrative purposes only.
- 10 Domain versions always include a list of versions from both sides, for example, D-VS-serverList
11 and D-VS-deviceList, or at least an agreed version from the responding side. When a commonly
12 supported version is found through the exchange of lists, then this is the “agreed version”. Where
13 relevant, this clarification is added throughout the specification.

14

Figure 2-3: Domain Versions



- 15
16 The vehicle-to-device (V-D) relationship is described by two domain versions.
17 1. (V-D-TX): Version of the transactions (fast, standard, etc.) between device and vehicle.
18 2. (V-D-BT): This domain version determines the Bluetooth (BT) version for Digital Key used
19 between device and vehicle.
20 The vehicle-to-owner-device (V-OD) relationship is described by one additional domain version:
21 1. (V-OD-FW): Version of the framework relationship established at owner pairing and features
22 based on this relationship.
23 The device-to-vehicle-server (D-VS) relationship and the device-server-to-vehicle-server
24 relationship do not require a differentiation between owner and friend device or owner and friend
25 device server. The versioning is managed independently of the key type.

1 The vehicle-side lists of V-D-TX, V-OD-FW and V-D-BT obtained by the owner are sent to the
2 friend device during key sharing. This allows Friend device to determine if there will be an
3 agreed V-D-TX and V-D-BT version to connect over Bluetooth LE and to transact with the
4 vehicle. This allows to determine whether a shared key can be accepted or shall be rejected.
5 The V-D-TX and V-D-BT finally used in transactions between the friend device and vehicle may
6 have different values than the V-D-TX and V-D-BT used in transactions between the owner
7 device and vehicle.
8 Owner and friend device exchange sharing data to create friend keys. This data is partially
9 generated and consumed by the framework and partially by the applet. Explicit applet domain
10 version agreement is not required as the frameworks of both devices represent the applet and
11 framework version in the key sharing domain version (OD-FD-KS).
12 Vehicle OEMs provide endpoints URLs to device OEMs to enable routing of device information
13 to the region and environment in which the vehicle is registered based on the
14 ROUTING_INFORMATION obtained during owner pairing and key sharing by the devices.
15 The vehicle OEM can define a D-VS and a DS-VS version per endpoint URL which allows
16 vehicle servers to run on different SW releases per endpoint. See examples in Section 2.10.4

17 **2.10.3 Domain Wise Version Agreement**

18 **2.10.3.1 V-OD-FW**

19 The framework version is agreed between vehicle and owner device during owner pairing based
20 on tags 5Ah and 5Bh in Table 5-3 and Table 5-4 respectively.

21 V-OD-FW-agreedVersion is provided during key tracking to the vehicle server (see Section
22 17.7.1.2)

23 When the device or vehicle SW is updated to support a new V-OD-FW version, then the updated
24 version list is provided via the versionUpdate API (see section 17.7.5.4) to the other side.

25 **2.10.3.2 V-D-TX**

26 The transaction version is agreed between vehicle and owner device during owner pairing based
27 on tags 5Ch (both sides) in Table 5-4.

28 The vehicle list V-D-TX-vehicleList is provided during key sharing in Tag 5Ch
29 (DIGITAL_KEY_APPLET_PROTOCOL VERSIONS) in Table 11-6. This allows the friend
30 device to verify version compatibility with the vehicle before accepting a shared key.

31 Independent of the V-D-TX version agreed during owner pairing or key acceptance, all devices
32 agree on a version during every fast and standard transaction. This allows updating vehicle or
33 device SW to support higher versions without the need to transfer this knowledge via servers to
34 the other side. See tags 5Ch in Table 15-29: AUTH0 Command Payload and Table 15-12:
35 SELECT Response

36

37 **2.10.3.3 V-D-BT**

38 The Bluetooth version is agreed between vehicle and owner device during owner pairing based
39 on tags 5Eh and 5Fh in Table 5-4 and Table 5-5 respectively.

1 The vehicle list V-D-BT-vehicleList is provided during key sharing in tag 5F_h in Table 11-6.
2 This allows the friend device to verify version compatibility with the vehicle before accepting or
3 rejecting a shared key.
4 Independent of the V-D-BT version agreed during owner pairing or key acceptance, all devices
5 agree on a version during every BT connection. This allows updating vehicle or device SW to
6 support higher versions without the need to transfer this knowledge via servers to the other side.
7 The negotiation of versions under the V-D-BT domain during owner pairing and passive entry is
8 described in detail in Section 19.2.1 and Section 19.2.3

9 **2.10.3.4 OD-FD-KS**

10 The key sharing version is agreed between the owner device and the friend device during key
11 sharing based on tags 54_h and 55_h in Table 11-6 and Table 11-7.

12 The key creation request (Table 11-2) cannot be versioned and must be managed by adding new
13 payloads with new tags if backwards-incompatible changes need to be done in the future.
14 Backwards-compatible changes can be done by adding new tags, as they shall be ignored by
15 earlier versions. It is strongly recommended that devices plan for enough buffer space to receive
16 larger key creation request messages than defined in this version of the specification.

17 **2.10.3.5 D-VS**

18 The vehicle server list D-VS-serverList is provided to the devices via a proprietary configuration
19 method of the device OEM.

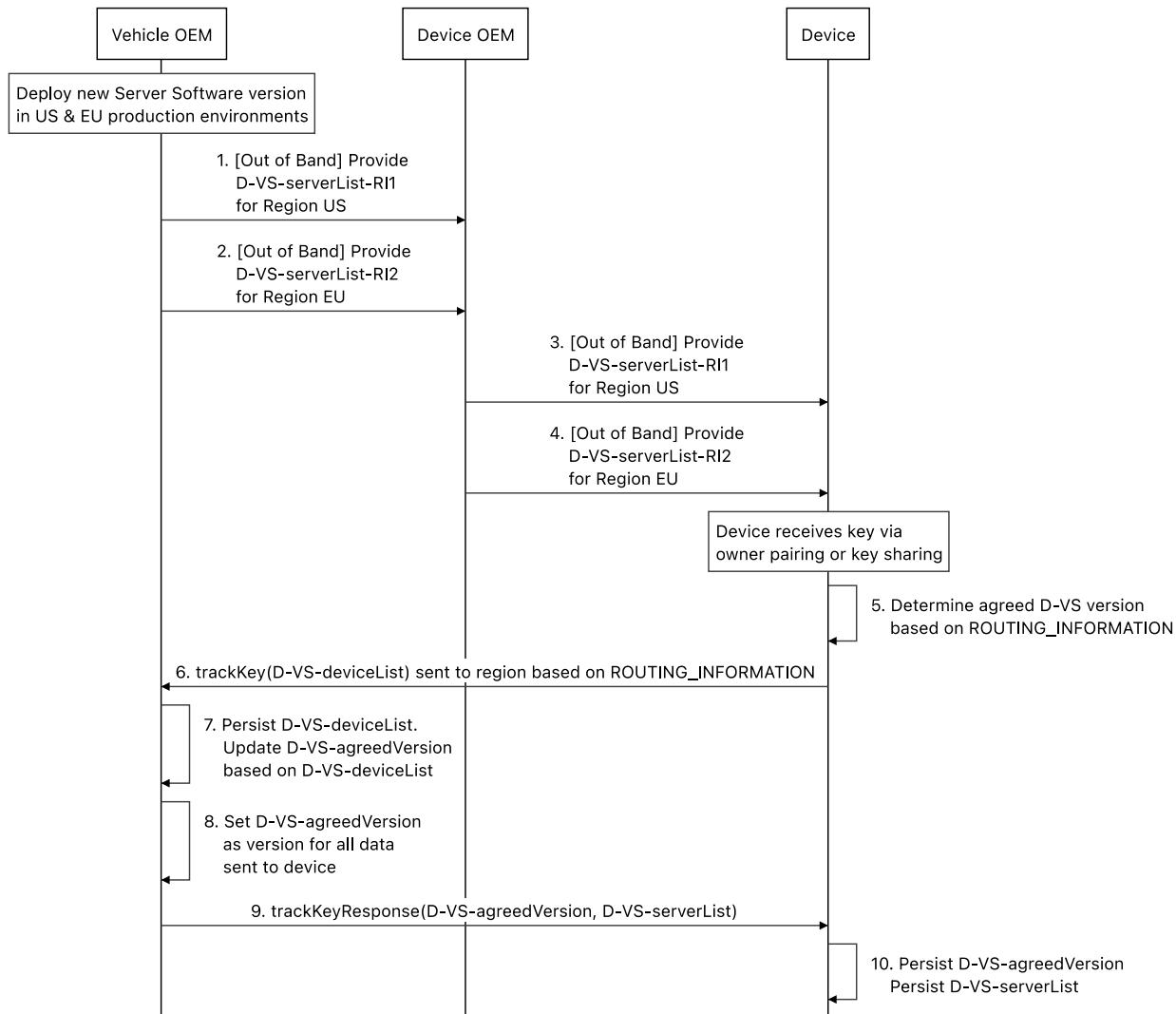
20 The device list D-VS-deviceList is provided during key tracking to the vehicle server (see
21 Section 17.7.1.2). The versionUpdate API is not required for this domain version.

22 One D-VS-serverList can be provided per endpoint URL by the Vehicle OEM to the Device
23 OEM, which corresponds to a specific value of the ROUTING_INFORMATION (RI) provided
24 by the vehicle at owner paring and by the owner device at key sharing.

25

1

Figure 2-4: D-VS Agreement for Key Tracking¹



2

3

4

- 5 [1 – 2] VS deploys new SW version for some regions. When deployed, the VS provides the new supported versions to connected DS in a proprietary way (based on the working terms agreed between VS and DS) for each server endpoint identified by the routing information (RI) as per above example (e.g., OEM1.USP.BRD1).
- 9 [3 – 4] DS provides the updated version lists in a proprietary way to all devices.

¹ Out-of-band in this figure refers to mechanisms that are outside the scope of the Digital Key Specification. E-mail or other mechanisms may be used for OEM-to-OEM communication compliant with pre-established agreements between device and vehicle OEMs; or Device OEMs may use proprietary APIs to communicate with devices similar to the telematics links deployed by Vehicle OEMs to communicate with Vehicles.

- 1 [5] ROUTING_INFORMATION is used by the device to determine the agreed D-VS
2 version to send the key tracking request based on e.g., D-VS-server-list-USP.
3 [6 – 7] The key tracking request is sent by the device in the newly determined D-VS
4 agreed version. The D-VS server list is determined using the routing information
5 provided during owner pairing. The key tracking request provides the device list
6 for the D-VS version to allow the server to determine the agreed D-VS version for
7 communication with this device (e.g., the key tracking response or a manage key
8 call).
9 Step 6: Although not shown in Figure 2-4, the device also provides the list of
10 supported V-OD-FW versions, including the agreed V-OD-FW version to the VS.
11 [8] From this moment on, (including the trackKeyResponse message) the server
12 sends all data in the new agreed D-VS version to this device. See Figure 2-6 as an
13 example.
14 [9 – 10] The key tracking response contains the server D-VS version list and the agreed
15 version, which is persisted in the device and should override the relevant
16 configuration that the device has obtained in steps 3 - 4.
17

18 2.10.3.6 DS-VS

19 The device server to vehicle server version is agreed between device OEM and vehicle OEM via
20 a proprietary configuration method. The versionUpdate API is not required for this domain
21 version.

22

23 2.10.4 Software Updates

24 A modification of the list of supported domain version values due to a SW update of one or more
25 entities shall lead to a version agreement update between all entities impacted by the version
26 change.

27 If required, each party needs to be able to handle temporarily different server versions on the
28 other side on a per region basis. Therefore, the version information may need to be connected to
29 a region-specific endpoint or server instance of the other OEM.

30 It is recommended that each OEM should always try rather to implement changes in a backwards
31 compatible way, especially for server-side updates. Non-backwards compatible changes in server
32 APIs should be avoided. The mechanism to inform OEM server or device counterparts is
33 described in Sections 2.10.3.5 and 2.10.3.6.

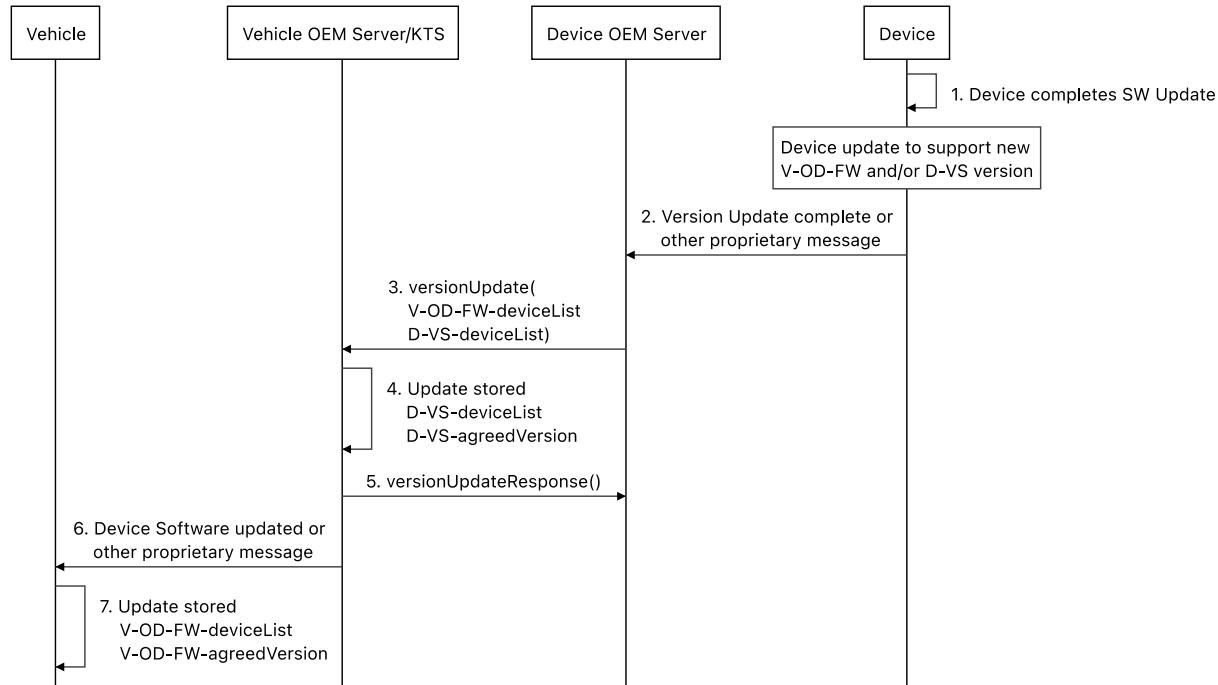
34

35 2.10.4.1 Device Software Update

36 Software updates to the owner or friend device that modify the list of supported V-OD-FW
37 and/or D-VS versions shall trigger the call of the versionUpdate() API to inform the vehicle and
38 vehicle OEM server about the updated device version lists. The call to the vehicle itself might be
39 asynchronous and/or the vehicle might be offline, so the response is not expected to contain

- 1 vehicle side version information and since the vehicle side version has not changed in this case it
2 does not need to be provided.

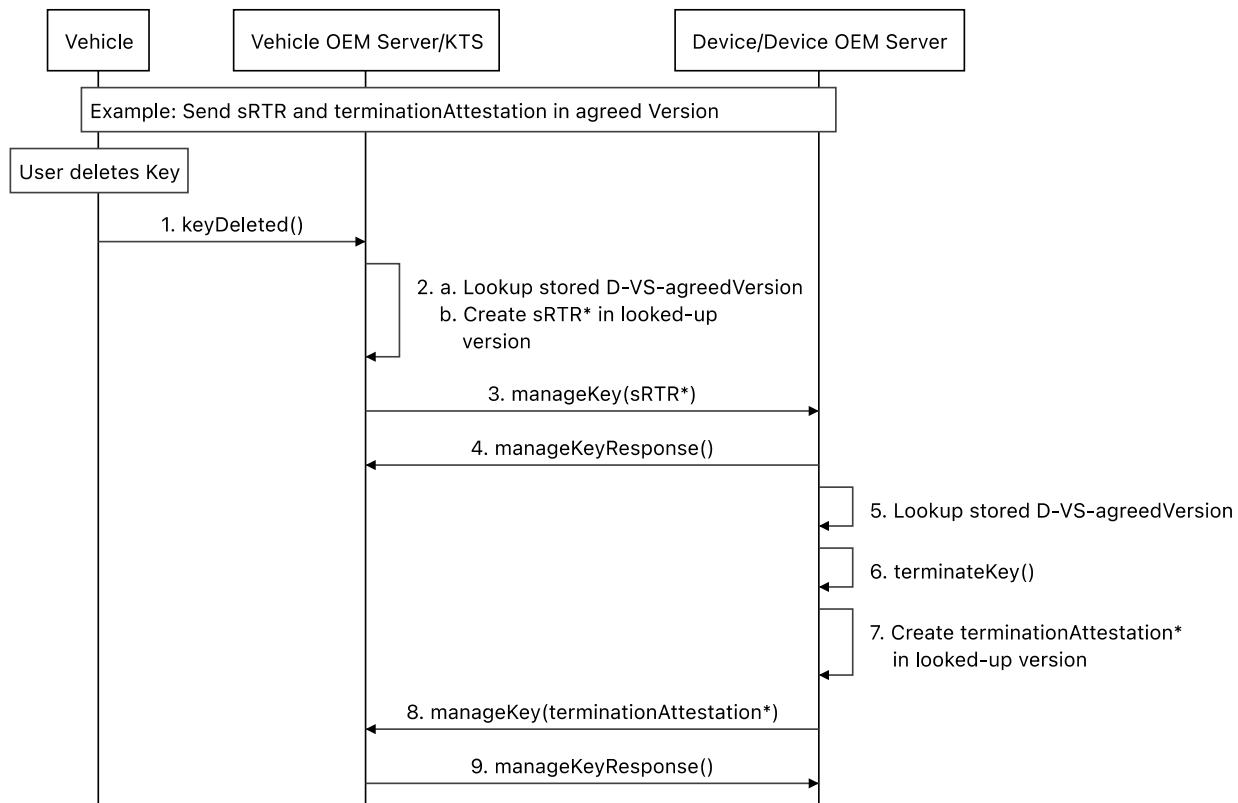
3 *Figure 2-5: Version Agreement after Device Software Update*



- 4
5 [1 - 2] Device provides updated device version lists to device OEM server. The link
6 between device and device OEM server is proprietary to the device OEM.
7 [3 – 5] Device OEM server calls versionUpdate API on vehicle OEM server, which
8 determines new agreed version for D-VS. This could result in a new agreed
9 version for V-OD-FW
10 [6 – 7] Vehicle OEM server provides new V-OD-FW device list to vehicle which
11 determines the new agreed version. The initiation of step [6] is asynchronous with
12 and not contingent on the completion of step [5]. The link between vehicle and
13 vehicle OEM server is proprietary to the vehicle OEM.

1

Figure 2-6: Example – manageKey() usage based on D-VS agreed version



2
3 Data elements exchanged between the server and the device, such as the server remote
4 termination request (sRTR, see Section 14.2) and terminationAttestation are versioned based on
5 the D-VS versions as shown in the example in Figure 2-6. In the example above, sRTR* and
6 terminationAttestation* are versions of the sRTR and the terminationAttestation that are
7 accepted by the device and the server in the agreed D-VS version.
8 Relationships between device and device OEM server as well as vehicle and vehicle OEM server
9 are proprietary, and their version management is out of scope.

10 **2.10.4.2 Device Server SW Update**

11 In case a device OEM server SW update that modifies DS-VS-serverList the device OEM may
12 notify the vehicle OEM using a proprietary mechanism that is deemed acceptable by both,
13 vehicle and device OEM. The endpoint of the device OEM server that received the update shall
14 increment or rollback the version that is included in the URL, depending on the applicable
15 specific region or instance of the vehicle OEM server and exact API call (e.g., .../2/manageKey).
16 The details of the URL scheme are provided in 17.4.

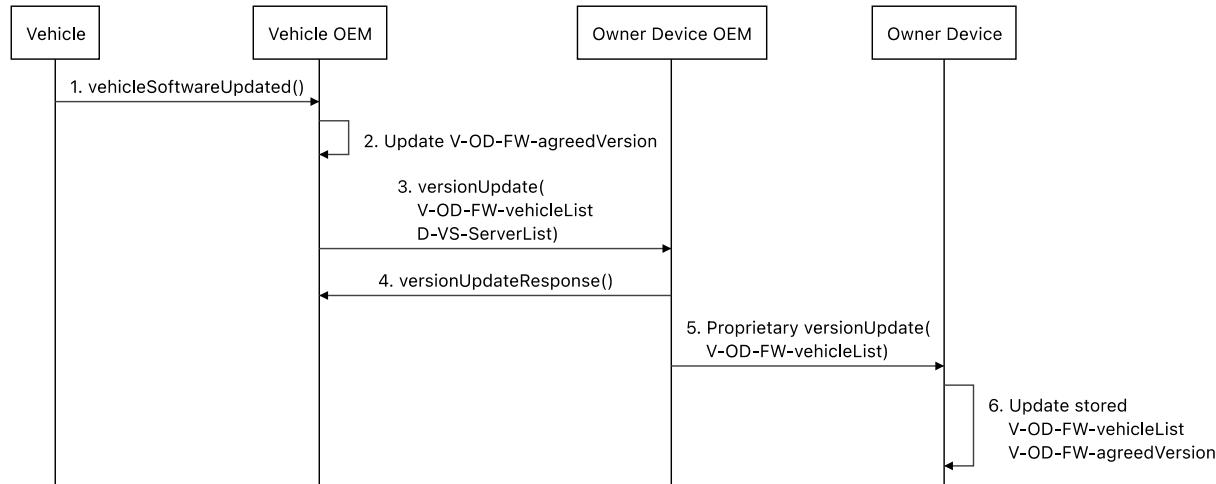
17

18 **2.10.4.3 Vehicle SW Update**

19 Software updates of the vehicle that modify the list of supported V-OD-FW shall trigger a call
20 of the versionUpdate() API to inform the owner device via the device OEM server about the new
21 vehicle version list (V-OD-FW-vehicleList).

1 Note: Updates that impact V-D-TX and V-D-BT are agreed during a Fast Transaction or
2 Standard Transaction with the vehicle.

3 *Figure 2-7: V-OD-FW Agreement after Vehicle SW Update*



- 4
5 [1] Vehicle SW update is confirmed to the server.
6 [2] Vehicle server updates agreed V-OD-FW version.
7 [3 – 4] Vehicle SW versions are sent to device server
8 [5 – 6] Device server provides new vehicle SW version lists to device using a proprietary
9 message.

10
11 Note that an update to a server version that impacts D-VS-serverList is configured in the devices
12 using a proprietary method, as described in Section 2.10.4.4.

13 *2.10.4.4 Vehicle Server SW Update*

14 In case a vehicle OEM server SW update that modifies DS-VS, the vehicle OEM notifies the
15 device OEM using a proprietary mechanism that is deemed acceptable by both device and
16 vehicle OEMs. The endpoint of the vehicle OEM server that received the update shall increment
17 or rollback the version that is included in the URL. This step is dependent on the region-specific
18 server instance, if any, and exact API call (e.g., .../v2/trackKey). The details of the URL scheme
19 are provided in Section 17.4.

20 In case of a vehicle OEM server SW update that modifies the D-VS-serverList, the vehicle OEM
21 notifies the device OEM in a proprietary way. The device OEM shall propagate this information
22 to relevant devices using proprietary mechanisms (as shown in Figure 2-4), so that devices are
23 made aware in advance of changes to the D-VS-serverList per region, environment, and brand for
24 every vehicle OEM.

25 The D-VS-serverList is provided in a proprietary way to every device OEM so that devices are
26 already aware of the D-VS-serverList per region, environment, and brand for every vehicle OEM.
27 Example:

28 OEM1.USP.BRD1: D-VS-serverList = v1.0, v2.0, v2.1; DS-VS = v1.0, v2.0, v2.2

29 OEM1.EUP.BRD1: D-VS-serverList = v1.0, v2.0, v2.1; DS-VS = v1.0, v2.0, v2.2

1 OEM1.CNP.BRD1: D-VS-serverList = v1.0, v2.0; DS-VS = v1.0, v2.0
2 Devices that have existing keys for this vehicle OEM re-calculate the agreed D-VS version for
3 each key immediately, whereby the re-calculation is based on the routing information obtained for
4 the corresponding vehicle (either at owner pairing or at friend sharing).
5 Device to vehicle server communication is then based on the new agreed version. A new key would
6 send the key tracking request based on the updated D-VS agreed version.

7

8 2.10.5 Version Numbers

9 Versions consist of a major and a minor version number (e.g., v2.3 with 2 = major and 3 =
10 minor).

11 Major versions are used for non-compatible versions of a data structure or API, or to add or
12 remove features, data structures and APIs.

13 Minor versions (with same major version) are compatible but limit the available feature set to the
14 one defined by the lower minor version.

15 To achieve forward compatibility, data elements in structures/commands/APDUs/API
16 parameters that have been added in higher versions should be ignored by the entity with the
17 lower version for TLV and JSON structures. (e.g., the key creation request data structure (Tag
18 7F31_h)).

19 Entities shall agree on an exact version that is mutually supported. Communication based on
20 different versions used by each entity (e.g., vehicle supporting versions 2.1 and 2.2, device
21 supporting 1.0, 2.0 and 2.3) shall not occur.

22 Introduction of anchor versions (in section 2.10.6) allow both entities to find a commonly
23 supported version.

24 2.10.6 Version Introduction

25 With the introduction of the versioning scheme all domain versions unless explicitly stated in
26 Table 2-2 shall be set to 2.0.

27 The following plan describes the smooth introduction of versioning into the system without
28 breaking compatibility with existing deployments of vehicles, servers and devices.

- 29 - All commands, responses, tags or data elements shall be transmitted as described in
30 Digital Key Specification v1.0.0 or v1.1.0 between devices and vehicles if one entity only
31 supports Domain Version 1.0 of V-OD-FW (referred to as SPAKE2+ Protocol version
32 1.0 in Digital Key Specification v1.0.0 or v1.1.0).
- 33 - For all other cases, any attempts to negotiate versions by an entity as defined in this
34 specification (e.g., using versionUpdate API) shall be gracefully rejected by the other
35 entity.
- 36 - All senders should start implementing and using the new data structures and API versions
37 for each relevant domain version 2.0.
- 38 - All receivers that gracefully fail attempts to negotiate versions are considered to support
39 the CCC Digital Key Release 3 specifications v1.0.0 and v1.1.0.

- 1 - Therefore, in the event that a versioning capable device or vehicle determines that it is
2 interoperating with a non-versioning capable vehicle or device, the sender shall revert to
3 using features defined in CCC Digital Key Release 3 specifications v1.0.0 and v1.1.0.

4 **2.10.7 Version Support**

5 Not all entities may be able to support all versions immediately. Therefore, anchor versions that
6 must be supported by all entities, as a coordinated fallback version, shall be used in order to
7 ensure interoperability. Anchor versions are not dedicated parameters, they are regular version
8 values in the domain version lists, which are determined to be mandatory for all entities.

9
10 Anchor versions shall be agreed upon in the CCC Technical Working Group and updated in
11 Table 2-2.

12 The definition of a grace period to support new versions in all entities after a CCC specification
13 release is outside of this specification. Deprecation of specific data structures/versions/APIs shall
14 be documented in this specification

15 Support period and deprecation rules for older versions are defined outside of this specification.

16 **2.10.7.1 Maintaining compatibility across Versions**

17 To maintain compatibility across differing domain versions supported by devices and vehicles
18 the following rules shall apply:

- 19 1. To prevent buffer overflows on the recipient, while also allowing for future incremental
20 additions as needed, a transmitter shall not exceed the maximum size of the SELECT,
21 SELECT Response, SPAKE2+ Request and SPAKE2+ Verify Commands as defined in
22 Section 5.1 by 128 bytes.
- 23 2. Using the messaging schemes defined in this specification, a CCC Digital Key Product
24 shall identify the version of the product it is interoperating with. Upon identification of
25 the domain version supported by the DK message recipient, the DK message transmitter
26 shall not transmit a message that includes a tag or data element that has not been defined
27 in the specification version supported by the recipient. The domain versions and
28 associated specification versions within which they are defined are outlined in Table 2-2.
- 29 3. Servers shall tolerate the reception of the data elements or server API calls as outlined in
30 Table 2-1, even if they don't implement versioning as defined in CCC Digital Key
31 Specification v1.2.0.

33 *Table 2-1: Reception of Versioning Parameters by non-versioning capable Servers*

Section	API	Recipient	Parameters (JSON)	Notes
17.7.1.2	trackKey	Vehicle OEM Server	vehicleOwnerDeviceFramework VersionList deviceVehicleServerVersionList (D-VS-deviceList)	
17.7.1.3	trackKeyResponse	Device OEM Server	serverSupportedVersions (D-VS-serverList)	If trackKey message does not include vehicleOwnerDeviceFramework

				VersionList or deviceVehicleServerVersionList
17.7.5	versionUpdate	Vehicle OEM Server Device OEM Server		Device OEM calls/tries to call versionUpdate() API on vehicle OEM server Vehicle OEM calls/tries to call versionUpdate() API on Device OEM server

1

2 2.10.8 Version Table

3 Table 2-2 provides a list of domain versions along with active current and past values for each
4 domain version. The table lists the following properties for each domain version value:

- 5 - Anchor Version (Y/N): whether the domain version value is an anchor version or not
- 6 - Specification Version: The version of the Digital Key Specification in which that
7 particular Domain Version value was first introduced
- 8 - Data Structure API: New data structures that were introduced in that specification version
- 9 - Notes: Informative Description of the Changes introduced in that version of the
10 specification

11

Table 2-2: Active Version Table

Domain	Domain Version Value	Anchor Version? (Y/N)	Specification Version	Notes
V-OD-FW	2.0	N	1.2.0	<ul style="list-style-type: none"> - Reuse of SPAKE2+ version Tags for Owner pairing Domain Version (Table 5-4) - Defined Standard and Proprietary access profiles (Table 11-4) - Changed Certificate Version (Tag 41_h) from 01_h to 02_h (Table 11-18) - Converted Key configuration from ASN.1 to BER-TLV - Changed Tag 58_h (Entitlement Data) to Tag 78_h due to change to BER-TLV format (Table 11-18) - Removed variable updateCounter as part of BER-TLV conversion
	1.0	Y	1.1.0/1.0.0	
V-D-BT	2.0	N	1.2.0	<ul style="list-style-type: none"> - Introduced new Tags 5E_h (Table 5-4, Table 11-5) and 5F_h (Table 5-5) to track BLE/UWB versioning - Designed new SPSM Version characteristics and GATT based message exchange to communicate version information (Section 19.2.1.7 and Section 19.2.1.8) - Modified Ranging Capability RQ/RS scheme to include versioning information (Section 19.3.1.1 and Section 19.3.1.2)
	1.0	Y	1.1.0/1.0.0	

V-D-TX	1.0	Y	1.2.0/1.1.0/1.0.0	- Defined protocol to support existing and new Applet versions. Applet version was not incremented in version 1.2.0 of the specification. (Table 15-12, Table 15-29)
D-VS	2.0	N	1.2.0	- Introduced Tag 5F22 _h to include Subject Key Identifier of certificate used for Signature Verification (Table 14-4) - Created new parameters in trackKey API (Section 17.7.1.2) and trackKeyResponse (Section 17.7.1.3) - Introduced new API - versionUpdate (Section 17.7.5) - Modified Unencrypted Event Notification Data (Section 17.8.11) - Added new Server Sub Status Code 50117 (Section 17.10)
DS-VS	2.0	N	1.2.0	- Negotiation of this domain version is out of scope of this specification
	1.0	N	1.1.0	
OD-FD-KS	2.0	N	1.2.0	- Introduced Tags 54 _h (Table 11-6) and 55 _h (Table 11-7) to version friend key sharing - Introduced Tags 7F2D _h and 7F11 _h for Sharing Method Attestation (Table 11-16) - Modified Entitlement Data Schema to Match BER-TLV format (Table 11-17)
	1.0	Y	1.1.0	

1 **2.10.9 Version Deprecation**

2 Version lists in every entity should be modifiable anytime, even without a SW update. This
3 allows to deprecate any deployed version and prevent it from being used. The mechanism to
4 modify version lists in vehicles, servers and devices is proprietary to device and vehicle OEMs
5 and is out of scope.

6 Example: Possible handling of flaw in a newer version:

- 7 - sRTR has been modified in specification between D-VS versions 2.0 and 2.2.
- 8 - sRTR processing with D-VS = 2.2 is flawed in a specific device OS version, but devices with
9 this flaw have already been deployed. All devices also support the anchor version 2.0.
- 10 - Version lists in devices supporting this flawed version are updated to not offer version 2.2
11 anymore, but still support version 2.0.
- 12 - The SW update agreement process is executed (as per section 2.10.4) to inform all entities
13 about the version update.
- 14 - Servers will now send the sRTR based on anchor version 2.0 to the device, which is able to
15 process it correctly.
- 16 - Devices that are fixed (e.g., device OS update installed) will add back D-VS version 2.2 to
17 their version lists so that servers can send sRTR in version 2.2 again.

18

1 3 NFC INTERFACE [WCC1]

2 This section defines the NFC features that shall be implemented by the devices and how these
3 features shall be operated for the Digital Key use case. This specification requires vehicle and
4 device to support NFC-A technology. Support for NFC-B technology and NFC-F technology is
5 optional on either side; corresponding requirements are only applicable if the device or vehicle is
6 intended to support those technologies.

7 3.1 NFC functional requirements

8 This section defines the functional requirements for the NFC implementation on the vehicle and
9 device sides.

10 3.1.1 *Vehicle*

11 The vehicle NFC readers shall be compliant with the poller requirements of the NFC Analog
12 Technical Specification [16] for:

- 13 • Radio Frequency Power and Signal Interface
- 14 • Modulation Poller to Listener – NFC-A
- 15 • Load Modulation Listener to Poller (only for NFC-A)

16 If the vehicle NFC readers are intended to support NFC-B technology, they shall be compliant
17 with the poller requirements for NFC-B as defined in NFC Analog [16]. If the vehicle NFC
18 readers are intended to support NFC-F technology, they shall be compliant with the poller
19 requirements for NFC-F technologies as defined in NFC Analog [16].

20 The vehicle NFC readers shall be compliant with the Poll Mode requirements relevant for the
21 Type 4A Tag Platform as defined in the NFC Digital Protocol Technical Specification [17]. If
22 the vehicle NFC readers are intended to support NFC-B technology, they shall be compliant with
23 the Poll Mode requirements relevant for the Type 4B Tag Platform Subset as defined in NFC
24 Digital Protocol [17]. If the vehicle NFC readers are intended to support NFC-F technology, they
25 shall be compliant to the Poll Mode requirements relevant for the Type 3 Tag Platform
26 Technology Subset as defined in NFC Digital Protocol [17].

27 The vehicle NFC readers shall be compliant with the Poll Mode requirements for the ISO-DEP
28 protocol as defined in [17].

29 3.1.2 *Device*

30 The device NFC implementation shall be compliant with the listener requirements of the NFC
31 Analog Technical Specification [16] for:

- 32 • The Radio Frequency Power and Signal Interface
- 33 • Modulation Poller to Listener – NFC-A
- 34 • Load Modulation Generic (only for NFC-A)
- 35 • Subcarrier Load Modulation NFC-A

36 The device NFC implementation may be compliant with the listener requirements for NFC-B
37 and/or NFC-F technologies as defined in NFC Analog [16].

1 The device NFC implementation shall be compliant with the Listen Mode requirements relevant
2 for the Type 4A Tag Platform defined in NFC Digital Protocol [17]. The device NFC
3 implementation may be compliant with the Listen Mode requirements for the Type 4B Tag
4 Platform and/or the Type 3 Tag Platform as defined in NFC Digital Protocol [17].

5 The device NFC readers shall be compliant with the Listen Mode requirements for the ISO-DEP
6 protocol as defined in [17].

7 The device NFC implementation shall be compliant with the generic requirements for Listen
8 Mode as defined in the NFC Activity Technical Specification [18]. The implementation also
9 shall be compliant with the states and transitions of the Listen-Mode State Machine as defined in
10 NFC Activity [18], which are relevant for the Type 4A Tag Platform. The device may implement
11 additional parts of the Listen Mode State Machine, namely the states and transitions relevant for
12 the Type 4B Tag Platform and/or Type 3 Tag Platform.

13 The device shall configure the Listen-Mode State Machine using the following settings:

- CON_LISTEN_T4ATP shall be enabled.

15 Other configuration parameter values are implementation specific. If the device is intended to
16 listen for NFC-B technology, it shall enable CON_LISTEN_T4BTP. If the device is alternatively
17 or additionally intended to listen for NFC-F technology, it shall enable CON_LISTEN_T3T.

18 The device NFC implementation shall support the following power modes:

- **Battery Operational Mode:** The battery of the device has sufficient power to support all
its functions.
- **Battery Low Mode:** The battery of the device has reached a threshold at which many
functions (e.g. display) are automatically disabled, but the NFC Controller function will
still be powered.

24 The device NFC implementation shall implement means to correctly route traffic addressed to
25 the Digital Key applet or to the Digital Key framework. Routing can be based on the application
26 identifiers contained in the SELECT command defined in [1]. One example of such means is the
27 routing mechanism as defined in the NFC Controller Interface Technical Specification [19],
28 specifically the AID-based Route Selection Process.

29 If the device is configured to work as a Digital Key, routing to the Digital Key applet shall be
30 enabled in Battery Operational Mode and Battery Low Mode. Additionally, routing to the Digital
31 Key framework shall be enabled during Owner Pairing in Battery Operational Mode.

32 **3.2 NFC Procedures**

33 This section defines how the vehicle operates the NFC interface to detect devices and to establish
34 and end NFC communication with a device.

35 **3.2.1 NFC Polling and Link Setup Procedure**

36 To turn on the RF field, the vehicle may perform RF Collision Avoidance as defined in NFC
37 Activity [18].

38 The vehicle shall run a polling loop that includes the Technology Detection activity as defined in
39 NFC Activity [18], with the following configuration settings:

- CON_POLL_A shall be enabled

1 Other configuration parameter values are implementation-specific. If the vehicle intends to poll
2 for NFC-B, it shall enable CON_POLL_B. If the vehicle intends to poll for NFC-F, it shall
3 enable CON_POLL_F.

4 If the Technology Detection procedure has identified one of these technologies, the vehicle NFC
5 readers shall perform the Collision Resolution activity as defined in NFC Activity [18].

6 After the Collision Resolution activity, and if an NFC-A or NFC-B device indicating support for
7 the ISO-DEP protocol as defined in NFC Digital Protocol [17], has been identified, the vehicle
8 shall perform the Device Activation activity with the following configuration:

- 9 • INT_TECH_SEL shall be set to 000_b for NFC-A or 001_b for NFC-B
- 10 • INT_PROTOCOL shall be set to 001_b to activate the ISO-DEP protocol

11 Other configuration parameter values are implementation-specific. If the vehicle intends to
12 activate a device using NFC-F technology, the vehicle NFC readers shall set INT_TECH_SEL to
13 010_b and INT_PROTOCOL to 100_b (Type 3 Tag Platform).

14 *Note:* The above requirements do not cover the case of multiple identified devices. If multiple
15 devices are identified by the vehicle, the handling is implementation-specific.

16 3.2.2 NFC Data Transfer Procedure

17 The APDU exchanges defined in this specification shall be performed during the Data Exchange
18 activity as defined in NFC Activity [18].

19 For devices using NFC-A or NFC-B technology, the vehicle NFC readers shall configure Data
20 Transfer activity as follows:

- 21 • INT_PROTOCOL shall be set to 001_b to use the ISO-DEP protocol

22 If the vehicle intends to activate a device using NFC-F technology, the vehicle NFC readers shall
23 set INT_PROTOCOL to 100_b (Type 3 Tag Platform).

24 *Note:* Please refer to [Appendix E](#) for details of how to operate the Digital Key contactless
25 transactions over NFC-F.

26 After successful Device Activation using the ISO-DEP protocol, the vehicle shall operate the
27 ISO-DEP protocol. The ISO-DEP protocol performs error processing in case of timeout or
28 transmission errors. If the ISO-DEP protocol cannot re-establish communication after an error, it
29 will raise an Unrecoverable Timeout Error or Unrecoverable Transmission Error, in which case
30 the vehicle shall perform an NFC reset procedure.

31 3.2.3 NFC Link Teardown Procedure

32 To tear down an NFC link, the vehicle NFC readers shall perform the Device Deactivation
33 activity as defined in NFC Activity [18].

34 For devices using NFC-A or NFC-B technology, the vehicle NFC readers shall configure Device
35 Deactivation activity as follows:

- 36 • INT_PROTOCOL shall be set to 001_b to deactivate the ISO-DEP protocol

37 *Note:* Please refer to Appendix [E.5.6](#) for devices using NFC-F technology.

1 3.2.4 *NFC Reset Procedure*

- 2 To reset the NFC communication, the vehicle shall perform a reset of the Operating Field as
3 defined in NFC Analog [16] and shall not generate any Operating Field for a time of at least 50
4 ms.
- 5 Afterwards the vehicle should perform an NFC polling and link setup procedure.
- 6 *Note:* The NFC reset procedure uses a longer Operating Field off duration than required for other
7 NFC use cases (the minimum time for an NFC reset defined in [16] is 5.1 ms).

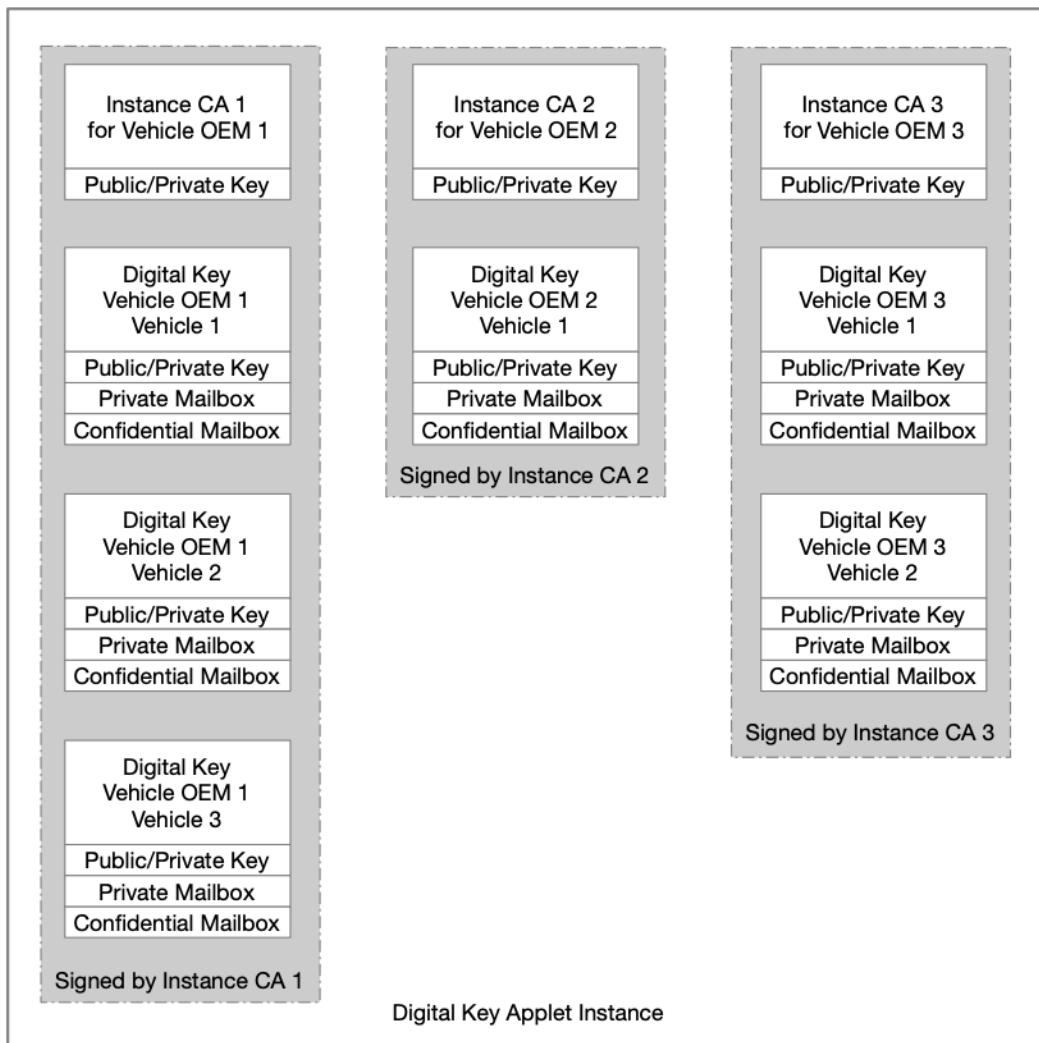
1 4 DATA STRUCTURE

2 4.1 Applet Instance Layout

3 One Digital Key applet instance hosts all the data necessary for a device to perform Digital Key
4 services, as shown in [Figure 4-1](#), including all Digital Keys and one or more Instance CAs.

5 A Digital Key is represented by a structure and is described in Section 4.2. The **Instance CAs**
6 are intermediate CAs that represent the Device OEM CA and reside within the device. An
7 Instance CA attests to (the public key of) a Digital Key created in the applet instance. Its
8 signature can be verified using the Device OEM CA Certificate. A different Instance CA shall be
9 used for each Vehicle OEM.

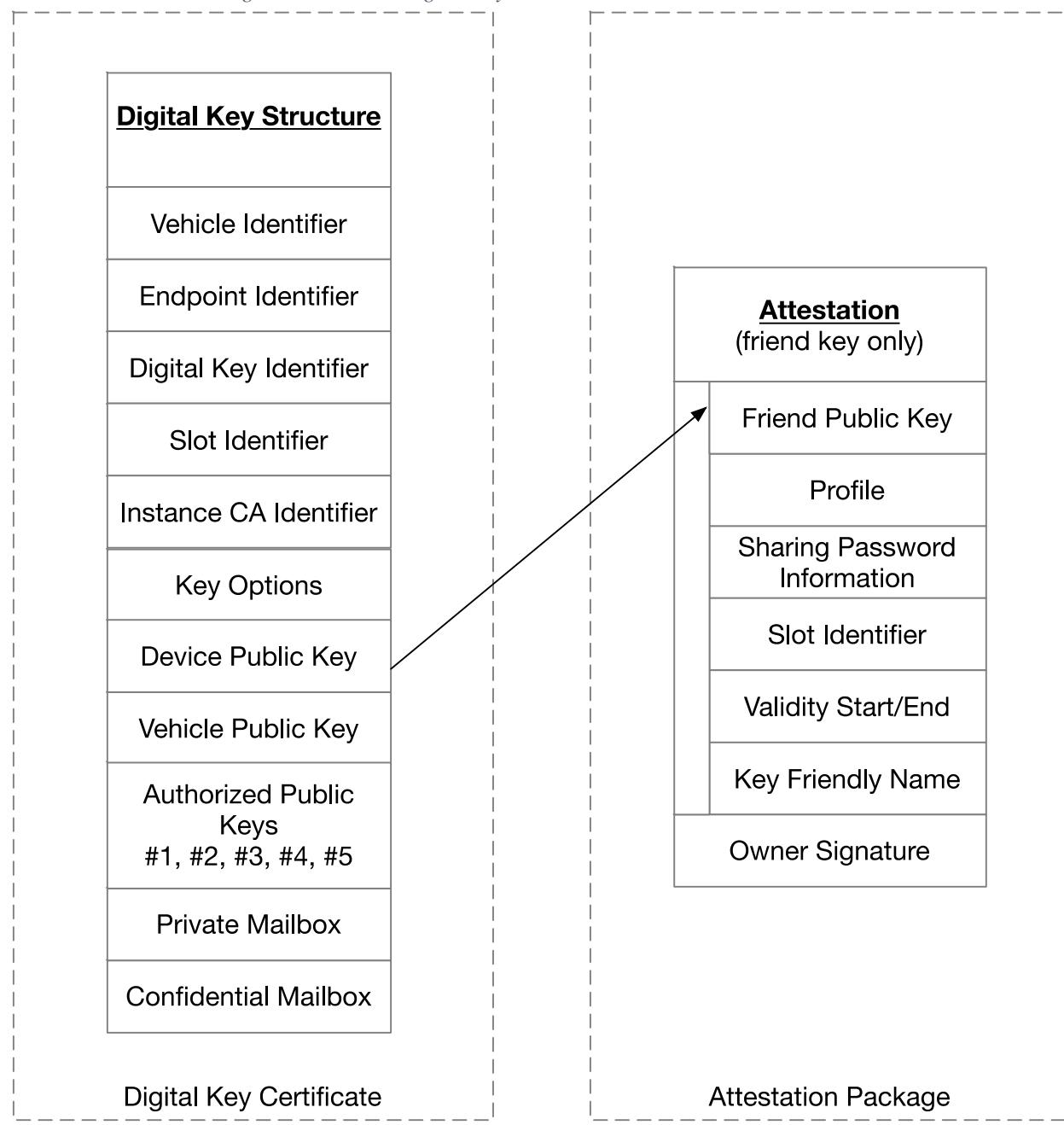
Figure 4-1: Applet Instance Layout



4.2 Digital Key Structure

- A Digital Key structure is stored in the applet instance and contains a public/private key pair, a private mailbox, a confidential mailbox, and other elements, as shown in [Figure 4-2](#).
An owner Digital Key consists of the Digital Key structure only. It does not have any limitations in validity and access rights.
A friend Digital Key consists of the Digital Key structure and an entitlements attestation section, which is linked to the friend key by containing the same device public key and by being signed by the owner key.

Figure 4-2: Friend Digital Key Structure and Entitlements Attestation



1 The elements of the Digital Key structure are:

- 2 • **Vehicle Identifier (also referred to as vehicle_identifier):** Uniquely identifies the
3 vehicle to which the Digital Key is associated. The vehicle identifier is unique per
4 Vehicle OEM. The vehicle identifier is not identical to the VIN used in the automotive
5 industry. It is transmitted by the vehicle during a contactless transaction.
- 6 • **Endpoint Identifier:** Used for device-internal key management. The device creates the
7 endpoint identifier based on the rules given in Appendix [B.2](#). In Section [15](#), the name
8 “endpoint_identifier” is used for key creation. The identifier is reflected in the subject
9 field of the Digital Key Certificate [H], also referred to as “Digital Key Endpoint
10 Certificate.” Rules associated with formatting and parsing of certificates are described in
11 Section A.4.7 of [Appendix A](#).
- 12 • **Digital Key Identifier (also referred to as keyID):** Enables the identification of the
13 Digital Key within the Vehicle OEM Server system. The Digital Key identifier shall be
14 unique per Vehicle OEM. The element is named “subject key identifier” in Section [14](#),
15 based on the X.509 certificate definition. It is the SHA-1 (see [\[23\]](#) and [\[24\]](#)) hash value
16 over the device public key. See Appendix [B.2](#) for details.
- 17 • **Slot Identifier:** A value provided by the vehicle to the owner device to locally identify
18 the key used. The value is transmitted within a contactless transaction. When sharing a
19 key, the owner device or the Vehicle OEM Server provides a slot identifier value to the
20 friend device to be used to create the friend key and, if applicable, to identify the
21 associated immobilizer token.
- 22 • **Instance CA Identifier:** Refers to the Instance CA that has signed the Digital Key. It is
23 assigned by the device at Instance CA creation.
- 24 • **Key Options:** Indicates which transactions (fast, standard) the key is allowed to perform.
25 There are several more options, such as executing a transaction over the wired interface,
26 etc. Note that those options are not access rights.
- 27 • **Device Public Key:** (Also referred to as **device.PK** and **endpoint.PK** in this
28 specification.) Used in the standard transaction. Device public key shall be globally
29 unique. It is generated at endpoint creation and is stored in the vehicle. It is represented
30 by the “subjectPublicKey” field of the endpoint certificate in Section [15](#). Note that
31 Device.Enc.PK (See Section [14](#)) is different from device public key. Similarly, device
32 private key, device.SK, and endpoint.SK are used interchangeably in this specification.
- 33 • **Vehicle Public Key:** (Also referred to as **Vehicle.PK** and **vehicle_pk** in this
34 specification.) Used in the standard transaction. The vehicle public key is the same for all
35 devices associated with the same vehicle.
- 36 • **Authorized public keys:** Provided by the vehicle and are used as root in the verification
37 chain of a friend public key at key sharing (see Section [11](#)). In this version of the
38 specification, the vehicle shall include only one single authorized_PK. Usage of
39 additional authorized public keys is left for future use cases.
- 40 • **Private Mailbox:** A buffer that allows for the transfer of elements to or from the vehicle
41 during a Digital Key Transaction (see Section [4.3.1](#) for more details).
- 42 • **Confidential Mailbox:** A buffer that allows for the transfer of elements which need
43 confidentiality protection to or from the vehicle during a Digital Key Transaction (see
44 Section [4.3.2](#) for more details).

All relevant data elements of the Digital Key shall be verified by the vehicle during the owner pairing procedure (see Section 6) before accepting the device public key, and by the owner device during key sharing (see Section 11) before signing the friend device public key.

Before signing the friend's public key, the owner verifies that the friend's public key pair has been created on an eligible SE. The verification chain starts with an authorized public key in the owner Digital Key structure, which has been provided and is trusted by the vehicle, e.g., the Vehicle OEM CA public key (see Section 16.8).

The elements provided in the attestation package (for friend keys only) are:

- **Friend Public Key:** The public key created by the friend device together with the private key. During key sharing, this public key shall be signed by the owner device together with the other fields in the attestation package.
- **Profile:** Selected by the sender of a Shared Key. It needs to comply with the Vehicle OEM policy and is verified by the vehicle. Digital Keys that do not comply with the Vehicle OEM policy settings are rejected by the vehicle (i.e., their public key is not accepted). The Vehicle OEM policy is out of scope of this specification.
- **Sharing password information:** Contains the seed for the vehicle to generate a sharing password from the shared secret established at owner pairing. It also contains the information regarding whether the owner device policy requires (or not) that the vehicle requests a sharing password to the friend before activating the shared Digital Key.
- **Validity start date:** The earliest date and time the key can be used. The Digital Key may be accepted when presented to a vehicle earlier, but it shall not have effect before the date and time is reached. This field requires an internal time to be available in the vehicle. Accuracy, reliability and security of this internal time are dependent on Vehicle OEM policy and vehicle capabilities.
- **Validity end date:** The latest date and time the key can be used. This field requires an internal time to be available in the vehicle. Accuracy, reliability and security of this internal time are dependent on Vehicle OEM policy and vehicle capabilities.
- **Key friendly name:** A field conveying a user-friendly name for the Digital Key that should be assigned at key sharing (see Section 11). It may be the same name as in the owner's contacts when sharing a Digital Key. The owner should be allowed to edit the name for identification and personalization. For privacy reasons, friendly name should not contain private information such as full name of the friend.

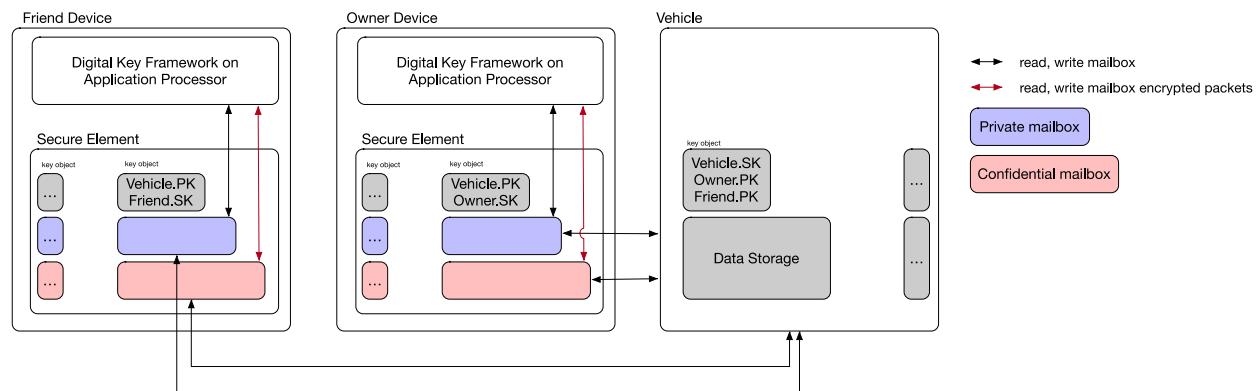
Digital Key elements except private and confidential mailboxes are not changeable during the lifetime of a Digital Key. A Digital Key can be created, terminated, and deleted. In "terminated" state, it is not usable but still able to provide the termination attestation until it is finally deleted from memory. The Digital Key states are managed internally by the applet.

4.3 Mailboxes

The mailbox mechanism allows the SE to store small data buffers accessible by the Digital Key framework and by the vehicle. These data buffers are stored in the SE's non-volatile memory and are read from/written to during transaction, key sharing, and pairing. Each Digital Key (referred to as an Endpoint) created in the applet instance of a device can have two mailbox types (private and confidential) with different security properties and targeted at different usages. The

1 mailboxes support random access, whereby content can be read from/written to using
2 offset/length parameters to access them.
3 Confidential mailboxes are not shared between Endpoints, and consequently only a vehicle
4 paired to an Endpoint can access the confidential mailbox of such Endpoint. Private mailboxes
5 are not shared among Endpoints, either; however, the Digital Key framework and the vehicle can
6 freely read from and write to their contents.
7 The confidential mailbox is meant for storage of secret data that is never available in plaintext to
8 the Digital Key framework. In addition, only the associated vehicle has read/write access
9 privileges to the contents of this mailbox. The Digital Key framework can only extract or push
10 portions of the confidential mailbox in an encrypted format. The confidential mailbox is used for
11 storage of immobilizer tokens.
12 The private mailbox is meant for storage of data accessible in plaintext to the Digital Key
13 framework, the vehicle, and the SE. The use of the private mailbox guarantees that this data is
14 never transferred in plaintext -, and the mailbox access is only possible using the EXCHANGE
15 command. The private mailbox is used for transfer of the attestation package and to indicate
16 which mailbox fields are present/absent (signaling) as described in the following sections.

17 *Figure 4-3: Mailboxes Read/Write Permissions*



19 4.3.1 Private Mailbox

20 The private mailbox can be read from and written to by the Digital Key framework using the
21 device internal wired link as described in Section 15. It can also be read from and written to by
22 the vehicle once the secure channel described in Section 15 is established between a vehicle and
23 device using the Digital Key. Data exchanged with vehicle is always protected by the established
24 secure channel.

25 The mapping of the private mailbox defines data structures required by the Digital Key
26 framework and provides a bi-directional signaling mechanism to indicate to the receiver that new
27 data is available as described in [Figure 4-4](#).

28 The private mailbox enables the Vehicle OEM to define its data structures by the signaling
29 mechanism. These data structures are opaque to the Digital Key framework and are only known
30 to the vehicle and a Vehicle OEM app or the Vehicle OEM Server. Arbitrary data that can be
31 stored in and retrieved from the private mailbox associated with each Digital Key includes the
32 signaling bitmap, the slot identifier bitmap, the slot identifiers, the Vehicle OEM proprietary
33 data, and the attestation package, in case a key is presented to a vehicle. The private mailbox

format is determined at owner pairing. It shall be the same for owner and friend mailboxes, though certain fields may not be used in friend mailboxes.

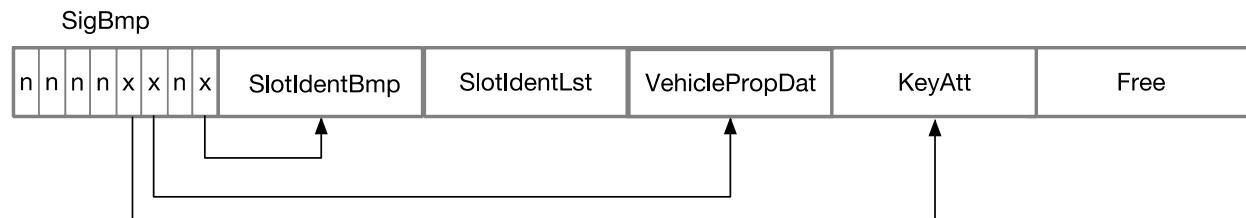
The signaling bitmap is required for owner and friend mailboxes. The slot identifier bitmap is used by the owner device to keep track of available slot identifiers and, if applicable, immobilizer tokens for Digital Key sharing. Slot identifiers are only used in the owner mailbox. The slot identifiers and, if applicable, the immobilizer tokens are provided to the owner device either by the vehicle or by the Vehicle OEM Server. This is controlled by the vehicle in the SHARING_CONFIGURATION field in [Table 5-14](#) (Tag 7F60_h and Tag DA_h).

The slot identifiers and, if applicable, the immobilizer tokens for the friend device are provided by the owner device or by the Vehicle OEM Server. This is controlled by the owner device in the SHARING_CONFIGURATION field in Table 11-6 (Tag 7F60_h and Tag DA_h) which derives from the SHARING_CONFIGURATION field in [Table 5-14](#) (Tag 7F60_h) the owner device received during owner pairing.

Vehicle proprietary data may be used in owner and friend mailboxes. The maximum number usable keys in a vehicle is implementation-specific. The limit of seven slot identifiers for friend keys in the private mailbox refers to the maximum number of shareable friend keys after which the owner is required to obtain new slot identifiers from the vehicle.

Key attestation packages are used with owner and friend mailboxes. Note that these mailboxes have different purposes and structures.

Figure 4-4: Owner Device Private Mailbox Signaling



n-not used (RFU), x -used (any value)

The **Signaling Bitmap** as contained in the SigBmp field allows the vehicle and the device to inform the other party of changes occurring or actions to be taken on some mailbox field by the other party. It is the responsibility of the vehicle or the device that consumes the bit to reset the bit and delete the data if needed. The signaling relationship and the offsets of the data elements are configured by the vehicle at owner pairing (See Section [6](#)).

The length of this field is 1 byte. This field is big-endian ordered, meaning that bit 0 is the rightmost bit of the field.

The following constants are defined for the rest of this document:

SLOT_IDENTIFIER_BITMAP_BIT = 00_h

SLOT_IDENTIFIERS_BIT = 01_h

VEHICLE_OEM_PROPRIETARY_DATA_BIT = 02_h

ATTESTATION_PACKAGE_BIT = 03_h

SLOT_IDENTIFIER_LENGTH = (VEHICLE_OEM_PROPRIETARY_DATA_OFFSET -

1 SLOT_IDENTIFIER_OFFSET) / 8

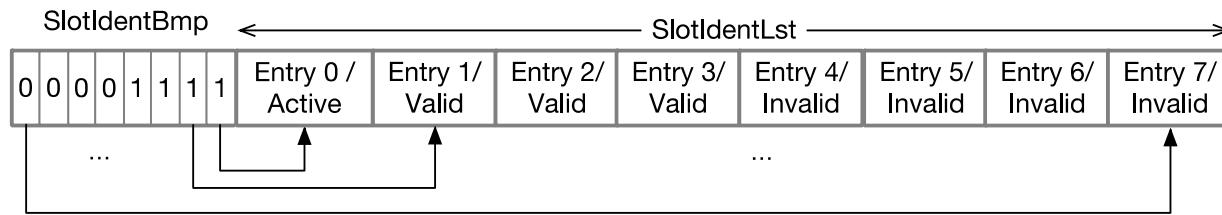
2 [Table 4-1](#) shows the purpose for each bit in the SigBmp field. The signaling bits may be read
3 and/or written by the vehicle and/or the device.

4 *Table 4-1: Signaling Bitmap Decoding*

Bit	Name	Value	Description	Set by
Bit0	SlotIdentBmp	0	All slot identifiers valid	Vehicle
Bit0	SlotIdentBmp	1	Not all slot identifiers valid	Device (when sharing) Vehicle (initial refill at owner pairing)
Bit1	RFU	0	RFU	RFU
Bit1	RFU	1	RFU	RFU
Bit2	VehiclePropDat	0	Device has read vehicle proprietary data	Device
Bit2	VehiclePropDat	1	Vehicle has updated vehicle proprietary data	Vehicle
Bit3	KeyAtt	0	No key attestation package present	Vehicle/Device (attestation package retrieved successfully or is not required any more)
Bit3	KeyAtt	1	Key attestation package present	Vehicle/Device (attestation package written in mailbox)
Bit4	RFU	0	RFU	RFU
Bit4	RFU	1	RFU	RFU
Bit5	RFU	0	RFU	RFU
Bit5	RFU	1	RFU	RFU
Bit6	RFU	0	RFU	RFU
Bit6	RFU	1	RFU	RFU
Bit7	RFU	0	RFU	RFU
Bit7	RFU	1	RFU	RFU

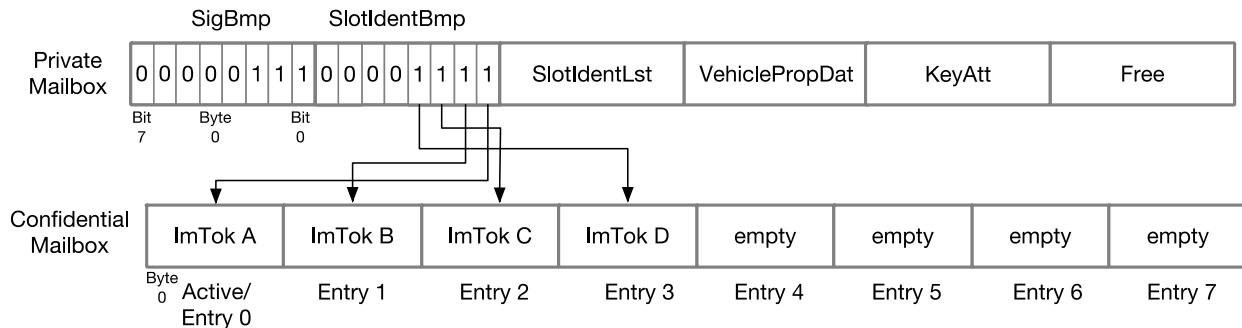
5 The **Slot Identifier Bitmap** as contained in the **SlotIdentBmp** field signals the validity of a slot identifier in the corresponding entry of the slot identifier, as shown in [Figure 4-5](#).

7 *Figure 4-5: Slot Identifier Bitmap Assignment*



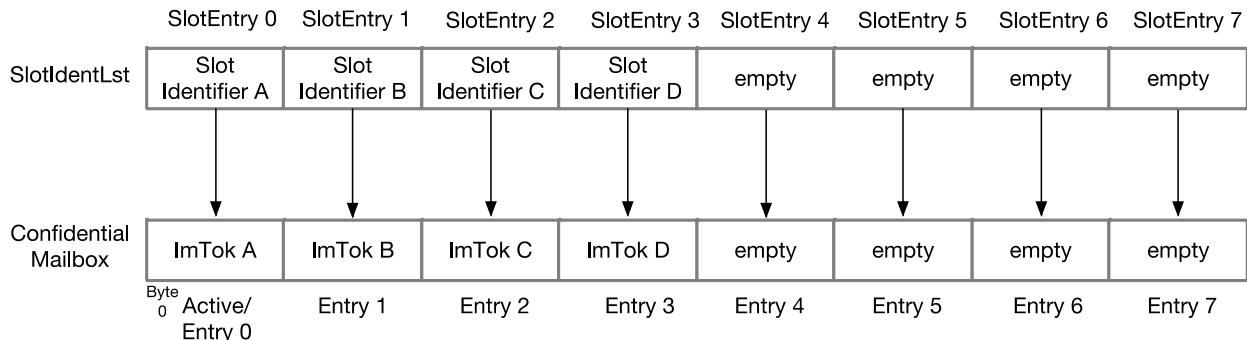
- 1 The length of this field is 1 byte. The most significant bit represents the state of entry 7; the least
2 significant bit represents the state of entry 0.
- 3 If a bit is set to 0, the slot identifier is invalid. If immobilizer tokens are retrieved from the
4 vehicle (see [Table 5-14](#)), in the next standard transaction, the vehicle shall write a new slot
5 identifier value and set the bit to 1 and also fill up a new immobilizer token, if applicable.
- 6 *Note:* The invalid slot identifier value is still present in the entry. The rules for managing the slot
7 identifier values are described in Section [13.7](#).
- 8 If a bit is set to 1, a valid slot identifier is present in the corresponding entry in the SlotIdentLst
9 field, and the corresponding immobilizer token is also present in the corresponding entry in the
10 confidential mailbox.
- 11 Each bit in the SlotIdentBmp shall also indicate the presence of an immobilizer token in the
12 corresponding entry and memory offset of the confidential mailbox (see [Table 4-3](#) and [Figure](#)
13 [4-6](#)). This bitmap allows the device and the vehicle to synchronize on the confidential mailbox
14 content, if immobilizer tokens are retrieved from the vehicle.
- 15 The active owner immobilizer token (ImTok A) shall always be located in entry 0
16 (corresponding to memory offset 0). Its presence shall be signaled by setting bit 0 to value 1 in
17 the SlotIdentBmp. Other immobilizer tokens (e.g., entry 1 to 3 in [Figure 4-6](#)) are used for friend
18 sharing.

19 *Figure 4-6: Owner Device Immobilizer Token Signaling*



- 20
- 21 The **slot identifiers** as contained in the SlotIdentLst field help the vehicle keep track of which
22 slot identifiers and immobilizer tokens (if slot identifiers and immobilizer tokens are retrieved
23 from the vehicle) have been distributed or are in use. SlotIdentLst field is comprised of eight
24 SlotEntry subfields, and each of the SlotEntry subfields contains a slot identifier value, as shown
25 in [Figure 4-7](#). The length of the SlotIdentLst field shall be eight times the size of a slot identifier.
26 Each slot identifier is mapped to an immobilizer token at the same position index in the
27 confidential mailbox. The length of this data field is defined by
28 (VEHICLE_OEM_PROPRIETARY_DATA_OFFSET – SLOT_IDENTIFIERS_OFFSET).

Figure 4-7: Slot Identifier List to Confidential Mailbox mapping



Slot identifiers are issued by the vehicle or by the Vehicle OEM Server. They are provisioned by the vehicle to the owner device along with associated immobilizer tokens, if immobilizer tokens are retrieved by the vehicle. Alternatively, they are provisioned by the Vehicle OEM Server after owner pairing or key sharing. The vehicle guarantees that slot identifiers are unique over the lifetime of a vehicle.

Slot identifiers and immobilizer tokens are included in the data provided to the friend device when sharing a key (See Section [11](#)).

10 The **Vehicle OEM proprietary data** as contained in the VehiclePropDat field is arbitrary data
11 specific to each Vehicle OEM.

12 The length of this data field is defined by (ATTESTATION_PACKAGE_OFFSET -

13 VEHICLE_OEM_PROPRIETARY_DATA_OFFSET). The Vehicle OEM proprietary data is
14 present in the private mailbox until consumed and cleared by the device or by the vehicle.

If the length of this data field is greater than 0, the Vehicle OEM proprietary data should be read by the Digital Key framework within the Digital Key termination process and sent along with the termination attestation to the KTS. The Vehicle OEM proprietary data may not be sent in all cases due to technical constraints.

19 The **attestation package** as contained in the KeyAtt field can be used for i) providing the Opaque
20 Attestation (see Section 6.3.4.1) to the device, ii) providing the kts-signature, indicating that the owner
21 key was successfully registered at the KTS, to the vehicle (see Section 6.3.4.4), iii) for supporting
22 friend sharing (providing the key Attestation Package to the vehicle; see Section 11.8.4), as well as for
23 iv) resuming a suspended key using a Resume Attestation provided by the device (see Section 13.4.3).

24 The attestation package is present in the private mailbox until consumed and cleared by the vehicle or
25 the device (depending on the attestation type). The length of this data field is defined by
26 (MAILBOX_CONTENT_END_OFFSET – ATTESTATION_PACKAGE_OFFSET).

27 [Table 4-2](#) describes the relationship among data elements (“fields”), memory offsets, and
28 signaling bit. The sizes of the data elements in the private mailbox are defined by offsets, which
29 are provided by the vehicle during owner pairing. Data structure sizes are calculated by
30 subtracting the offset values.

Table 4-2: Private Mailbox Content

Field	Memory Offset	Description	Sig-Bit
SigBmp	SIGNALING_BITMAP_OFFSET	Signaling bitmap	n/a

Field	Memory Offset	Description	Sig-Bit
SlotIdentBmp	SLOT_IDENTIFIER_BITMAP_OFFSET	Slot identifier bitmap	Bit 0
SlotIdentLst	SLOT_IDENTIFIERS_OFFSET	Vehicle-assigned values to manage immobilizer tokens and termination	Bit 1
VehiclePropDat	VEHICLE_OEM_PROPRIETARY_DATA_OFFSET	Vehicle OEM proprietary data structure	Bit 2
KeyAtt	ATTESTATION_PACKAGE_OFFSET	Cryptographic key attestation package	Bit 3
MbxEnd	MAILBOX_CONTENT_END_OFFSET	Offset of first byte of free memory after last data structure	n/a

- 1
- 2 The Sig-Bit column in the above table identifies the bit of the signaling bitmap responsible to
3 signal a value change in the associated data structure. The association is configured by the
4 vehicle (see [Table 5-13](#)).
- 5 SIGNALING_BITMAP_OFFSET shall be set to 0.
- 6 SLOT_IDENTIFIER_BITMAP_OFFSET shall be set to 1. The field SlotIdentBmp assigns eight offsets into the confidential mailbox (entries 0–7) to the eight signaling bits.
- 8 SLOT_IDENTIFIERS_OFFSET defines the area used by the Vehicle OEM to store eight slot identifiers, which are used for management of slot identifiers and immobilizer tokens if slot identifiers and immobilizer tokens are retrieved from the vehicle. The size of the SlotIdentLst data block shall be a multiple of eight. The size of a slot identifier is determined by the size of tag 4E_h in [Table 15-13](#).
- 13 VEHICLE_OEM_PROPRIETARY_DATA_OFFSET defines the start position of the structure VehiclePropDat, defined by the Vehicle OEM. This structure is opaque to the Digital Key framework, and part of the structure may be disclosed to the Digital Key framework.
- 16 ATTESTATION_PACKAGE_OFFSET defines the start position of the structure KeyAtt used for key sharing on friend devices when the vehicle is offline at the first transaction (see Section [11](#)).
- 19 MAILBOX_CONTENT_END_OFFSET indicates the offset of first byte of the free memory after the last data structure of the private mailbox.
- 21 The order of the data structures in the mailbox buffer shall be in the order of the associated signaling bits (S-Bit), starting with Bit 0.
- 23 The offset values are defined together by both the Vehicle OEM and the Device OEM and are out of scope of this specification.
- 25 **4.3.2 Confidential Mailbox**
- 26 The confidential mailbox can be read from and written to by the vehicle once the secure channel described in Section [7](#) or [8](#) is established between a vehicle and device using the Digital Key.
28 Data exchanged with vehicle is always protected by the established secure channel. The

1 confidential mailbox can also be written to by the Digital Key framework via the internal wired
2 link when data is encrypted with the Endpoint encryption public key. The content of the
3 confidential mailbox may be exported in encrypted form with the encryption public key of
4 another trusted SE if allowed by Endpoint configuration and with user consent.
5 If required by the Vehicle OEM, the immobilizer token may be stored and retrieved in the
6 confidential mailbox and may be associated with each Digital Key.
7 The friend confidential mailbox typically contains only one entry corresponding to its active
8 immobilizer token. The active immobilizer token is a secret piece of data to be presented to the
9 vehicle during a Digital Key transaction.
10 The owner confidential mailbox typically contains eight entries. The first entry (index 0) is
11 dedicated to the active immobilizer token, whereas the other seven entries contain immobilizer
12 tokens to be distributed to friends during sharing operations.
13 [Table 4-3](#) describes the confidential mailbox memory mapping when immobilizer tokens are
14 required.

15 *Table 4-3: Confidential Mailbox Content*

Memory Offset	Description
0	active immobilizer token, 0
IMMOBILIZER_TOKEN_LENGTH	immobilizer token 1
2 × IMMOBILIZER_TOKEN_LENGTH	immobilizer token 2
3 × IMMOBILIZER_TOKEN_LENGTH	immobilizer token 3
4 × IMMOBILIZER_TOKEN_LENGTH	immobilizer token 4
5 × IMMOBILIZER_TOKEN_LENGTH	immobilizer token 5
6 × IMMOBILIZER_TOKEN_LENGTH	immobilizer token 6
7 × IMMOBILIZER_TOKEN_LENGTH	immobilizer token 7

16 The active immobilizer token shall be present at offset 0 (see [Figure 4-6](#)) and is required by the
17 vehicle during a Digital Key transaction. This token shall be present on owner and friend devices
18 if immobilizer tokens are required by the Vehicle OEM.
19 The immobilizer tokens present in entries 1 to 7 shall be used for distribution during key sharing
20 (see Section [11](#)).
21 The IMMOBILIZER_TOKEN_LENGTH is provided by the Digital Key framework. It is
22 transmitted in the mailbox configuration table in the first NFC session (see [Table 5-13](#)).

23 4.3.3 *Immobilizer Token*

24 An optional Vehicle OEM requirement for the device is to store and release an immobilizer
25 token, which is a confidential data element provided to the vehicle when it is requested (e.g., at
26 engine start). The format and content of the immobilizer token are out of scope of this
27 specification.
28 If implemented, either the vehicle or the Vehicle OEM Server provides the immobilizer tokens to
29 the owner device during or after owner pairing. The immobilizer tokens for the friend devices are
30 provided either by the owner device or the Vehicle OEM Server. The Immobilizer tokens are

- 1 stored in pre-allocated entries in the confidential mailbox. The confidential mailbox size shall be
2 configured to eight times the size of the immobilizer token.
3 Immobilizer tokens shall be associated with a slot identifier, as shown in [Figure 4-7](#).

4 **4.3.4 Mailbox Access Rights**

5 The following table summarizes access rights for the mailboxes from vehicle and Digital Key
6 framework.

7 *Table 4-4: Mailbox Access Rights*

Mailbox	Digital Key Framework	Vehicle
Confidential	write encrypted, export encrypted	read, write
Private	read, write	read, write

1 5 OWNER PAIRING COMMANDS [WCC1/WCC2/WCC3]

2 This section defines the commands and data elements used for the owner pairing procedure.
3 The owner device (framework) – vehicle version (V-OD-FW), which is exchanged in the
4 SELECT (5.1.1) and SPAKE2+ REQUEST (5.1.2) commands covers all commands defined in
5 Chapter 5.
6 The digital key applet protocol versions cover the applet transactions with the vehicle for unlock,
7 engine start, etc., as described in sections 7, 8 and 10. The applet APIs used for endpoint
8 creation, set confidential data, setup endpoint, etc. are managed based on the owner device
9 (framework) – vehicle version.

10 5.1 Supported Commands for Owner Pairing

11 [Table 5-1](#) lists all commands defined for the owner pairing flow as described in Section [6](#).

12 *Table 5-1: Owner Pairing Command Set*

Command	Instruction Byte (hex)	Implemented by
SELECT	A4	Digital Key Framework
SPAKE2+ REQUEST	30	Digital Key Framework
SPAKE2+ VERIFY	32	Digital Key Framework
WRITE DATA	D4	Digital Key Framework
GET DATA	CA	Digital Key Framework
GET RESPONSE	C0	Digital Key Framework
OP CONTROL FLOW	3C	Digital Key Framework
See Table 15-1	See Section 15.3.2	Digital Key Applet

13 [Table 5-2](#) lists the generic status words to be retrieved in case of error during basic input
14 command checking, which apply to all owner pairing commands listed in [Table 5-1](#).

16 *Table 5-2: Generic Status Words*

Status Word	Comment
6700 _h	wrong length
6A80 _h	Incorrect parameters in command payload
6A82 _h	file not found
6B00 _h	wrong P1 or P2
6C00 _h	wrong Le
6D00 _h	wrong INS code
6E00 _h	wrong CLA code
9000 _h	command successfully executed

1 5.1.1 *SELECT Command*

2 5.1.1.1 *Purpose*

3 The vehicle sends the SELECT AID command to the device. The Digital Key framework AID is
4 A0000008094343444B467631_h.

5 When the Digital Key framework is selected, the device shall respond with the data as described
6 in [Table 5-3](#).

7 The device shall indicate its current pairing state to the vehicle. The possible states are:

- 8 • Not in pairing mode
- 9 • Pairing mode started and pairing password entered

10 The SELECT command used to select the Digital Key applet instance (using Instance AID) is
11 described in Section [15.3.2.1](#).

12 5.1.1.2 *Definition*

13 **command: 00 A4 04 00 Lc [Digital_Key_Framework_AID] 00**

14 **response: [Table 5-3]90 00**

15 *Table 5-3: Response to SELECT Command*

Tag	Length (bytes)	Description	Field is
5A _h	2 × n	n supported vehicle – owner device framework versions (V-OD-FW-deviceList). (ver.high ver.low).	mandatory
5C _h	2 × m	m supported Digital Key applet protocol versions (ver.high ver.low)	mandatory
D4 _h	1	00 _h = not in pairing mode 02 _h = pairing mode started, and pairing password entered	mandatory

16 5.1.1.3 *Usage*

17 The device shall return all supported owner device framework – vehicle (V-OD-FW) and Digital
18 Key applet protocol versions using two bytes per version in big-endian coding.

19 The reception of at least 16 versions shall be supported by all entities. An entity may transmit up
20 to 16 versions.

21 Version h.l is represented as ver.high= ‘h’ (one byte) and ver.low= ‘l’ (one byte).

22 The versions shall be ordered from the highest to the lowest. At least one version shall be
23 provided; otherwise the service is considered as not available.

24 Domain version V-OD-FW 1.0 maps to SPAKE2+ protocol version 1.0 and Domain version V-
25 D-TX 1.0 maps to Digital Key Applet protocol version 1.0 (coded 0100_h). Domain versions and
26 their status as Anchor versions are listed in Table 2-2. The vehicle shall match the versions with
27 its supported versions as described in Section [6.3.3.8](#).

28 The device signals if it is in pairing mode or not. If the device is “not in pairing mode,” the
29 vehicle should continue only if the vehicle is in pairing mode itself.

30 This command shall return a general status word as listed in [Table 5-2](#).

1 5.1.2 SPAKE2+ REQUEST Command

2 5.1.2.1 Purpose

3 In this command, the vehicle shall send the agreed vehicle – owner device framework version list
4 with agreed vehicle – owner device framework (V-OD-FW-agreedVersion) first, all supported
5 Digital Key applet protocol versions, and the Scrypt parameters of the SPAKE2+ protocol. The
6 vehicle shall retrieve the curve point X of the SPAKE2+ protocol in the response. The
7 parameters used in the SPAKE2+ REQUEST command are described in Section [18](#).

8 5.1.2.2 Definition

9 **command: 80 30 00 00 Lc [Table 5-4] 00**

10 **response: [Table 5-5] 90 00**

11 *Table 5-4: SPAKE2+ REQUEST Command Fields*

Tag	Length (bytes)	Description	Field is
5B _h	2 × n	Vehicle - owner device framework version list (V-OD-FW-vehicleList) (ver.high ver.low), agreed version first	mandatory
5C _h	2 × m	m supported Digital Key applet transaction (V-D-TX) protocol versions (ver.high ver.low), agreed version first.	mandatory
5E _h	2 × w	w supported BLE (V-D-BT) Supported_DK_Protocol_Version versions (ver.high ver.low)	Conditional. For WCC2/WCC3 capable vehicles. Shall be included if and only if agreed version in tag 5B _h is greater than 1.0 ²
7F50 _h	32	Scrypt configuration parameters	mandatory
C0 _h	16	Cryptographic salt, s	mandatory
C1 _h	4	Scrypt cost parameter, N _{scrypt}	mandatory
C2 _h	2	Block size parameter, r	mandatory
C3 _h	2	Parallelization parameter, p	mandatory
D6 _h	2	Vehicle Brand (see Table 2-1 in [35]), include this tag for all vehicles including vehicle that supports NFC only.	mandatory ³

12 *Table 5-5: SPAKE2+ REQUEST Response Fields*

Tag	Length (bytes)	Description	Field is
50 _h	65	Curve point X of the SPAKE2+ protocol, prepended with 04 _h as per Listing 18-3	mandatory

² The introduction of versioning for Bluetooth subsystem (WCC2) is gated by V-OD-FW version 1.0

³ Vehicle models launched before 01-01-2021 may not support this. Vehicle models launched after 01-01-2021, including vehicle models only supporting NFC shall support this.

Tag	Length (bytes)	Description	Field is
5F _h	2 or 0	DK (V-D-BT) Protocol Version supported by device from Vehicle List, (as per Table 19-11). Tag with length 0 is returned if no commonly supported version is found.	Conditional. For WCC2/WCC3 capable devices. Present if Tag 5E _h was included in SPAKE2+ REQUEST message

- 1 5.1.2.3 Usage
- 2 Before sending the SPAKE2+ REQUEST command, the vehicle shall check the counter for SPAKE2+ pairing attempts. If the counter indicates 7 pairing attempts have been made, the vehicle shall not send the SPAKE2+ REQUEST command. Instead, the vehicle shall send an OP CONTROL FLOW to abort (error counter limit for wrong pairing password is exceeded, new pairing password needed (0D_h) or no specific reason(00_h)). When a new pairing password is generated, the counter shall be reset.
- 3 As described in section 2.10.6, the vehicle shall send the V-OD-FW version list with agreed V-OD-FW version, to be used by the owner device.
- 4 The vehicle shall also send the list of supported Digital Key applet protocol versions, whereby the first listed version shall be used by the vehicle as the agreed version with the owner device.
- 5 The whole list of Digital Key applet protocol versions (Tag 5C_h) shall be included in the Key Creation Request (See Section 11.8.1, DIGITAL_KEY_APPLET_PROTOCOL_VERSION).
- 6 This allows the determination of the best version to be used for key sharing with friend device.
- 7 Further, this allows the friend device to determine the best transaction (fast transaction, standard transaction) version with the vehicle (V-D-TX).
- 8 The WCC2/WCC3 capable vehicle shall send the version list (tag 5E_h) to the device in the SPAKE2+ Request. The device sends the selected version, to the vehicle in tag 5F_h. The set of V-D-BT version lists (tag 5E_h) obtained from the vehicle shall also be included in the Key Creation Request (see Section 11.8.1). This is done because owner pairing may be first done over the NFC link and Bluetooth activation happens later. If a commonly supported BT protocol version is not found, then the device shall return empty Tag 5F_h with length 0. If the owner device does not support Bluetooth, but a friend device does, the domain version V-D-BT becomes applicable. The process of selection of V-D-BT at the time of Bluetooth activation is shown in Figure 19-2. The version selection is described in detail in Section 19.2.1.7.
- 9 Reception of at least 16 versions shall be supported by all entities. Transmitters may transmit up to 16 versions.
- 10 Version h.l is represented as ver.high='h' (one byte) and ver.low='l' (one byte).
- 11 The first version in the list shall be the agreed version value, the following version values shall be ordered from the highest to the lowest. At least one version shall be provided; otherwise the service is considered as not available.
- 12 The vehicle choice for the Digital Key applet protocol version is provided as part of the Digital Key applet transaction (see Section 15).

All curve points of the SPAKE2+ protocol are defined following x9.63 standard [22] as the byte stream $0x04 \parallel \langle x \rangle \parallel \langle y \rangle$ where $\langle x \rangle$ and $\langle y \rangle$ are 32-byte integers in big-endian representation (see Section 18.1).

If the returned X value is at infinity or is not a valid point on the defined ECC curve, the vehicle shall abort the flow and shall send an OP CONTROL FLOW command with P2 value set to 0C_h as described in Section 5.1.7.

The number of Scrypt iterations (cost parameters) is a positive 4-byte unsigned integer value that configures the Scrypt function (see Section [18.4](#)) to derive the verifier values in the Vehicle OEM Server and on the device. Other transmitted Scrypt parameters are the block size and the parallelization parameters (see Section [18.1.2](#)). The value part of the Scrypt cost parameter TLV, Block size parameter TLV, and Parallelization TLV is encoded in big-endian format.

If the vehicle does not find any version supported by both sides for either SPAKE2+ or the Digital Key applet protocol or both, the vehicle shall send an OP_FLOW_CONTROL (“Owner Pairing Flow Control”) command with an appropriate reason code as defined in [Table 5-24](#), instead of the SPAKE2+ REQUEST command.

If the SPAKE2+ REQUEST command is successfully processed, the vehicle and device shall calculate the shared secret K as per Listing 18-4 and Listing 18-5, respectively.

This SPAKE2+ REQUEST command may return either a general error condition as listed in [Table 5-2](#) or one of the error conditions listed in [Table 5-6](#). No response data shall be returned with an error status word.

Table 5-6: SPAKE2+_REQUEST Response Error Status Words

Status Word	Comment
6985 _h	Command used out of sequence
6A88 _h	Received data invalid or zero
9484 _h	Device not ready for pairing

5.1.3 SPAKE2+ VERIFY Command

5.1.3.1 Purpose

This command mutually exchanges evidence to prove that the calculated shared secret is equal on both sides.

5.1.3.2 Definition

command: 80 32 00 00 Lc [Table 5-7] 00
response: [Table 5-8] 90 00

Table 5-7: SPAKE2+ VERIFY Command Fields

Tag	Length (bytes)	Description	Field is
52 _h	65	Curve point Y of the SPAKE2+ protocol, prepended with 04 _h as per Listing 18-2	mandatory

Tag	Length (bytes)	Description	Field is
57 _h	16	Vehicle evidence M[1]	mandatory

1 Table 5-8: SPAKE2+ VERIFY Response Fields

Tag	Length (bytes)	Description	Field is
58 _h	16	Device evidence M[2]	mandatory

2 5.1.3.3 Usage

3 Before sending the SPAKE2+ VERIFY command, the vehicle shall increase the counter for
4 SPAKE2+ pairing attempts.

5 The vehicle shall calculate evidence M[1] as described in [Listing 18-7](#) and send it, together with
6 curve point Y, to the device.

7 The device shall verify the following:

- 8 1. if the received curve point Y is a valid point on the defined ECC curve
- 9 2. the received M[1]

10 If both verifications are successful, the device shall calculate evidence M[2] as described in
11 [Listing 18-8](#) and shall send M[2] back to the vehicle in SPAKE2+ Verify Response.

12 Only if the vehicle successfully verifies the received M[2], the vehicle shall continue the owner
13 pairing flow.

14 If any of the above verification fails, i.e., an error occurs, the device shall not calculate M[2] and
15 shall not return M[2] or any other response data except the status word. In this case, the vehicle
16 shall send an OP CONTROL FLOW command to abort the owner pairing with P2 value set to
17 09_h as described in Section [5.1.7](#), instead of the next regular command.

18 The SPAKE2+ VERIFY command may return either a general error condition as listed in [Table](#)
19 [5-2](#) or one of the error conditions listed in [Table 5-9](#).

20 Table 5-9: SPAKE2+ VERIFY Response Error Status Words

Status Word	Comment
6985 _h	Command used out of sequence
6A88 _h	Verification of evidence failed

21 The SPAKE2+ VERIFY step leads to secure channel keys used to establish a SCP03 channel
22 between the Framework and the vehicle for the subsequent command exchanges. The used
23 SCP03 channel is established following [Listing 18-10](#) and [Listing 18-11](#). The created secure
24 channel keys shall be reset under the following conditions:

- 25 • After successful owner pairing
- 26 • When device returns a response other than 9000_h or 61XX_h (see [Table 5-15](#))
- 27 • A SPAKE2+ REQUEST is sent
- 28 • When a tearing of communication occurs
- 29 • When the maximum allowed time has expired without successfully terminating owner
30 pairing

1 The vehicle needs to start with SPAKE2+ again to establish new keys. Note that the pairing
2 password shall remain the same in those cases. The vehicle shall rotate the pairing password after
3 seven failed attempts.

4 **5.1.4 WRITE DATA Command**

5 **5.1.4.1 Purpose**

6 This command sends all data needed to generate the Digital Key to the device. It is also used to
7 provide an attestation from the vehicle of the device public key (PK) for key tracking purposes.

8 **5.1.4.2 Definition**

9 The following WRITE DATA command structure shall be used:

10 **command: 84 D4 P1 00 Lc [command_data] [command_mac] 00**

11 **response: [response_mac] 90 00**

12 Parameters P1 and P2 are always set to 00, except for the last WRITE DATA command, in
13 which P1 shall be set to 80h.

14 The command shall only be allowed to be sent in a secure channel.

15 The command may return either a general error condition as listed in [Table 5-2](#) or one of the
16 error conditions listed in [Table 5-10](#).

17 *Table 5-10: WRITE DATA Response Error Status Words*

Status Word	Comment
6985 _h	Command used out of sequence
6A84 _h	Not enough memory

18 **5.1.4.3 Usage**

19 One or more WRITE DATA commands may be used to write the requested data objects into a
20 buffer in the device.

21 *Table 5-11: Objects for Digital Key Creation*

Tag	Length (bytes)	Data Content	Field is
7F4A _h	variable	Endpoint creation data, elements from Table 15-13, except the fields listed in the text below	mandatory
7F4B _h	variable	DER-encoded X.509 Vehicle Public Key Certificate [K], Listing 5-3	mandatory
7F4C _h	variable	DER-encoded X.509 Intermediate Certificate	optional
7F4D _h	variable	Mailbox Mapping Table 5-13	mandatory
7F4E _h	variable	Device Configuration Table 5-14	mandatory
5F5F _h	0	Completion of Digital Key Data Sending	mandatory

1

Table 5-12: Objects for Device Digital Key Certificate

Tag	Length (bytes)	Data Content	Field is
5F5A _h	variable	Attestation of the device PK by the vehicle (opaque)	optional
5F5F _h	0	Completion of Digital Key Certificate Sending	mandatory

- 2 All required objects of [Table 5-11](#) shall be written to the device in the order given in the table.
 3 One or several TLVs are allowed to be written per (sequence of) WRITE DATA command(s).
 4 The TLV 5F5F_h shall be written at last to mark the end of the transmitted data sequence. The last
 5 WRITE DATA command of this sequence, which contains TLV 5F5F_h, shall be indicated by
 6 setting P1=80_h.
 7 All required objects of [Table 5-12](#) shall be written to the device when the vehicle has accepted
 8 the device PK. The last WRITE DATA command of this sequence shall be indicated by setting
 9 P1=80_h.
 10 5F5F_h shall be the last TLV written to mark the end of the transmitted data sequence.
 11 Tag 7F4A_h shall contain all data elements from Table 15-13(not including the outer tag 7F27_h,
 12 only the inner tag value 4D_h, 46_h, etc.), except for the following elements:
 13 • Endpoint identifier (defined by device)
 14 • Instance CA identifier (defined by device)
 15 • Counter limit (deprecated, shall not be used)
 16 If the vehicle_identifier is provided as part of Tag 7F4A_h, the device shall compare it to the value
 17 provided in the Vehicle Public Leaf Certificate [K] and abort the owner pairing if the check fails.
 18 In this version of the specification, the vehicle shall include only one single authorized_PK
 19 (inner Tag 49_h). This authorized_PK shall correspond to the Vehicle intermediate CA or the
 20 Vehicle OEM CA PK (root).
 21
 22 The maximum command data length shall be 239 bytes: len = [command_data] + [padding] +
 23 [command_mac]
 24 len = 239 + 1 + 8 ≤ 255 (ok)
 25 len = 240 + 16 + 8 > 255 (not ok)
 26 [command_data] + [padding] shall be a multiple of the AES block size (16 bytes). The
 27 padding scheme is described in [\[9\]](#). At least one byte padding (80_h) shall be required. The
 28 maximum response data length shall be 239 bytes.
 29 The vehicle public key shall be provided as a X.509 certificate signed by the Vehicle OEM CA
 30 as described in [Listing 5-3](#).
 31 The X.509 [\[3\]](#) definition of the basic constraints extension is described in [Listing A-5](#).
 32 The X.509 [\[3\]](#) definition of the key usage extension is described in [Listing A-4](#).

33

Listing 5-1: Vehicle Certificate Extension Schema

```

1 VehicleCertificateExtensionSchema ::= SEQUENCE
2 {
3   extension_version INTEGER (1..255)
4 }
```

1 *Listing 5-2: Vehicle Certificate Extension Data*

```
1 vehicle-cert-extension-data VehicleCertificateExtensionSchema ::=  
2 {  
3   extension_version 1 --value shall be 1  
4 }
```

2 The Vehicle Public Key Certificate data is described in [Listing 5-3](#). See [3] for details on X.509
3 v3 certificate format.

4 *Listing 5-3: Vehicle Public Key Certificate Data*

```
1 vehicle-key-cert-data Certificate ::=  
2 {  
3   tbsCertificate  
4   {  
5     version v3, --shall be v3--  
6     serialNumber ..., --a random integer chosen by the certificate issuer,  
7     Signature  
8     {  
9       algorithm {1 2 840 10045 4 3 2} --OID for ecdsaWithSHA256 (ANSI X9.62 ECDSA algorithm with SHA-256)  
10      },  
11      issuer rdnSequence:  
12      {  
13        {  
14          {  
15            type {2 5 4 3}, --OID for CommonName  
16            value "..." --shall match the subject of the issuing certificate, shall use PrintableString or UTF8String format  
17          }  
18        }  
19      },  
20    },  
21    validity  
22    {  
23      notBefore Time: "..." --shall use UTCTime or GeneralizedTime as defined in [3]  
24      notAfter Time: "..." --shall use UTCTime or GeneralizedTime as defined in [3]  
25    },  
26    subject rdnSequence:  
27    {  
28      {  
29        {  
30          type {2 5 4 3}, --OID for CommonName  
31          value "Vehicle OEM Identifier" --contains the subject of the certificate, as per Appendix B.2.6  
32          shall use PrintableString or UTF8String format  
33        }  
34      }  
35    },  
36    subjectPublicKeyInfo  
37    {  
38      algorithm  
39      {  
40        algorithm {1 2 840 10045 2 1} --OID for ecPublicKey (ANSI X9.62 public key type)  
41        parameters {1 2 840 10045 3 1 7} --OID for prime256v1(ANSI X9.62 named elliptic curve)  
42      },  
43      subjectPublicKey '04...H --the public key pre-pended with 04h to indicate uncompressed format  
44    },  
45    extensions  
46    {  
47      {
```

```

48     extnID {1.3.6.1.4.1.41577.5.1}, --OID for Vehicle Public Key Certificate (see Appendix B.2.2)
49     critical TRUE,
50     extnValue '...'H --DER encoding for VehicleCertificateExtensionSchema extension as per Listing 5-1
51 },
52 {
53     extnID {2 5 29 15}, --KeyUsage standard extension
54     critical TRUE,
55     extnValue '03020780'H --DER encoding for KeyUsage, digitalSignature only
56 },
57 {
58     extnID {2 5 29 19}, --BasicConstraints standard extension
59     critical TRUE,
60     extnValue '3000'H -- DER encoding for cA=FALSE
61 },
62 {
63     extnID {2 5 29 35}, --OID for AuthorityKeyIdentifier standard extension
64     critical FALSE,
65     extnValue '...'H- DER encoding of an AuthorityKeyIdentifier sequence, containing only a KeyIdentifier element.
66 -- The KeyIdentifier is an OCTET STRING containing the 160-bit SHA-1 hash of the value of the BIT STRING
subjectPublicKey
67             --from the issuer certificate (excluding the tag, length, and number of unused bits)
68 },
69 {
70     extnID {2 5 29 14}, -- OID for SubjectKeyIdentifier standard extension
71     critical FALSE,
72     extnValue '...'H --160-bit SHA1 hash of the value of the BIT STRING subjectPublicKey
           --(excluding the tag, length, and number of unused bits)
73 }
74 },
75 signatureAlgorithm
76 {
77     algorithm {1 2 840 10045 4 3 2}
78 },
79 signatureValue '...'H --the certificate signature computed as per \[3\]
80 --ECDSA signature
81 }

```

- 1 The subject field is formatted as described in Appendix [B.2.3](#)
- 2 The ROUTING_INFORMATION (as used in [11.5](#)) is defined in the common name of the
- 3 subject field of [Listing 5-3](#), i.e., the subject ID of the Vehicle Public Key Certificate [K]
- 4 (Appendix B.2.6).

5 *Table 5-13: Mailbox Mapping*

Tag	Length (bytes)	Description	Field is
7F4D _h	variable	Mailbox mapping	mandatory
D0 _h	2	SIGNALING_BITMAP_OFFSET as per Table 4-2	mandatory
D1 _h	2	MAILBOX_CONTENT_END_OFFSET as per Table 4-2	mandatory
D2 _h	1	IMMOBILIZER_TOKEN_LENGTH as per Table 4-3	optional
C0 _h	2	Sig-Bit 0: corresponding data offset as per Table 4-2	mandatory
C1 _h	2	Sig-Bit 1: corresponding data offset as per Table 4-2	conditional

Tag	Length (bytes)	Description	Field is
C2 _h	2	Sig-Bit 2: corresponding data offset as per Table 4-2	conditional
C3 _h	2	Sig-Bit 3: corresponding data offset as per Table 4-2	conditional
C4 _h	2	Sig-Bit 4: corresponding data offset as per Table 4-2	conditional
C5 _h	2	Sig-Bit 5: corresponding data offset as per Table 4-2	conditional
C6 _h	2	Sig-Bit 6: corresponding data offset as per Table 4-2	conditional
C7 _h	2	Sig-Bit 7: corresponding data offset as per Table 4-2	conditional

1 Conditional fields shall be present when the data element designated by the corresponding Sig-
2 Bit is used.

3 [Table 5-14](#) provides all supplementary configuration information to the device.

4 *Table 5-14: Device Configuration Data*

Tag	Length (bytes)	Description	Field is
7F4E _h	variable	Device configuration	mandatory
7F49 _h	variable	7F49 template is used to provide wireless capability (NFC/UWB) and related data; this template shall be included by vehicle. If this tag is not present, only NFC is supported. (equivalent to only 5F50 tag present) See Table 19-84 for details	optional
7F60 _h	variable	SHARING_CONFIGURATION (for key sharing, see Section 11)	optional
DA _h	1	If this tag is not present, the immobilizer token and slot identifier are retrieved from vehicle. If this tag is present, the following values are assigned: 00 _h immobilizer token and slot identifiers are retrieved from vehicle 01 _h immobilizer token and slot identifiers are retrieved online 02 _h no immobilizer token, and slot identifiers are retrieved from vehicle 03 _h no immobilizer token, and slot identifiers are retrieved online 04 _h immobilizer token and slot identifiers for owner are retrieved from vehicle and friend immobilizer token and slot identifiers are retrieved online. If this setting is used during owner pairing, the owner device shall use 01 _h for friend sharing. 05 _h no immobilizer token but with slot ID retrieved from vehicle for owner device; no immobilizer token but with slot ID retrieved online for friend device. If this setting is used during owner pairing the owner device shall use 03 _h for friend sharing.	optional

Tag	Length (bytes)	Description	Field is
DB _h	0	If this tag is present, owner and friend devices have to obtain a key tracking receipt (owner during Phase 4 as per Section 6.3.5 , and friend during Step 7 as per Section 11.8.3.1)	optional
DC _h	0	If this tag is present, vehicle manufacturer supports online certificate delivery feature for key sharing as described in Section 9 .	optional
D7 _h	1	2nd Factor activation required (00 _h =not required, 01 _h =required for non-approved sharing methods, other=RFU)	optional
D8 _h	1	SHARING_PASSWORD_LENGTH (vehicle OEM specific as per appendix B.1) If absent, sharing password is not required.	optional
D9 _h	variable	Supported Digital Key profile list as described in Section 11	optional
Tag 4A _h and 4B _h of Table 15-52 (SETUP ENDPOINT Command Payload)			optional

- 1 If tag 7F60_h is absent, the default SHARING_CONFIGURATION is
 - 2 • DA_h: 00_h
 - 3 • DB_h: present
 - 4 • DC_h: not present
- 5 If tag D7_h is absent, then no sharing password is required.
- 6 **Note:** Information in Tag D7_h could be overwritten by the presence of the activationRequired field in the key tracking response and/or eventNotification.
- 7 **Note:** Information in Tag D8_h could be overwritten by the presence of the sharingPasswordLength field in activationOptions of the uiBundle sent in the key tracking response and/or eventNotification (Section [17.8.19](#))
- 8 The LONG_TERM_SHARED_SECRET shall be used to generate the password used for key sharing (see LONG_TERM_SHARED_SECRET in Section [11](#)). The LONG_TERM_SHARED_SECRET shall be derived from the SPAKE2+ shared secret (K).
- 9 The sharing password length shall be used as a parameter for the derivation algorithm. The supported key profiles list defines the key profiles for key sharing that are supported by the vehicle. The profiles are defined as a list of 1-byte enumeration values corresponding to Table 11-4.
- 10 If neither tag 4A_h nor 4B_h is included in the Device Configuration Data sent by vehicle to device using WRITE DATA command (see step 3 in Figure 6-3), then device shall transmit no data in the AUTH1 response to the vehicle (see step 6 in Figure 6-10 and Section [6.3.4.2](#)).
- 11 If tag 4A_h or 4B_h or both are included in the Device Configuration Data sent by vehicle to device using WRITE DATA command (see step 3 in Figure 6-3), then device shall transmit the corresponding data in the AUTH1 response to the vehicle (see step 6 in Figure 6-8).

1 5.1.5 *GET DATA Command*

2 5.1.5.1 *Purpose*

3 This command shall continue to use the established session keys to retrieve all data needed to
4 verify the Digital Key created by the Digital Key framework in the Digital Key applet instance.

5 5.1.5.2 *Definition*

6 **command: 84 CA 00 00 Lc [encrypted_tag] [command_mac] 00**

7 **response: [response_payload] [response_mac] 90 00 or 61XX**

8 Only one tag shall be requested per GET DATA command. The tags may be requested in
9 arbitrary order. Elements may be omitted if they are not required.

10 X.509 certificates shall not be wrapped into additional TLV structures, but tags shall be used to
11 refer to the certificates in the command tag list, as indicated below.

12 Le shall be set to 00_h as the exact length of data depends on the variable certificate field sizes.

13 GET RESPONSE shall be used if the response status word indicates that more data is to be
14 retrieved.

15 This command may return either a general error condition as listed in Table 5-2 or one of the
16 error conditions listed in Table 5-15.

17 *Table 5-15: GET DATA Response Error Status Words*

Status Word	Comment
61XX _h	XX indicates the number of response bytes still available (use GET RESPONSE)
6985 _h	Command used out of sequence
6A88 _h	Reference data not found
6400 _h	Digital Key could not be created
6402 _h	Requested elements not available

18 5.1.5.3 *Usage*

19 The command requests the transfer of the required data element by providing the corresponding
20 tag. The response data length shall not exceed 239 bytes, as the maximum length of an APDU
21 response shall not exceed 258 bytes (including status word):

22 len = [response_data] + [padding] + [response_mac] + [status_word]

23 len = 239 + 1 + 8 + 2 ≤ 258 (ok)

24 len = 240 + 16 + 8 + 2 > 258 (not ok)

25 [response_data] + [padding] equals a multiple of the AES block size (16 bytes). At least one byte
26 padding (80_h) is required. The maximum response data length is then 239 bytes.

27 See Section 18.4.12 for calculation of [command_mac] and [response_mac].

28 Each element shall be requested in a separate GET DATA command. GET RESPONSE shall be
29 used if the returned data exceeds the GET DATA response data maximum length.

30 When the response is a TLV structure (e.g., certificate) then the response shall not be wrapped
31 in the requested tag. Otherwise (e.g., string), the response shall be wrapped in the requested tag.

- 1 If all the remaining bytes cannot be sent in the current response, the status word 61XX_h shall be
2 used instead of 9000_h.
3 If XX is between 01_h and EF_h, XX number of bytes (1..239) are still available and shall be sent in
4 the next response (which shall be the last one). Note that XX does not include padding length.
5 If XX equals 00_h, not all remaining bytes can be sent in the next response.
6 Values between F0_h and FF_h are not allowed for XX.

7 *Table 5-16: GET DATA Response Decrypted Payload for tag 7F20_h*

Description
Vehicle OEM signed Device OEM CA Certificate (DeviceOEM.PK.VehicleOEMCert), [F] as per Listing 15-12

8 *Table 5-17: GET DATA Response Decrypted Payload for tag 7F22_h*

Description
Device OEM signed Instance CA Certificate, [E] as per Listing 15-15

9 *Table 5-18: GET DATA Response Decrypted Payload for tag 7F24_h*

Description
Digital Key Certificate signed by Instance CA (DigitalKey.DKCert) as per Listing 15-5

10 *Table 5-19: GET DATA Response Decrypted Payload for Tag D3_h*

Tag	Length (bytes)	Description
D3 _h	1–30	Friendly name of the owner key (UTF-8) with length of 1 to 30 bytes

11 The friendly name of the owner Digital Key should be assigned during owner pairing. The
12 friendly name shall be a UTF-8 encoded string. The owner should be allowed to edit the name
13 for identification and personalization. For privacy reasons, the friendly name should not contain
14 private information, such as the full name of the owner.

15 5.1.6 GET RESPONSE Command

16 5.1.6.1 Purpose

17 This command shall only be used to retrieve remaining data that could not be fully transmitted in
18 the GET DATA response. This command shall only be accepted if the GET DATA response
19 received previously was with a status word
20 61XX_h

21 5.1.6.2 Definition

22 **command: 84 C0 00 00 Lc [command_mac] 00**

23 **response: [response_payload] [response_mac] 90 00 or 61XX**

24 When the response data is not fully transmitted in the response payload of GET RESPONSE, a
25 status word 61XX_h shall be returned. XX_h indicates the number of response bytes still available.
26 When a response data is completely transmitted in the response payload of GET RESPONSE and
27 no more remaining data is to be returned, a status word 9000_h is returned.

1 This command may return either a general error condition as listed in Table 5-2 or one of the
2 error conditions listed in [Table 5-20](#).

3 *Table 5-20: GET RESPONSE Response Error Status Words*

Status Word	Comment
6985 _h	Command used out of sequence

4 **5.1.6.3 Usage**

5 One or more GET RESPONSE commands shall only be used to retrieve all response data that
6 could not be fully transmitted in the GET DATA response.

7 The command shall only be accepted when sent immediately after a GET DATA or GET
8 RESPONSE response with the status word 61XX_h.

9 The vehicle shall send the last GET RESPONSE command when a GET RESPONSE response
10 with XX of 1..239 is received. Values between F0_h and FF_h are not allowed for XX. When XX is
11 00_h, at least one more GET RESPONSE command shall follow.

12 **5.1.7 OP CONTROL FLOW Command**

13 **5.1.7.1 Purpose**

14 With this command, the vehicle may control the flow at predefined points and abort the flow at
15 any time while giving a status to the device.

16 “Continue” and “End” shall be used where specified in the flow diagrams. Status information
17 about the state, successful or unsuccessful end, etc. shall be provided.

18 **5.1.7.2 Definition**

19 This command is always sent outside of the secure channel.

20 **command: 80 3C [Table 5-21][Table 5-22 or Table 5-23 or Table 5-24]**

21 **response: 90 00**

22 This command may return a general error condition, as listed in Table 5-2.

23 **5.1.7.3 Usage**

24 The vehicle shall not consider the session as terminated before it has received the response to the
25 OP CONTROL FLOW command (applicable for P1=11_h or P1=12_h).

26 The OP CONTROL FLOW response should be kept as short as possible to minimize the risk of
27 false positives when a tearing happens on the response.

28 When aborted, the vehicle shall perform the NFC link teardown procedure, followed by the NFC
29 reset procedure, before it restarts owner pairing or any other transaction.

30 P1 indicates the main action triggered by the command; P2 provides the reason code.

31 The Digital Key Framework shall ignore unknown P1/P2 values, if a P1 value is not known it
32 shall be treated as P1=10_h (continue flow) and ignored.

1

Table 5-21: OP CONTROL FLOW P1 Parameters

P1 Value	Description
10 _h	continue flow, see reason code for details
11 _h	end flow, see reason code for details
12 _h	abort flow, see reason code for details

2

Table 5-22: OP CONTROL FLOW P2 Parameters for P1=10_h (continue)

P2 Value	Description
01 _h	key creation data transmitted to device
02 _h	key certificate chain received by vehicle
03 _h	Waiting for user confirmation on vehicle UI
0F _h	Vehicle waiting time extension, keep busy, no action on device side

3

Table 5-23: OP CONTROL FLOW P2 Parameters for P1=11 (end with success)

P2 Value	Description
11 _h	successful end of key creation and verification, key not tracked by vehicle

4

Table 5-24: OP CONTROL FLOW P2 Parameters for P1=12 (end with failure)

P2 Value	Description
00 _h	no specific reason
01 _h	no matching SPAKE2+ version found between vehicle and device
02 _h	no matching Digital Key applet protocol version found between vehicle and device
03 _h	pairing failed due to timeout of timer 1
04 _h	pairing failed due to timeout of timer 2
05 _h	pairing failed due to timeout of timer 3
06 _h	pairing failed due to timeout of timer 4
07 _h	pairing failed due to timeout of timer x (spare, not used)
08 _h	preconditions for owner pairing not fulfilled
09 _h	evidence verification on vehicle side failed
0A _h	wrong Digital Key configuration
0B _h	certificate verification failed
0C _h	curve point X zero or invalid
0D _h	error counter limit for wrong pairing password is exceeded, new pairing password needed
7F _h	pairing failed due to response data or format error of previous command

5 The option to end the flow refers to the regular end of the transaction and the possible outcomes (successful, not successful) and the UI actions.

6 The option to abort the flow refers to the vehicle aborting the flow before its regular end due to unexpected or erroneous behavior.

1 5.2 Data Elements

2 5.2.1 Certificates

3 *Table 5-25: Certificates*

Tag	Certificate	Reference
7F20 _h	Device OEM Certificate signed by Vehicle OEM CA	Listing 15-12
7F22 _h	Instance CA Certificate signed by Device OEM CA	Listing 15-15
7F24 _h	Digital Key Certificate signed by Instance CA	Listing 15-5
7F4B _h	Vehicle Public Key Certificate	Listing 5-3
7F4C _h	Vehicle OEM Intermediate Certificate	optional, (Vehicle OEM specific)

- 4 The Vehicle OEM intermediate certificate may be required by the Vehicle OEM if the Vehicle OEM CA is not able to sign the Vehicle Public Key Certificate directly.

1 6 OWNER PAIRING

2 6.1 Overview

3 The owner pairing flow is operated by the Digital Key framework running on the device. As
4 shown in [Figure 2-2](#), the Digital Key framework uses the APDU commands described in Section
5 [5](#) to manage the configuration of the Digital Key, protected by the SE.

6 The SE operating system provides the root of trust, which is the starting point in the trust chain
7 as defined in Section [16.1](#).

8 The vehicle is able to select the Digital Key applet over NFC using its AID and to select the
9 Digital Key framework using the corresponding AID. The NFC controller may be reconfigured
10 for changing the routing of the communication from the SE to the Digital Key framework and
11 vice versa, based on the selected AID.

12 A new owner device pairing flow or owner device change does not imply an implicit unpairing,
13 i.e., a new device owner pairing flow only changes the owner's key. Existing shared/friend keys
14 that are already paired and vehicle public keys are not necessarily impacted.

15 The Digital Key applet instance shall be available on the SE before the time of owner pairing
16 execution.

17 6.2 Keys and Data

18 This section defines the key and data elements used in owner pairing:

- 19 • **device.PK/device.SK**: Long term key pair generated by device during Digital Key
20 creation. One per Digital Key.
- 21 • **vehicle.PK/vehicle.SK**: Key pair generated by the vehicle. Same for all Digital Keys of
22 the vehicle. The lifecycle of this key pair is defined and managed by the Vehicle OEM
23 and is out of scope of this specification.
- 24 • **Pairing password**: SPAKE2+ pairing password entered into the device; eight numeric
25 digits (0–9) in UTF-8 format provided by the Vehicle OEM account to authenticate the
26 owner.
- 27 • **Kenc**: Derived symmetric key (from SPAKE2+ shared secret), used to encrypt
28 confidential command and response payloads.
- 29 • **Kmac**: Derived symmetric key (from SPAKE2+ shared secret), used to calculate
30 command MACs.
- 31 • **Krmac**: Derived symmetric key (from SPAKE2+ shared secret), used to calculate
32 response MACs.

33 6.3 Owner Pairing Implementation

34 This section describes the phases of the owner pairing flow, which include:

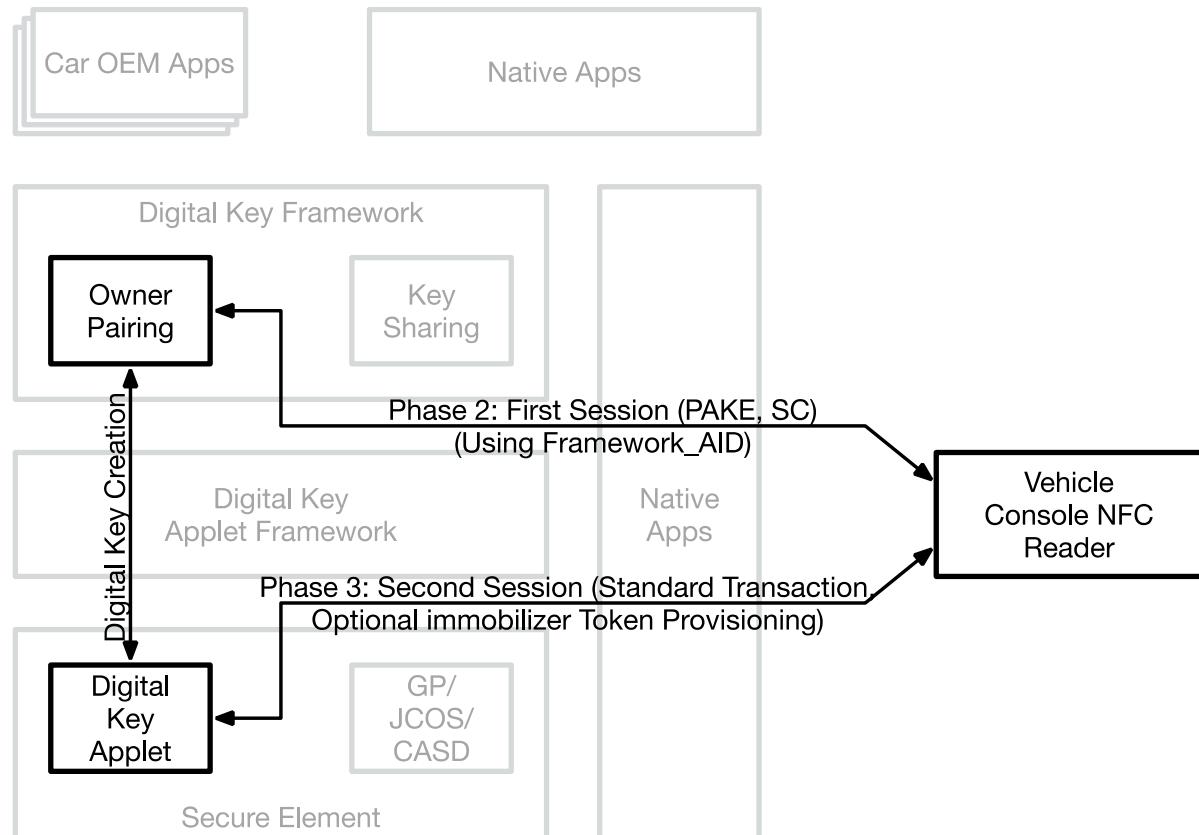
- 35 • **Phase 0**: Preparation (see Section [6.3.1](#))
- 36 • **Phase 1**: Initiate pairing procedure on vehicle and device (see Section [6.3.2](#))
- 37 • **Phase 2**: First session with NFC reader (see Section [6.3.3](#))

- **Phase 3:** Second session with NFC reader (see Section [6.3.4](#))
 - **Phase 4:** Finalization of pairing procedure (see Section [6.3.5](#))

The owner pairing flow consists of two sessions: the first session (Phase 2), executed with the Digital Key framework, and the second session (Phase 3) with the Digital Key applet, as shown in Figure 6-1. In owner pairing Phase 2, the vehicle configures whether the owner device has to retrieve the immobilizer tokens online from the Vehicle OEM Server or from the vehicle. If online retrieval is configured, the owner immobilizer token is not transferred in the second session with the NFC reader but within the Key Tracking Request/Response to/from the Vehicle OEM Server over the owner Device OEM Server during or after the owner pairing. Figure 6-1 describes the owner pairing process whereby the immobilizer tokens are retrieved from the vehicle.

The first session consists of two NFC transactions. The first transaction is to negotiate protocol versions, execute the SPAKE2+, and transmit all key creation data to the device. The second transaction, which is executed after the creation of the Digital Key in the device, provides the creation attestation and certificate chain to the vehicle.

Figure 6-1: Owner Pairing NFC Exchanges



1 6.3.1 *Phase 0: Preparation*

2 6.3.1.1 *Device Preparation*

3 It is assumed that the device has already installed a Digital Key applet and created an Instance
4 CA per Vehicle OEM prior to owner pairing operation. The Digital Key framework is updated
5 with the list of Vehicle OEM partners.

6 The Instance CA Certificates are obtained by the Digital Key framework.

7 See [Listing 15-15](#) for the description of the Instance CA Certificate [E]. All signatures are
8 generated as described in Section [18.4.10](#).

9 6.3.1.2 *Vehicle Provisioning*

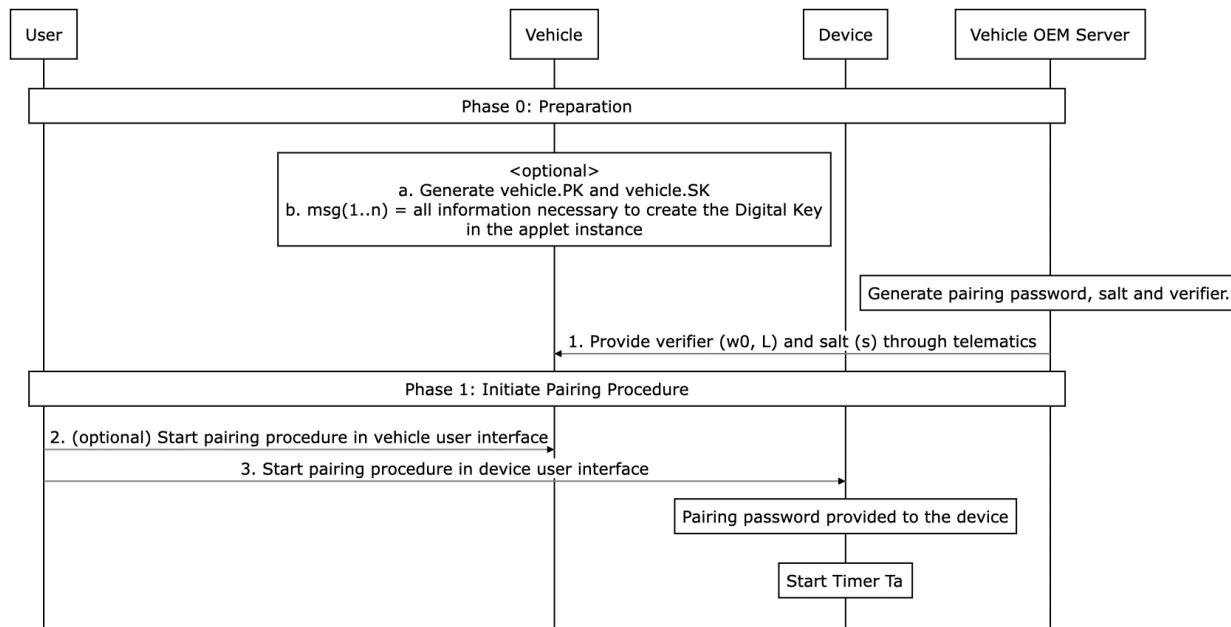
10 The owner pairing secure channel creation is based on the SPAKE2+ protocol, which is
11 described in Section [18.4.1](#). The protocol binds the device to the user and is resistant against
12 eavesdropping and MITM attacks. See [\[10\]](#) for more information about the computational and
13 security aspects of the protocol.

14 Before the owner pairing procedure starts, the Vehicle OEM Server generates the pairing
15 password for the device and the password verifier for the vehicle as described in [Listing 18-1](#),
16 and sends the verifier and salt through a Vehicle OEM proprietary secure channel to the vehicle,
17 as shown in Step 1 of [Figure 6-2](#).

18 6.3.2 *Phase 1: Initiate Pairing Procedure*

19 On the vehicle side, the pairing procedure initiation is under the responsibility of the Vehicle
20 OEM. Appropriate owner pairing preconditions need to be met, e.g., the presence of a key fob.

21 *Figure 6-2: Owner Pairing Flow - Phase 0/1: Preparation/Initiation*



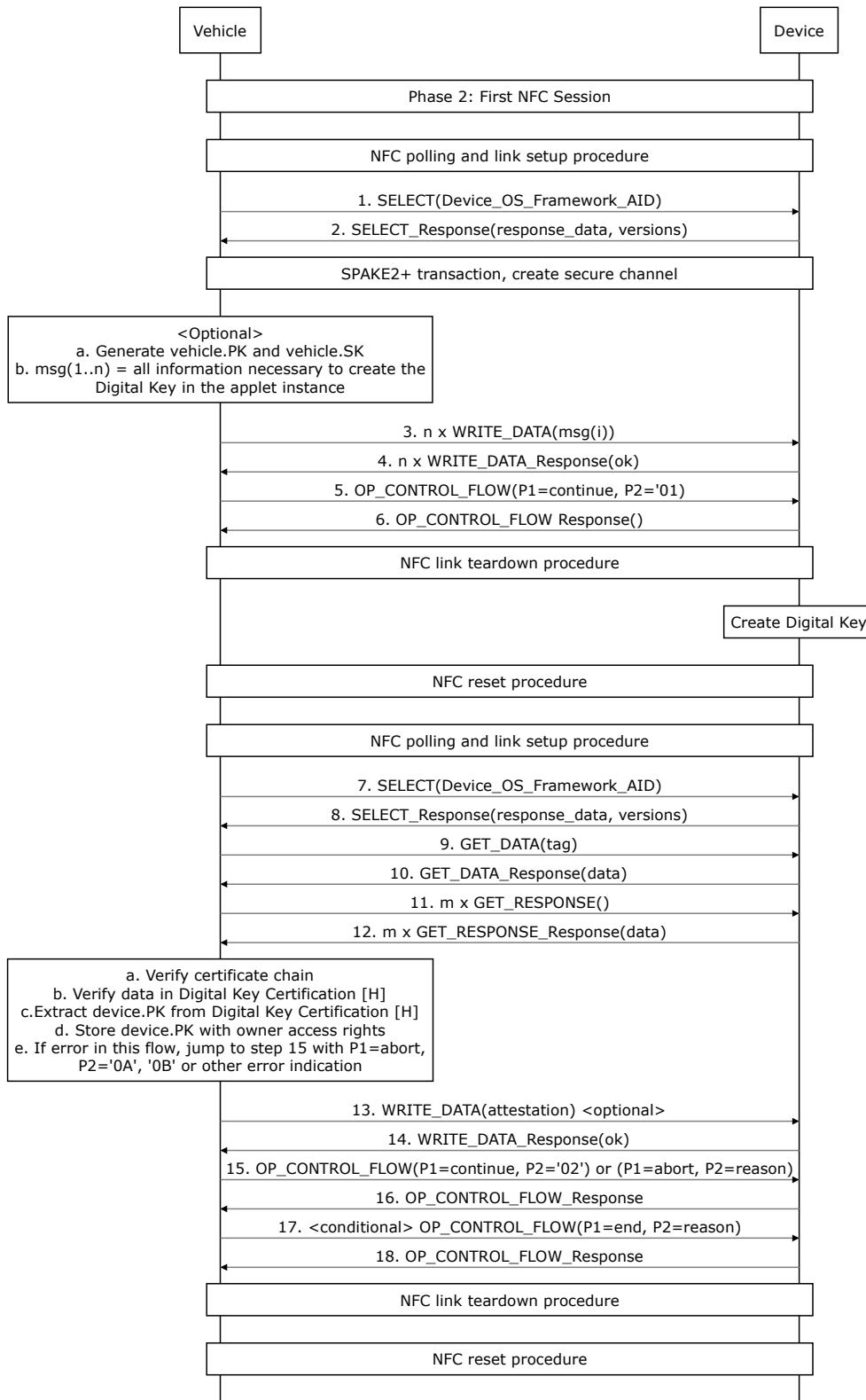
1 The vehicle is either set into pairing mode by the user or tries to select the framework AID on the
2 console NFC reader in the vehicle as long as no owner device is successfully paired.
3 To start owner pairing mode (Step 3 of [Figure 6-2](#)), the device receives the password, either
4 through user input, a URL (see Section [6.3.7](#)), or through an API directly from the Vehicle OEM
5 app, if installed.
6 The device shall turn off the NFC interface after the pairing procedure has been started in the
7 device (step 3 of [Figure 6-2](#)). The NFC link shall be started again after the user has entered the
8 pairing password, or the user interface is dismissed.
9 NOTE: The deactivation of the NFC interface is introduced to stop additional polling due to
10 external influence on the device or vehicle. For example, with activated NFC interface, while
11 waiting for the input of the User (Step 3 of [Figure 6-2](#)), due to movements of the device the
12 vehicle could restart the polling again. This could lead to a restart of the owner pairing while the
13 first one is still running.

14 6.3.3 *Phase 2: First Session with NFC Reader*

15 The first NFC session consists of two distinct NFC transactions.
16 The first transaction negotiates the protocol versions (see Section [6.3.3.8](#)), mutually authenticates
17 both device and vehicle, establishes a secure channel using SPAKE2+, and transmits all key
18 creation data to the device, as shown in Figure 6-3.

1

Figure 6-3: Owner Pairing Flow - Phase 2: First NFC Session



2

- 1 The SPAKE2+ creates a symmetric session key pair used to establish a secure channel between
- 2 vehicle and device.
- 3 Before the second NFC transaction starts, the device creates the Device Key.
- 4 In the second NFC transaction, the vehicle reads the key creation data from the device, verifies it
- 5 and, if successful, stores the device public key. The NFC reset procedure is performed at the end
- 6 of each transaction.
- 7 The following steps or sequence of events takes place in the first NFC session:

8 *6.3.3.1 Step 1: Digital Key Framework Selection*

9 When the device is communicating to the console NFC reader in the vehicle, the vehicle shall
10 select the Digital Key framework using its AID, through the SELECT command. If the vehicle
11 selects the Digital Key Applet AID before the framework AID, then the device shall respond to
12 SELECT Digital Key Applet AID with Status Word (SW) = 6A82_h. The device shall return all
13 supported SPAKE2+ versions and all Digital Key applet protocol versions to the vehicle through
14 SELECT Response command. This determines the SPAKE2+ version (used for creating secure
15 channel) and Digital Key applet protocol version (for key sharing) to be used on both sides.
16 The version information shall be checked for compatibility on the vehicle side before continuing
17 as described in Section [6.3.3.8](#).

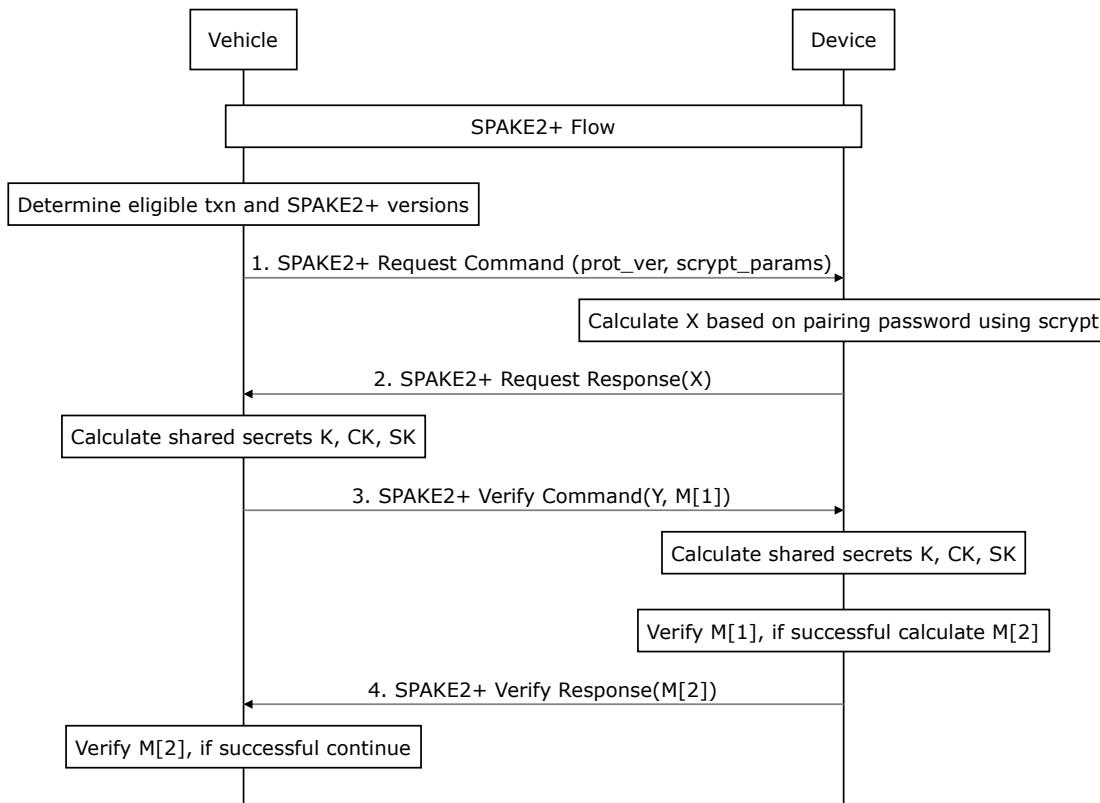
18 The SELECT command is described in Section [5.1.1](#).

19 *6.3.3.2 Step 2 and 2a: SPAKE2+ Transaction*

20 The vehicle shall select and send to the device the SPAKE2+ protocol version to be used and the
21 list of all supported Digital Key applet protocol versions (as described in Section [6.3.3.8](#)).
22 The SPAKE2+ transaction is described in [\[10\]](#).

1

Figure 6-4: SPAKE2+ Flow

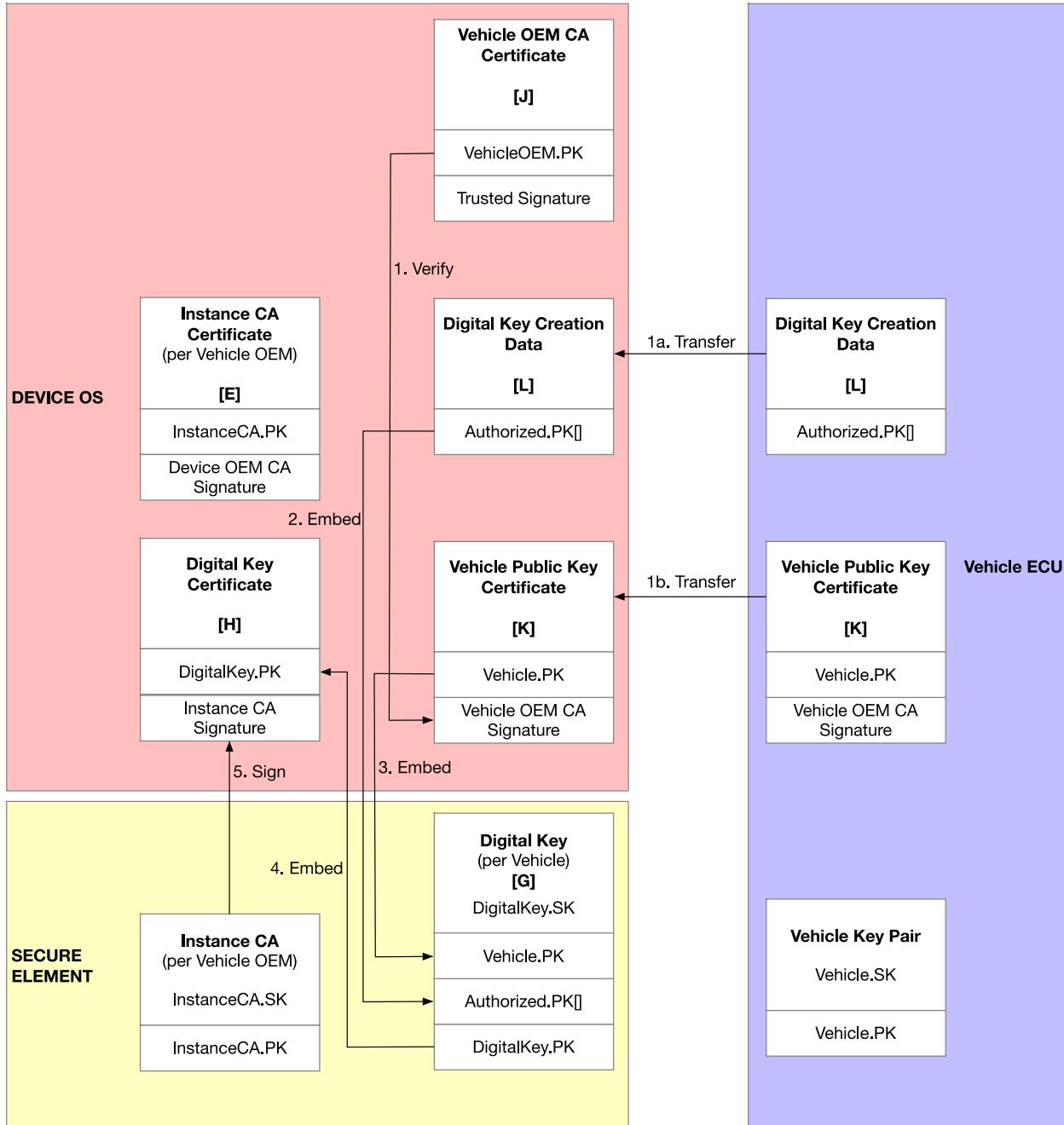


2

- 3 The SPAKE2+ transaction is shown in Figure 6-4. The SPAKE2+ transaction establishes a secure channel for the data exchange between vehicle and device. The secure channel shall remain active across the RF resets.
- 4 The secure channel is terminated when the OP CONTROL FLOW command indicating success is sent (Step 17 of Figure 6-3), when an OP CONTROL FLOW command indicates an error, or after any other abortion of the flow. An incorrect decryption or MAC value on APDU commands also terminates the secure channel.
- 5 When the vehicle is not ready for owner pairing, it shall use the OP CONTROL FLOW to abort and indicate the condition to the user, such as “preconditions for owner pairing not fulfilled.”
- 6 *Note:* Only commands with the appropriate class byte indication (bit 2 = 1) are part of the secure channel.
- 7 **6.3.3.3 Steps 3 to 4: WRITEDATA**
- 8 The vehicle shall send all data required to create a Digital Key through the secure channel to the device using multiple WRITE DATA commands (see [Figure 6-5](#) and [Figure 6-6](#)).
- 9 After successful reception, the device shall verify the Vehicle Public Key Certificate [K] as described in Section [6.3.3.10](#), using one of the following:
- 10 • Vehicle OEM CA Certificate [J] (see [Figure 6-5](#))
- 11 • Vehicle OEM CA Certificate [M] signed by Device OEM CA [D] (see [Figure 6-6](#))

- 1 In case of SE-centric applet model implementation, the Digital Key framework may omit the
2 verification since it is conducted by the Digital Key applet during Endpoint creation as described
3 in Section [15.3.2.4](#).

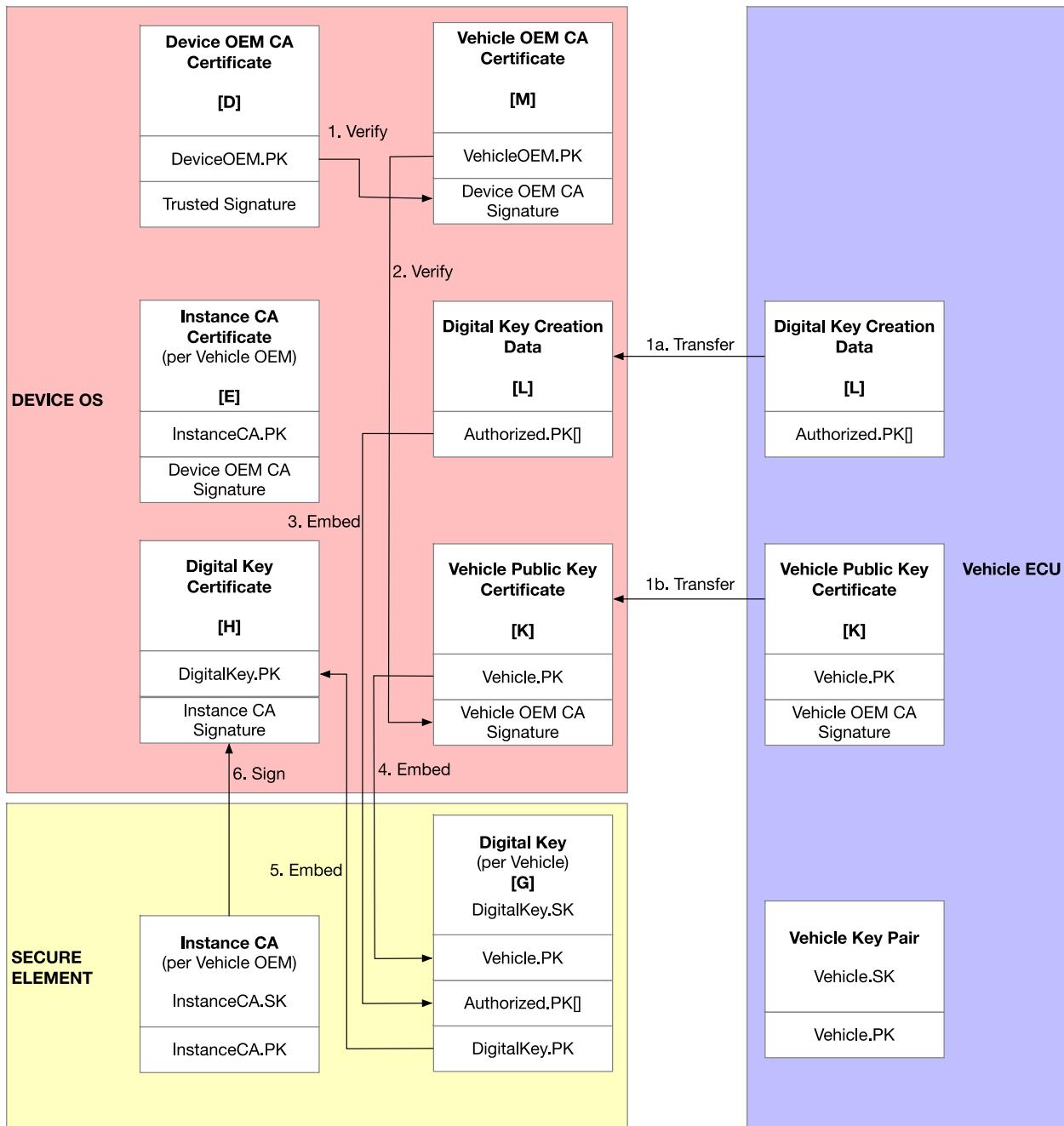
4 *Figure 6-5: Key Creation Data Transfer to Device*



5

1

Figure 6-6: Key Creation Data Transfer to Device



2

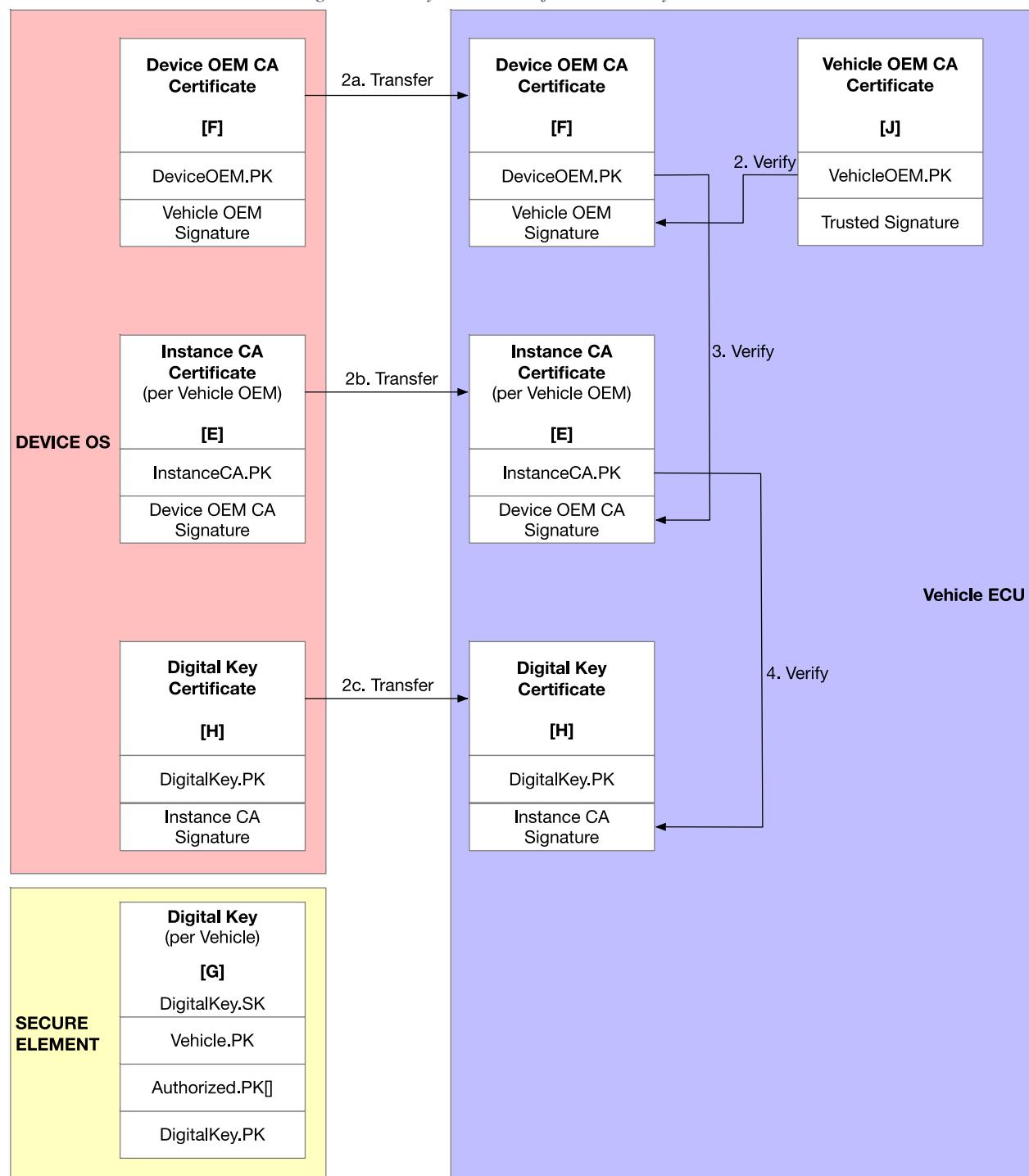
- 3 Certificate details are specified in Section [16](#).
- 4 **6.3.3.4 Steps 5 to 6: OP CONTROL FLOW**
- 5 The OP CONTROL FLOW command indicates that all data has been transmitted without error.
- 6 The device OS then switches off card emulation and creates the Digital Key as described in section [6.3.3.9](#).

1 6.3.3.5 *Step 9 to 12: GETDATA*

- 2 The derived secure channel shall continue to be used in the second session.
- 3 Before sending the GET DATA command, the vehicle shall perform the NFC polling and link
4 setup procedure and reselect the device OS framework AID using SELECT command.
- 5 The vehicle shall poll for the device to respond before considering the key creation as failed. See
6 also section [6.3.3.12](#). For timeouts in case the device is removed from the reader or other
7 connectivity issues occur, see Section [6.3.6](#).
- 8 The vehicle shall send GET DATA command to request the Device OEM CA Certificate signed
9 by the Vehicle OEM [F] (see [Listing 15-13](#)), the Instance CA Certificate signed by the Device
10 OEM CA [E] ([Listing 15-16](#)), and the Digital Key Certificate from the Instance CA [H] (see
11 [Listing 15-5](#)).
- 12 The vehicle shall verify all certificates and the relevant data elements as described in Section
13 [6.3.3.11](#). If all the steps are successful, the vehicle shall store the device public key as the
14 owner's public key.
- 15 The command GET DATA is described in Section [5.1.5](#).

1

Figure 6-7: Key Creation Info Retrieval by Vehicle



- Certificate details are specified in Section [16](#).
- 6.3.3.6 Step 13 to 14: WRITE DATA*
- If all verifications were successful, the vehicle may write back to the device with an opaque attestation (see Table 5-12) using WRITE DATA command to confirm the device PK that has

1 been presented to the vehicle. This attestation shall be sent to the KTS when registering the key.
2 The device shall respond with a WRITE DATA response.
3 If any verification failed, no attestation is written by the vehicle. The vehicle shall send a OP
4 CONTROL FLOW command to the device to abort, with an appropriate error indication as
5 defined in Table 15-27.

6 *6.3.3.7 Step 15 to 18: OP CONTROL FLOW*

7 If vehicle has received all key certificates without error, then it shall send OP CONTROL FLOW
8 ($P1=10_h$, $P2=02_h$) in step 15. Otherwise, the vehicle shall send OP CONTROL FLOW ($P1=12_h$,
9 $P2=\text{reason}$) to abort the owner pairing due to error as indicated by the $P2$ value.
10 If the OP CONTROL FLOW in step 15 is with $P1=10_h$, $P2=02_h$, then the vehicle shall send OP
11 CONTROL FLOW ($P1=11_h$, $P2=11_h$) in step 17 to end the owner pairing Phase 2 flow.

12 *6.3.3.8 Checking the Protocol Version*

13 Device and vehicle may support different versions of SPAKE2+ and Digital Key applet
14 protocols. Device and vehicle shall support older but not deprecated protocol versions.
15 The owner device shall first return all the supported versions of SPAKE2+ and Digital Key
16 applet protocols in the SELECT response. The vehicle shall return the highest matching version
17 of the SPAKE2+ protocol and the agreed version of the Digital Key applet protocol at the
18 beginning of the list, followed by all lower versions supported, ordered from the highest to the
19 lowest. The list of all the supported Digital Key applet protocol versions allows the vehicle and
20 owner device to find a different but compatible version on the friend device for key sharing. An
21 example is given below.

22 When no matching version can be determined, the owner pairing procedure shall be aborted. In
23 this case, the vehicle should indicate this on its UI and shall send an error message to the device
24 using the OP CONTROL FLOW command with “abort” indication.

25 Example of supported Digital Key applet protocol versions:

26 The Digital Key applet protocol versions supported by the vehicle (m supported Digital Key
27 applet protocol versions (ver.high | ver.low)) as contained in SPAKE2+ REQUEST command
28 are received by the device:

- 29 • 0104_h
- 30 • 0103_h
- 31 • 0102_h
- 32 • 0101_h
- 33 • 0100_h

34 The Digital Key applet protocol versions supported by the device (m supported Digital Key
35 applet protocol versions (ver.high | ver.low)) as contained in SELECT RESPONSE are received
36 by the vehicle

- 37 • 0103_h
- 38 • 0102_h
- 39 • 0101_h
- 40 • 0100_h

1 The SELECT RESPONSE APDU would have a TLV with Tag 5Ch: 5C 08 0103010201010100.
2 The SPAKE2+ REQUEST command would have a TLV with Tag 5Ch: 5C 0A
3 01030104010201010100.

4 *6.3.3.9 Creating the Digital Key*

5 The Digital Key framework creates a Digital Key in the SE using the CREATE ENDPOINT
6 command (see Section [15.3.2.4](#)) with the following parameters:

- 7 • **vehicle identifier:** Assigned by the vehicle. It is recommended to apply 2 bytes Vehicle
8 Brand identifier (see [35]) and 6 bytes unique identifier (within Vehicle OEM) to the
9 vehicle identifier. As the vehicle identifier is transmitted in the AUTH0 command, it
10 shall be changed when the owner changes (i.e., when the vehicle goes through an
11 “unpaired” state; see Section [2.7](#) for a description of pairing states) if needed for vehicle
12 privacy.
- 13 • **endpoint identifier:** The common name in the subject identifier in the related certificate.
14 It is used to identify the Digital Key. The endpoint identifier is unique for each Digital
15 Key in an applet instance. The format is defined in [Appendix B.1](#).

16 **Instance CA identifier:** Determines the Instance CA which signs the Digital Key Certificate after key creation. The value shall
17 match exactly the common name in the **subject identifier of the Instance CA** to be associated (see **Note:** applet_version
18 and platform_information are immutable values assigned at the time of the Instance CA
19 certificate is issued. It might not reflect the current applet_version or platform_information (e.g.,
20 in case of applet or Secure Element platform upgrade).

- 21 • Listing 15-16). **Note:** applet_version and platform_information are immutable values
22 assigned at the time of the Instance CA certificate is issued. It might not reflect the
23 current applet_version or platform_information (e.g., in case of applet or Secure Element
24 platform upgrade). The format is described in [Appendix B](#).
- 25 • **Digital Key option group 1 and 2:** Should be set according to the Vehicle OEM policy.
- 26 • **protocol version:** The agreed upon Digital Key applet protocol version for the Digital
27 Key.
- 28 • **vehicle public key:** The public key of the vehicle. For privacy reasons the key may
29 change when the owner changes (i.e., when the vehicle is going through the “unpaired”
30 state; see Section [2.7](#)).
- 31 • **authorized.PK[]:** This array shall contain the Vehicle OEM CA PK. Note that the
32 authorized public key shall not be updated once the Digital Key is created.
- 33 • **confidential mailbox size:** Should be set according to the Digital Key structure
34 configuration and the number of immobilizer tokens to be stored, as defined by the
35 Vehicle OEM. When immobilizer tokens are not needed, the size shall be set to 0. The
36 Digital Key framework shall calculate the size of the confidential mailbox as follows:
37
$$\text{confidential mailbox size} = (\text{SLOT_IDENTIFIERS_OFFSET} - \text{SLOT_IDENTIFIER_BITMAP_OFFSET}) \times 8 \times \text{IMMOBILIZER_TOKEN_LENGTH}$$
- 38 • **private mailbox size:** Should be set according to the Digital Key structure and attestation
39 sizes, as defined by the Vehicle OEM and provided by the vehicle in the WRITE DATA
40 command (see Section [4.3.1](#)).

- **slot identifier:** Provided by the vehicle. It is used for Digital Key identification and anti-replay of the deleted Digital Keys as described in Section [13](#). The vehicle shall provide slot identifier for owner during owner pairing. Depending on vehicle OEM's implementation, the slot identifiers for sharing (for friend) may be retrieved by the owner from vehicle or by the friend online from Vehicle OEM server.

- **counter limit:** Deprecated, shall not be provided.

An owner slot identifier shall in all cases (slot identifier retrieved from vehicle or online) be provided by the vehicle in the endpoint creation data. The value provided here is used for the standard transaction in Phase 3 of owner pairing.

If a Digital Key corresponding to the vehicle identifier already exists in the device, the framework shall terminate the existing Digital Key and send termination attestation along with the Vehicle OEM proprietary data to KTS (if applicable) before creating a new Digital Key.

6.3.3.10 Verifying the Vehicle Public Key Certificate Chain

The device shall parse the certificate chain listed below and validate the critical data elements and the signature before accepting the vehicle public key:

- Vehicle Public Key Certificate format [K]
- Intermediate Certificate (optional)
- Vehicle OEM CA Certificate [J] or Device OEM CA signed Vehicle OEM CA Certificate [M]

All certificates shall be in X.509/ASN.1 format and shall be DER encoded.

The formats of the subject identifier and issuer identifier are defined in [Appendix B.1](#). After successful verification of the vehicle public Key Certificate Chain, the owner pairing Phase 2 is completed.

6.3.3.11 Verifying the Endpoint Creation Attestation Chain

The vehicle shall parse the certificate chain listed below and validate all data elements, following the rules defined in Section [15](#), before accepting the device public key:

- Device OEM CA Certificate (signed by Vehicle OEM) [F]
- Instance CA Certificate (signed by Device OEM) [E]
- Digital Key Certificate [H]

All certificates shall be in X.509/ASN.1 format and shall be DER encoded.

6.3.3.12 Error Handling for the first NFC session

The following errors may appear in the first session of the NFC pairing flow:

- General errors
 - Communication error
 - RF transmission error
 - owner removes device before end of protocol flow, i.e., no deselect command received before link loss
 - commands not received in correct order
 - unknown command received

- 1
 - 2
 - 3 ○ device sends other error status word
 - 4 ○ implementation bug leading to timeout
 - 5 • SELECT command failure
 - 6 ○ owner pairing not activated on device
 - 7 • SPAKE2+ protocol failure
 - 8 ○ wrong password entered on device
 - 9 ○ maximum number of pairing attempts reached (enforced by the vehicle)
 - 10 ○ protocol version mismatch
 - 11 ○ vehicle sends OP CONTROL FLOW command with error indication
 - 12 • Owner key creation on device side fails
 - 13 ○ Vehicle Public Key Certificate [K] verification failed
 - 14 ○ Digital Key Creation Data inconsistency
 - 15 ○ Digital Key configuration not allowed
 - 16 ○ not enough memory in applet to create the Digital Key

17 The device shall enforce that the vehicle sends the commands in the specified order. If this does
18 not occur, the device shall abort the transaction with the error code “Command used out of
19 sequence.”

20 Vehicle and device shall allow up to a maximum of seven retries (using try_cnt, try counter) of
21 the first session. If the maximum number of cumulated failures is reached, vehicle or device shall
22 end the pairing mode and re-provision a new SPAKE2+ verifier.

23 When no matching protocol versions for SPAKE2+ or Digital Key transaction protocol can be
24 found by the vehicle, it shall send an appropriate OP CONTROL FLOW command (namely,
25 “abort”) and shall not send the SPAKE2+ REQUEST command. In this case the device does not
26 expect additional commands from the vehicle.

27 After the OP CONTROL FLOW command, the vehicle shall perform the NFC link teardown
28 procedure followed by the NFC reset procedure, before a new or updated device is ready to
29 begin the first step in owner pairing (Phase 2) again.

30 In the case of a failed transaction due to tearing that cannot be compensated by the contactless
31 protocol, the vehicle shall perform the NFC reset procedure (as defined in Section [3.2.4](#)) and
32 start the NFC polling and link setup procedure.

33 In case of an application timeout (not on contactless protocol level), the vehicle shall send an OP
34 CONTROL FLOW command indicating a timeout error.

35 If any error occurs during the first NFC session before key creation, no Digital Key shall be
36 created in the Digital Key applet.

37 When a communication error occurs, the Digital Key framework shall not delete the Digital Key
38 and shall wait for the vehicle to retry the flow.

39 If a device sends an error SW (i.e., SW not equal 9000_h or 61XX_h) which is listed in the table
40 below, the listed behavior shall occur.

1

Table 6-1: Error SW Condition List 1.

Command	Error SW	Device Action	Vehicle Action
SPAKE2+ VERIFY	6A88 _h	Stop HCE UI: "Wrong Password"	Reset

- 2 When a unlisted error SW is sent, the error counter shall be increased, and the flow shall be
3 retried if the counter has not yet reached the maximum.
4 When an error is indicated in an OP CONTROL FLOW command after the Digital Key is
5 created and the vehicle rejects the pairing due to failing verification of the Digital Key
6 Certificate data, the Digital Key framework shall delete the Digital Key that has been created.
7 On receipt of an OP CONTROL FLOW command indicating a timeout error, if the maximum
8 number of cumulated failures is reached, vehicle or device shall end the pairing mode. The user
9 shall request a restart of the owner pairing procedure for which the Vehicle OEM Server has to
10 re-provision a new SPAKE2+ verifier and password.
11 When the device detects a link loss or receives an unexpected command without receiving an OP
12 CONTROL FLOW command immediately beforehand, the transaction shall be considered as
13 failed and should be allowed to be repeated a limited number of times (e.g., 5).
14 When the Digital Key has been created but the Digital Key Certificate [H] has not been
15 requested by the vehicle, the Digital Key shall not be usable.
16 A policy on the vehicle side shall limit the number of unsuccessful owner pairing attempts per
17 set of pairing password/vehicle verifier to seven as outlined in Section 5.1.2. When a successful
18 pairing is completed or when the maximum number of failed pairing attempts is reached, the
19 vehicle shall delete the verifier immediately. In case of a successful owner pairing, the verifier
20 shall not be updated before the vehicle has verified the key tracking signature unless it was
21 invalidated for a different reason.
22 When the owner pairing flow is permanently aborted (e.g., via timeout on device, maximum
23 number of retries reached, etc.), the created Digital Key shall be deleted by the device.
24 During owner pairing Phase 2, the device shall prevent selection of the Digital Key applet by the
25 vehicle and shall respond to SELECT Digital Key applet AID with Status Word (SW) = 6A82.
26

27 6.3.4 Phase 3: Second Session with NFC Reader

28 The second NFC session is executed between vehicle and Digital Key applet as described in
29 Figure 6-8.

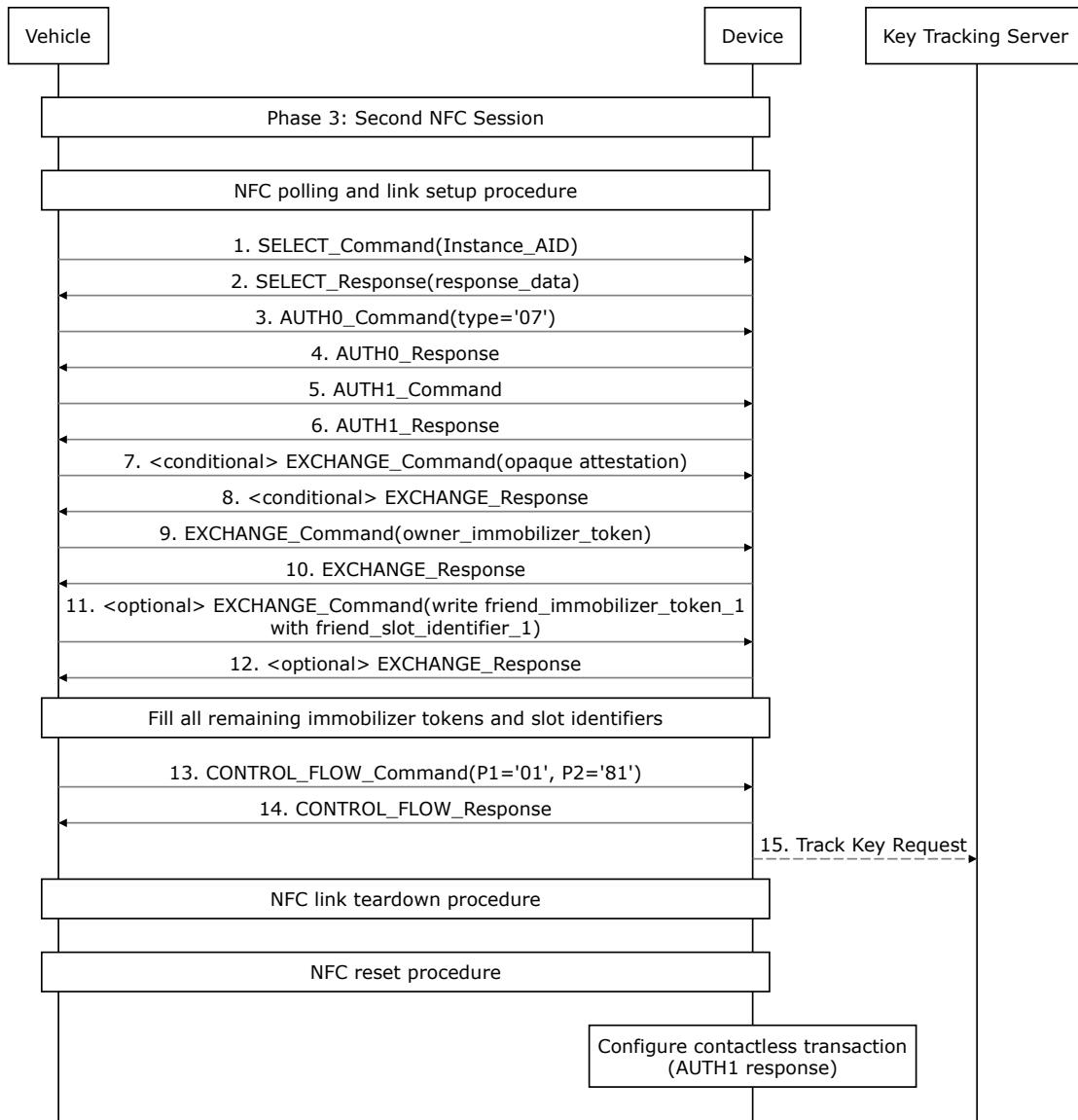
30 6.3.4.1 Steps 3 to 12: Standard Transaction

31 The vehicle shall select the Digital Key applet and execute a standard transaction (step 3–6; see
32 Section 7) using transaction type 07_h (see Table 9-1).
33 If the vehicle did not write the opaque attestation as shown in step 13 in Figure 6-3 (i.e., no
34 WRITE DATA command is sent by vehicle), then the vehicle shall first write the opaque
35 attestation into the private mailbox. If the immobilizer token is required and retrieved from
36 vehicle, it shall be written into the confidential mailbox of the owner Digital Key in the Digital
37 Key applet.

- 1 In order to write the appropriate data structures into the confidential and private mailboxes, the
2 vehicle shall use the EXCHANGE command (see Section [15.3.2.15](#)) with the appropriate
3 parameters to:
- 4 • Write the opaque attestation with the tag 5F5A_h tag and length into the KeyAtt field in
5 the private mailbox, if the opaque attestation has yet to be written
6 • Write the owner immobilizer token into the confidential mailbox (if immobilizer tokens
7 are retrieved from the vehicle)
8 • Write the slot identifier bitmap into the private mailbox (conditional)
9 • Write the owner slot identifier into the slot identifier list in the private mailbox
10 (conditional)
11 • Write the Vehicle OEM proprietary data structure into the private mailbox
12 • Write the signaling bitmap to indicate the update of the above data structures into the
13 private mailbox
14 • Write immobilizer tokens for sharing into the confidential mailbox (if immobilizer tokens
15 are retrieved from the vehicle and if KTS is not implemented by the Vehicle OEM)
16 (optional)
- 17 The owner slot identifier is not required as it is included in the endpoint certificate. The vehicle
18 OEM decides whether it needs to be written or not.
- 19 It is assumed that no owner access rights need to be provisioned into the Digital Key applet. The
20 vehicle shall grant full access to the owner.
- 21 The Digital Key applet shall notify the framework at the end of the transaction, which then shall
22 configure the contactless transaction as described in Section [6.3.4.4](#).

1

Figure 6-8: Owner Pairing Flow – Phase 3: Second NFC Session



- 2
- 3 6.3.4.2 *Step 11 and 12: <optional> Provision Disabled Friend Immobilizer Tokens*
4 If the vehicle can provide appropriate security measures and if immobilizer tokens/slot identifiers
5 are provided by the vehicle, the vehicle may write the friend immobilizer tokens and slot
6 identifiers as part of phase 3 (see Figure 6-8). If key tracking is required, the vehicle shall keep
7 them disabled until the key tracking signature has been successfully verified.
- 8 6.3.4.3 *Steps 13 and 14: CONTROL FLOW*
9 The vehicle shall send the CONTROL FLOW command (see Section [15.3.2.16](#)) to indicate that
10 all data has been written successfully into the mailboxes. The P2 parameters shall be set as
11 follows:

P2=81_h to indicate to the Digital Key applet that vehicle has completed owner pairing phase 3.

At the end of Phase 3, the owner immobilizer token shall be provisioned in the owner device if immobilizer token is retrieved from the vehicle, and the key may or may not yet be tracked by the vehicle.

6.3.4.4 Step 15: Key Tracking Request

If the standard transaction was successful, the owner device shall send an asynchronous message to register the owner key to the KTS. Key registration is optional for the vehicle. If the vehicle attempts to register the key as well, a race condition may occur. Thus, key registration by vehicle is not recommended.

Note: When the standard transaction fails, the device shall not send a key tracking request to the server. The data elements listed in Table 6-2 shall be sent to the KTS by the device.

Table 6-2: Owner Key Tracking Request

Tag	Length (bytes)	Description	Element is
7F3E _h	variable	Owner key tracking request	
		[E] Instance CA Certificate per Vehicle OEM as per Figure 6-7 and Table 11-13 .	mandatory
		[H] Digital Key Certificate as per Figure 6-7 and Table 11-14 .	mandatory
		Endpoint encryption key attestation as per Table 15-47 signed by endpoint private key (present only if immobilizer token is required)	conditional
D5 _h	variable	Attestation of the device PK by the vehicle (opaque)	mandatory
D3 _h	variable	Friendly name of the owner key	mandatory
5F49 _h	65	Device privacy encryption key (Device.Enc.PK)	mandatory
DA _h	variable	Device privacy encryption version Default: “ECIES_v1” (ASCII)	mandatory

Table 6-3: Owner Key Tracking Response Parameters

Parameter	Length	Description
kts-Signature	variable	See Section 17.8.5 unencrypted uiBundle
slotIdentifier	1-8	See Section 17.8.15 vehicleMobilizationDataFields
confidentialMailboxData	variable	See Section 17.8.15 vehicleMobilizationDataFields

All parameters of the Key Tracking Response from the KTS shall not be provided in TLV format. The server API parameters shall be provided as a hex string within the JSON structure defined in the relevant sections referenced in Table 6-3.

- 1 If the slot_identifier is provided, it shall be cryptographically bound to the kts-Signature (by
2 vehicle OEM proprietary methods not defined in this specification) so that the vehicle can verify
3 the authenticity of the slot_identifier.
- 4 The content of the kts-Signature is defined by the Vehicle OEM; it is opaque to the device OEM.
5 The signature shall contain a cryptographic confirmation that the owner key has been tracked by
6 the KTS and shall link the signature to the device public key.
- 7 The device shall add the tag 45_h and length fields to the kts-Signature (see Section 17.8.5) and
8 stores the TLV structure in the KeyAtt field of the private mailbox (see Table 4-2).
- 9 Endpoint encryption key attestation (Tag 7F26_h) shall be included in Table 6-2 when Tag DA_h in
10 Tag 7F60_h in Table 5-14 is set to 01_h.
- 11 Owner slot identifier shall be included in the Key Tracking Response as defined in Table 6-3
12 when Tag DA_h in Tag 7F60_h in Table 5-14 is set to 01_h or 03_h.
- 13 Owner immobilizer elements (confidentialMailboxData and ephemeralPublicKey parameters)
14 shall be included in the Key Tracking Response as defined in Table 6-3 when tag DA_h in tag
15 7F60_h in Table 5-14 is set to 01_h.
- 16 If the owner Key Tracking Response contains a slot identifier, then this slot identifier shall be
17 written to the owner endpoint using the SETUP ENDPOINT command. If no slot identifier is
18 provided in the owner key tracking response, then the slot identifier provided by the vehicle in
19 the endpoint creation data is the one to be used.
- 20 All certificates in Table 6-2 shall be in X.509/ASN.1 format and shall be DER encoded.
- 21 The opaque attestation is obtained from the KeyAtt field of the private mailbox. Note that
22 sending the attestation as part of the standard transaction is recommended. Only value field of
23 Tag 5F5A_h shall be included in the value field of Tag D5_h.
- 24 See Section [17](#) for details on the format of the key tracking request.
- 25 If immobilizer tokens and/or slot identifiers are retrieved online, the owner device shall send Key
26 Tracking Request to the server (see Section [17.7.1.2](#)) to request owner immobilizer token and/or
27 slot identifier.

28 *6.3.4.5 Configuration of the Contactless Transaction*

29 There are two options to access mailbox data within the standard transaction:

- 30 • Transmit the required data as part of the AUTH1 response
- 31 • Use the EXCHANGE command (mandatory for the fast transaction)
- 32 If the WRITE DATA command received from the vehicle contains tag 4A_h and/or tag 4B_h in the
33 Device Configuration Data field (see step 3 in [Figure 6-3](#) and Section [5.1.4](#)), the device shall
34 transmit the signaling bitmap and/or owner immobilizer token in the AUTH1 response as
35 described in Table 15-33 and the Digital Key framework shall set the endpoint using the SETUP
36 ENDPOINT command (see Section [15.3.2.21](#)) with the following parameters:
- 37 • **confidential offset:** Shall be set to 0 as defined in Table 4-3.
- 38 • **confidential length:** Shall be set to IMMOBILIZER_TOKEN_LENGTH as defined in
39 Table 4-3 if immobilizer tokens are used.
- 40 • **private offset:** Shall be set to SIGNALING_BITMAP_OFFSET as defined in Table 4-2.

- 1 • **private length:** Shall be set to (SLOT_IDENTIFIER_BITMAP_OFFSET –
2 SIGNALING_BITMAP_OFFSET) as defined in Table 4-2.

3 The other arguments in this method shall not be used.

4 *6.3.4.6 Error Handling for the second NFC session*

5 The following errors that appear in the second session of the NFC pairing flow include:

- 6 • General errors that may include
7 ○ Communication error
8 ■ RF transmission error
9 ■ owner removes device before end of protocol flow, i.e., no deselect
10 command received before link loss
11 ○ commands not received in correct order
12 ○ device sends other error status word
13 ○ implementation bug leading to timeout
14 • SELECT command failure
15 ○ applet not installed
16 ○ device configured for applet when HCE selected over NFC

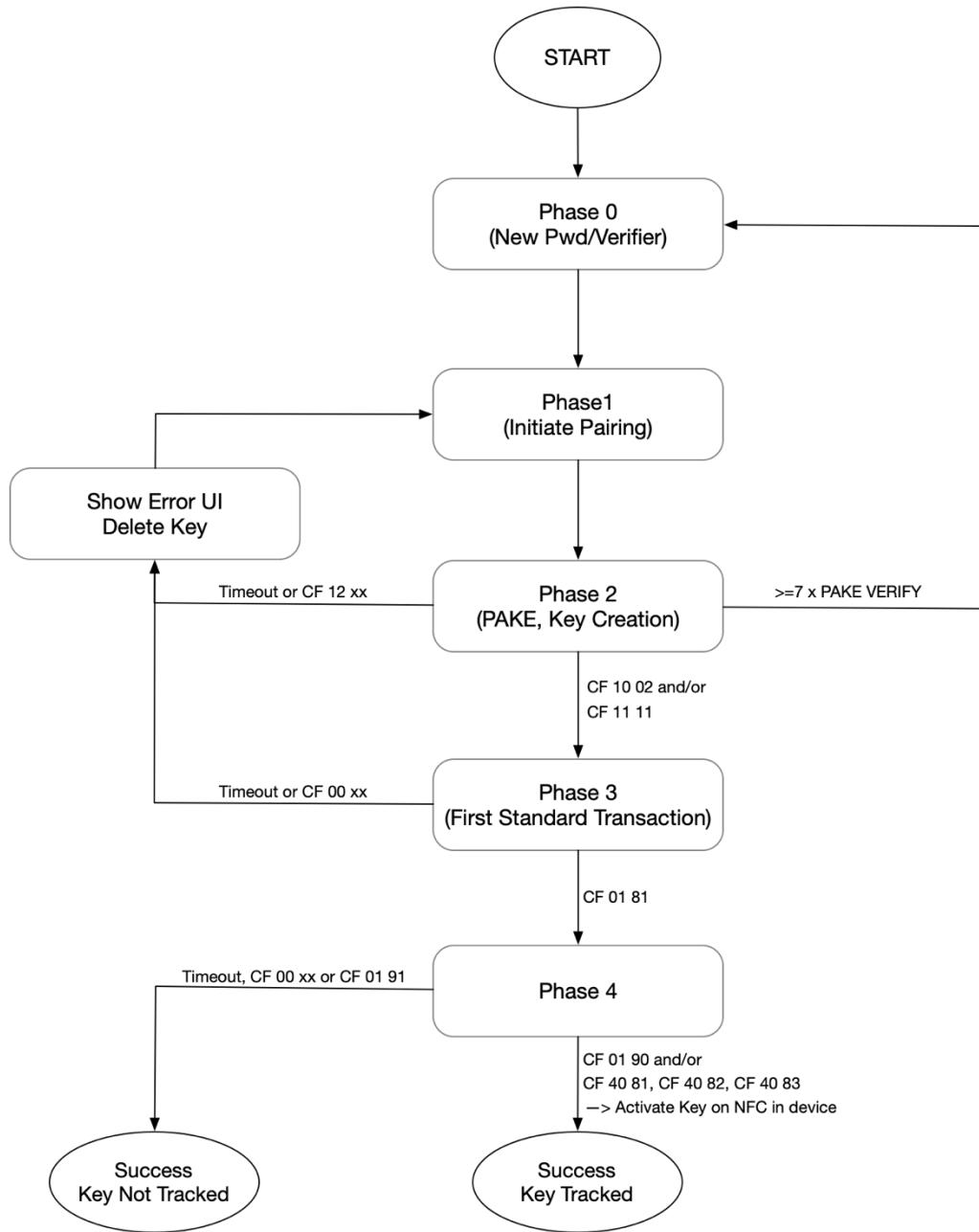
17 Figure 6-9 shows the error management in owner pairing and the relationship between the
18 different phases. No retry shall be allowed in a specific phase and any error which is not fixed on
19 NFC protocol level shall lead to complete restart.

20 If the standard transaction (in step 3–6) failed, the endpoint may be deleted.

21 As soon as the Digital Key is successfully created, it might be usable on the, even if the owner
22 immobilizer token is not yet provisioned. For a consistent user experience, it is recommended to
23 enable the created owner key only when the device has confirmed that the owner immobilizer
24 token and the opaque attestation has been correctly written.

1

Figure 6-9: Error Management of Different Phases in Owner Pairing



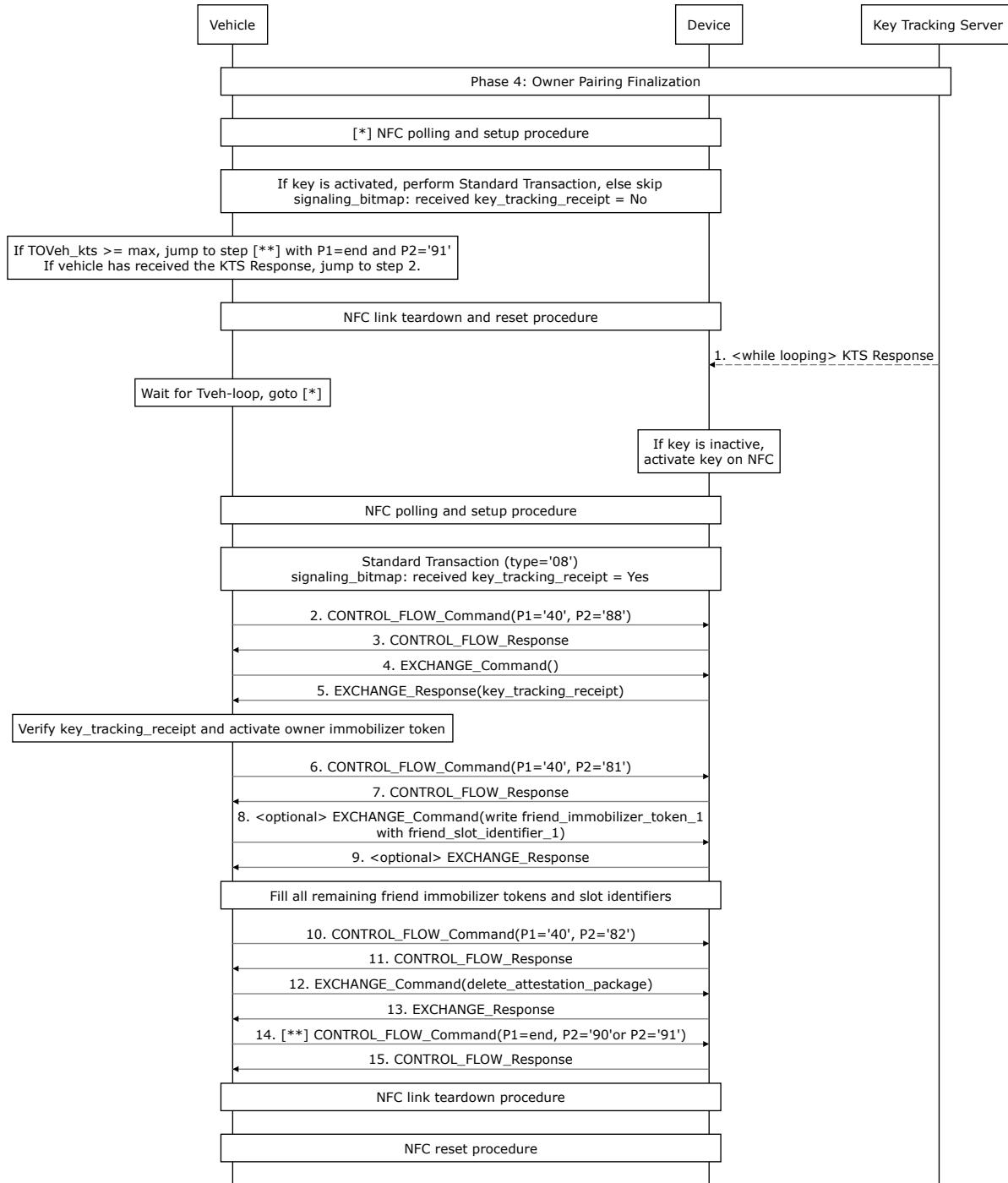
2

6.3.5 Phase 4: Finalization of Pairing Procedure (optional)

- 3 The finalization step executes a standard transaction using transaction type 08_h (see Table 9-1) in regular intervals to poll the signaling bitmap indicator for an attestation package in the device.
- 4 Figure 6-10 shows the final phase of the owner pairing flow.

1

Figure 6-10: Owner Pairing Flow – Phase 4: Finalization



2

3 6.3.5.1 Before Step 2: Reader Polls for KTS Response

4 The owner device and (optionally) the vehicle have reached out to the KTS by sending a key
5 tracking request, as defined in Table 6-1 in Phase 2.

- 1 If the device receives a Key Tracking Response from the KTS (see Table 6-3), the device shall store the kts-Signature which is stored in the KeyAtt field in the private mailbox (see Figure 4-4), the corresponding bit in the signaling bitmap (SigBmp) shall be set by the device to indicate to the vehicle the presence of the kts-Signature.
- 5 The reader executes standard transactions to check the signaling bitmap in the device. If the kts-Signature is not present or the device does not respond⁴, the reader shall try again after a time (Tveh-loop) defined in Section [6.3.6](#)). The reader shall perform the NFC Reset procedure after each standard transaction.
- 9 If the reader experiences a loss of connection, it shall perform the NFC Reset procedure and shall restart the NFC Polling and Link Setup procedures.
- 11 If the vehicle receives the Key Tracking Response, the vehicle shall first skip the verification of the kts-Signature in the device and then continue by writing the friend immobilizer tokens into the mailbox if immobilizer tokens are retrieved from the vehicle.
- 14 If neither vehicle nor device receives a response before TOveh_kts has expired, the vehicle shall not provision friend immobilizer tokens and shall signal the failure to get a KTS signature through a CONTROL FLOW command (see Section [6.3.5.5](#)).
- 17 *Note:* The device may not respond to NFC polling until it has received the key tracking response.
18 The vehicle shall time out (by TOveh_kts) even if the device never responds.

19 *6.3.5.2 Steps 2 to 5: Verification of the Key Tracking Receipt in Device*

20 When the device obtains a KTS signature, the vehicle shall execute a standard transaction and shall send a CONTROL FLOW command (see Section [15.3.2.16](#)) to indicate the status:

- 22 1. P1=40h and P2=88h: continue, key tracking response received in device; next step is to read the receipt from the mailbox
- 24 2. P1=40h and P2=89h: continue, key tracking response received in vehicle, go directly to verification of key tracking receipt

26 Steps 2, 3, 4, 5, 10, 11, 12 and 13 shall be executed only when the KTS signature is stored in the device. If a key tracking response is received by the vehicle, only steps 2 and 3 are executed as defined in 2, above.

29 The vehicle shall then verify the KTS signature.

30 If the verification of the KTS signatures was successful, the vehicle shall continue with the next step. Otherwise, the vehicle shall abort and execute steps 14 and 15 with an appropriate error indication (see Section 6.3.5.5). If immobilizer tokens are not required, the vehicle can proceed directly to step 10 (see Section [6.3.5.4](#)). Otherwise, the vehicle shall abort and execute steps 14 and 15 with an appropriate error indication.

35 *6.3.5.3 Steps 6 to 9: Provisioning Immobilizer Tokens for Sharing*

36 If offline provisioning of immobilizer token is not required (indicated by the SHARING_CONFIGURATION field in Table 5-14: Tag 7F60_h and Tag DA_h), the vehicle can proceed directly to step 10 (see Section 6.3.5.4).

⁴ A device may choose not to enable the key on the NFC interface or BLE/UWB interface until the kts-Signature is present in the private mailbox.

1 If the vehicle requires offline provisioning of immobilizer token, the vehicle shall indicate the
2 start of the procedure by sending a CONTROL FLOW command (step 6):

3 P1=40_h and P2=81_h: continue, optional friend immobilizer token refill.

4

5 The immobilizer tokens required for key sharing shall be provisioned into the confidential
6 mailbox. Signaling bits and slot identifiers shall be written accordingly using the EXCHANGE
7 command.

8 *6.3.5.4 Steps 10 to 13: Deleting the attestation package*

9 If there is a key tracking receipt present in the private mailbox, which was already verified by the
10 vehicle, the vehicle shall delete the key tracking receipt as follows:

11 The vehicle shall indicate the deletion of the key tracking receipt by sending a CONTROL
12 FLOW with:

13 P1=40_h and P2=82_h: continue, attestation package delete start.

14 The vehicle shall clear the signaling bitmap and delete the attestation package from the private
15 mailbox using an EXCHANGE command, so that the vehicle does not detect the same attestation
16 package a second time, on the next transaction.

17 *6.3.5.5 Steps 14 to 15: CONTROL FLOW*

18 The CONTROL FLOW command indicates the end of the owner pairing flow with the following
19 options:

20 1. P1=01_h and P2=90_h: end, owner key is tracked, and all data has been written successfully
21 into the mailboxes.

22 2. P1=01_h and P2=91_h: end, owner key is not tracked, key sharing is not possible, and the
23 owner needs to go online to track the key before using it.

24 **6.3.6 Timers**

25 The owner pairing flow is limited by retry counters and by timers on vehicle and device sides.

26 The vehicle limits the overall time of a pairing attempt, from the first detection of the device
27 responding successfully to the selection of the framework AID until the successful provisioning
28 of the immobilizer token in the Digital Key.

29 Phases and sessions are limited by shorter timer values, to be able to detect failures more
30 quickly.

31 Vehicle overall timer TOveh-1 is used to control the owner pairing procedure. This timer is not
32 linked to the device timer TODEV-1.

33 The device should at least have a timer (TODEV-1) to limit the overall time of the owner pairing
34 procedure until the key and immobilizer token are provisioned. On timeout, the previously
35 created key shall be deleted, and a new entry of the pairing password shall be required.

36 A timer should not be restarted if it is already running, and the flow passes again through the
37 “Start Timer” state (which is unlikely).

38 The following scheme provides more details about the timer start and end conditions. It shows
39 the specific timeouts from vehicle and device perspective.

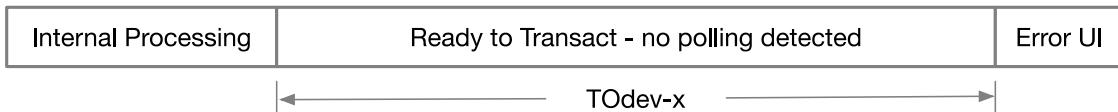
1

Figure 6-11: Vehicle and Device Transaction Timeout

Vehicle Transaction Timeout



2



3 Table 6-4 defines the timer values. The timer numbers refer to Figure 6-11 with a split into
4 vehicle and device side.

5

Table 6-4: Recommended Minimum Vehicle and Device Timeout Values

Timer	Description	Value
TOveh-1	Overall OP transaction timer (detection of device until final UI)	30 sec
TOdev-1	Overall OP transaction timer (first response to polling until KTS signature not received)	30 sec
TOveh-2	not required	
TOdev-2	Start of pairing mode until detection of polling	Device OEM specific
TOveh-3	Last WRITE DATA command until first response to polling for GET DATA	10 sec
TOdev-3	Endpoint created until detection of polling	5 sec
TOveh-4	Endpoint data processed until first response to polling for first standard transaction	5 sec
TOdev-4	Endpoint data sent to vehicle until detection of polling	10 sec
Tveh-loop	Time of repeated standard transaction to retrieve KTS receipt (Note 1) - this is not a timeout, but a loop timer	1 sec
TOveh-kts	Time until KTS receipt should be received	20 sec

6 Note 1: The device might not respond to polling until the KTS receipt is received.

7

6.3.7 Pairing Password URL

8 To start owner pairing mode, the device may receive the pairing password through a URL. The
9 Vehicle OEM may provide the owner with the pairing password as fragment of a URL (e.g. via
10 an email). When the owner consumes this URL on the device, the device will redirect this link to
11 the Framework, which extracts the pairing password from the URL and uses it to start the owner
12 pairing. The URL is not intended to be reached out to. It is used to transport the parameters to be
13 interpreted by the framework.

- 1 The URL formatting rules are specified in RFC 3986 [27]. The URL syntax shall take the
2 following form:
3 scheme://authority/version/redirect?query#fragment
4 For example:
5 https://digitalkeypairing.org/v1/0001?technology=NFC,BLE&graphics=0001A0A1A2A3
6 #pwd=12345678

7 *Table 6-5: Pairing Password URL Syntax*

Component	Length (character)	Description	Field is
scheme	5	“https”: HTTP protocol operating over TLS	mandatory
authority	21	Domain name defined and registered by CCC Domain: digitalkeypairing.org	mandatory
version	2	“v1”: Version number to identify the fields in the URL	mandatory
redirect	4	Vehicle Brand Identifier [35]. This is used for server redirection in owner pairing.	mandatory
query		Non-sensitive URL parameters	mandatory
technology	variable	Communication interfaces supported for pairing by the vehicle. This parameter may be “NFC” or “NFC,BLE” If this query is not present, the communication interface is expected to be “NFC”	optional
graphics	12	The NFC reader position graphics identifier of the vehicle which is provided by vehicle OEM. This identifier is given as [Vehicle Brand Identifier 4-byte vehicle defined graphics identifier], represented as string. The Vehicle Brand Identifier is defined in [35].	optional
fragment		Sensitive URL parameter Device strips the fragment from the URL when using the URL to contact the authority to preserve the confidentiality.	mandatory
pwd	8	pairing password	mandatory

8
9

1 **7 STANDARD TRANSACTION**

2 The standard transaction protocol is intended to provide the following properties:

- 3 • Mutual authentication
- 4 • Forward secrecy
- 5 • Tracking resilience
- 6 • Integrity and confidentiality

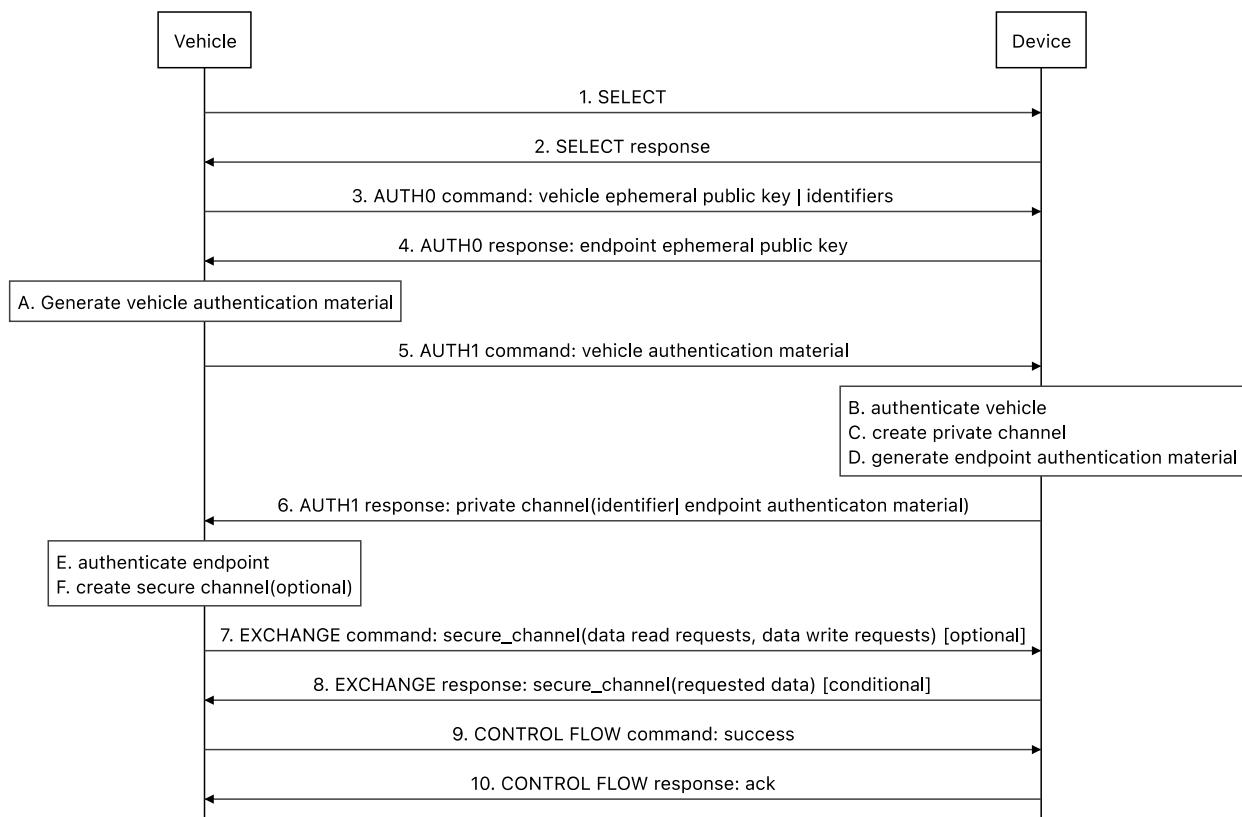
7 A secure channel between vehicle and device is initiated by generating ephemeral key pairs on
8 the vehicle and device sides. Using a key agreement method, a shared secret can be derived on
9 both sides and used for generation of a shared symmetric key, using Diffie-Hellman and a key
10 derivation function.

11 The ephemeral public key generated on the vehicle side is signed with the vehicle's private key,
12 vehicle_SK. This results in an authentication of the vehicle by the device. From the device's
13 perspective, this guarantees that no privacy-sensitive data can be leaked by a MITM attack. This
14 principle also allows the device to transmit data to the vehicle without any possibility of leakage
15 by a passive or active eavesdropper.

16 Finally, the device uses the established secure channel to encrypt its public key identifier along
17 with the signature computed on a vehicle's data-derived challenge and some additional
18 application-specific data. This verification of the device's signature by the vehicle allows the
19 vehicle to authenticate the device.

1

Figure 7-1: Standard Transaction Flow



2

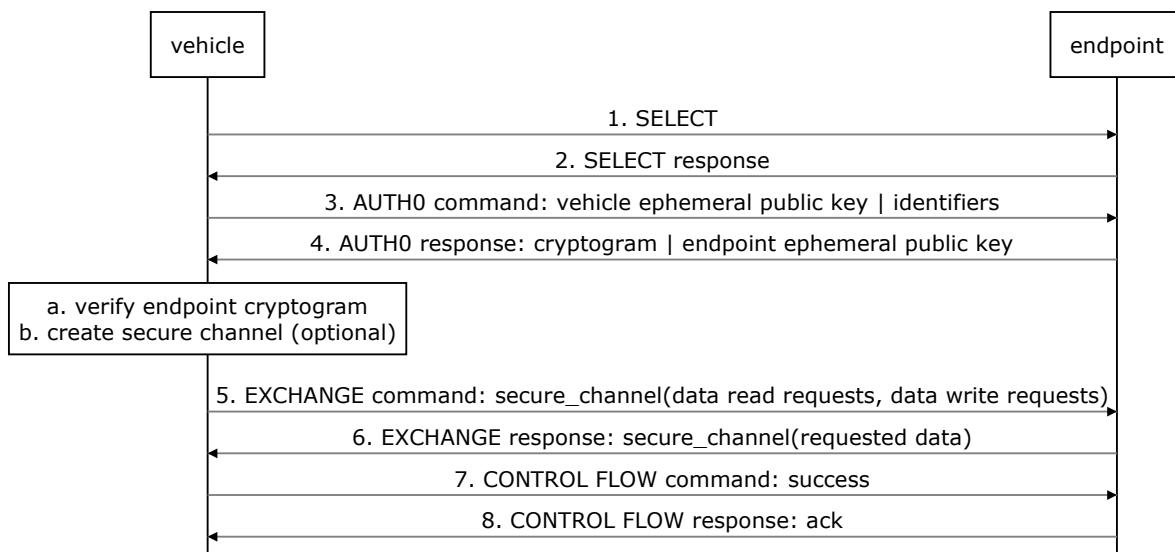
1 8 FAST TRANSACTION

2 The fast transaction protocol is intended to provide the following properties:

- Device authentication or Mutual authentication
 - Integrity and confidentiality
 - Tracking resilience

6 The device generates a cryptogram based on a secret previously shared during a standard
7 transaction, and this allows the vehicle to authenticate the device. Optionally, a secure channel
8 between vehicle and device is established by deriving session keys from a secret previously
9 shared during a standard transaction and from the ephemeral keys. The ability of the vehicle to
10 establish the secure channel authenticates the vehicle to the device. The shared secret shall
11 always be regenerated in the standard transaction when fast transaction is allowed by the
12 endpoint configuration defined during endpoint creation. Refer to Section 15.3.2.4 (see Table
13 15-13)

Figure 8-1: Fast Transaction Flow



1 9 USER AUTHENTICATION

2 Devices shall provide the functionality to enforce user authentication according to their user
3 authentication (UA) policies. The device shall provide at minimum one of the UA policies ([A],
4 [AL], [E], [T]) (see [Table 9-2](#)). The user authentication policy is set by the user on the device.
5 UA is enforced by the device only. The conditions for enforcing UA are determined by the
6 selected UA policy, based on the transaction type (see [Table 9-2](#)).

7 The transaction type is set as a domain specific transaction_code value in P2 parameter of the
8 AUTH0 command. Transaction type assignments are specified in [Table 9-1](#).

9 User authentication may be enabled or disabled individually for each Digital Key. The UA
10 policy is stored as a list of transaction types that require user authentication. For every new
11 Digital Key, the user authentication should be set to Off [O] by default (see [Table 9-2](#)).

12 *Table 9-1: AUTH0 Command Transaction Type Coding*

P2 (transaction_code)	Transaction Type
00 _h	RFU
01 _h	Door unlock
02 _h	Door lock
03 _h	First engine start authentication, first contact of a device with engine start rights in a vehicle session (e.g. door is in unlock state before the vehicle session starts and then first engine start in a new vehicle session)
04 _h	First engine start authentication, subsequent contact of a device with engine start rights in a vehicle session (e.g. door is in lock state before the vehicle session starts, and door is unlocked before first engine start)
05 _h	Other authentication request by vehicle
06 _h	User authentication request by vehicle
07 _h	First standard transaction at owner pairing (owner immobilizer token provisioning)
08 _h	Second standard transaction at owner pairing (read KTS receipt, provision friend immobilizer tokens)
09 _h - 0F _h	RFU
10 _h	Derive ranging key (standard transaction only)
11 _h	First approach (standard transaction only), i.e. first standard transaction after Bluetooth LE pairing between vehicle and device outside of owner pairing
12 _h - FF _h	RFU

13 A vehicle session starts with the first usage of the Digital Key after the end of a preceding
14 vehicle session and ends when the vehicle is locked or after an appropriate timeout of non-usage
15 of the vehicle.

16 [Table 9-2](#) shows the assignment of policies to the transaction types and the associated resulting
17 user experience.

1
2

Table 9-2: User Authentication Policies

		Device UA Policy												
Notation	Resulting User Experience	UA based on Transaction Code												
		00 _h	01 _h	02 _h	03 _h	04 _h	05 _h	06 _h	07 _h	08 _h	09 _h –0F _h	10 _h	11 _h	12 _h –FF _h
Off [O]	UA never requested	-	-	-	-	-	-	X	-	-	-	-	-	-
Access [A]	UA requested on first access or first engine start	X	X	-	X	-	-	X	-	-	X	-	-	X
Access [AL]	UA requested on first access and lock or first engine start and lock	X	X	X	X	-	-	X	-	-	X	-	-	X
Engine [E]	UA on every engine start	X	-	-	X	X	-	X	-	-	X	-	-	X
Always [T]	UA on every usage	X	X	X	X	X	-	X	-	-	X	-	-	X

- 3
4 • X: UA to be performed on the device
5 • -: UA not required for the transaction to be successful
6 The owner should not be able to restrict the UA setting options of a shared key.
7 Note: When user authentication is set for a Digital Key, the key may not be usable in Battery
8 Low Mode.

9.1 Explicit User Authentication Policy

To improve security, it may be beneficial to require explicit UA before an action can be triggered. Explicit UA shall require an intent for RKE actions to be registered before explicit UA is enforced. The associated RKE action shall only be performed if the explicit UA has been performed successfully. The time between explicit UA and the action shall not exceed 1 minute except if the device is set to a state where only Digital Key functions can be triggered. Upon leaving this state where only Digital Key functions can be triggered, an explicit UA shall be performed before another action can be performed.

9.2 Implicit User Authentication Policy

To improve the user experience, it may be beneficial for the device to allow a single successful UA event to authorize several Digital Key actions within a short period of time (grace period). This may be coupled to a UA used for device unlock. If the user device implements such a grace period, it should not exceed 5 minutes. If supported by the device, additional successful UA events should cause this grace period timer to start/restart. The grace period shall expire immediately when the UA becomes invalid, such as when the device is locked.

This implicit UA shall not be used for critical actions, such as endpoint authorization, key sharing, or when a device's UA policy explicitly forbids implicit UA to be used for a transaction type or remote feature execution.

- 1 The user should be able to enable or disable usage of this implicit UA. The user may also be able to configure the length of the grace period between zero minutes (disabled) and the maximum time defined above.
- 2
- 3
- 4

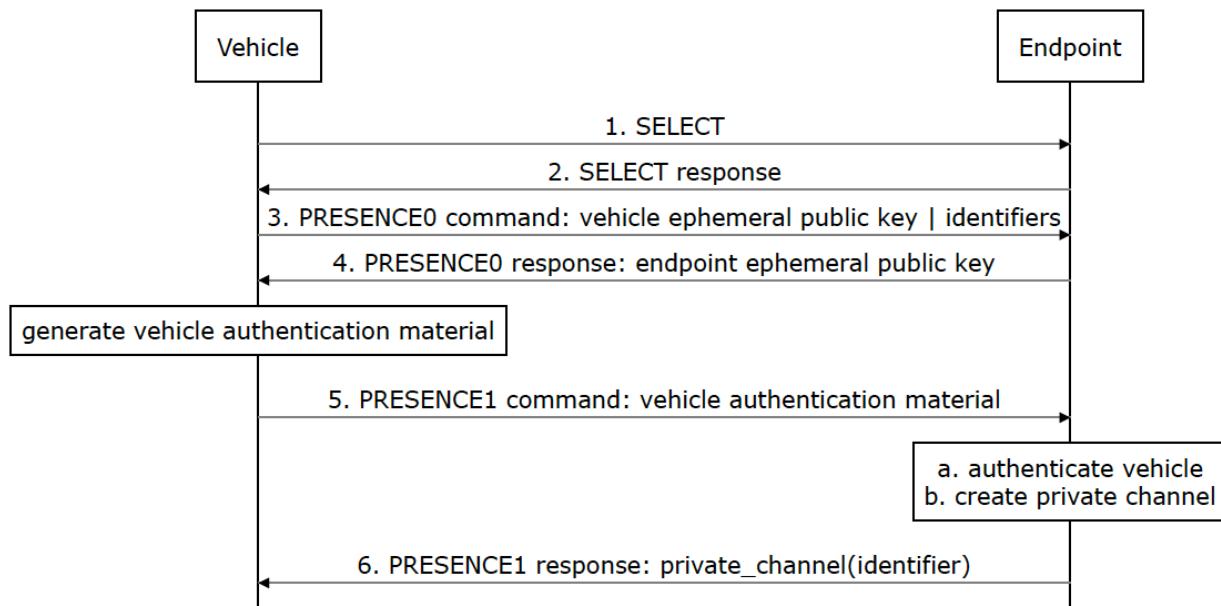
10 CHECK PRESENCE TRANSACTION

2 The check presence transaction protocol is intended to provide the following properties:

- 3 • Vehicle authentication
- 4 • Device identification
- 5 • Integrity and confidentiality
- 6 • Tracking resilience

7 The mechanism is similar to the standard transaction mechanism described in Section 7, except
8 that the device signature is not sent to the vehicle, and user authentication is disabled. The goal is
9 to allow verification of device presence near the vehicle without requiring user authentication,
10 while preventing tracking.

11 *Figure 10-1: Check Presence Transaction*



12

11 DIGITAL KEY SHARING [WCC1/WCC2]

11.1 Encoding

All messages exchanged over the communication channels described in Section 11.2 are compliant with Basic Encoding Rules-TLV(BER-TLV) as outlined in [40]. Certificates are compliant with X.509 format, as per document [3].

The TLV fields shall be ordered as described in this specification. A different field order is considered invalid unless specified otherwise. The nesting level is represented by indentation of tag values in the tag column.

11.2 Sharing Principles

11.2.1 Definitions

11.2.1.1 Sharing URL

To initiate the key sharing process, the Owner device requests the Relay Server to create a Mailbox on the relay server. The sharing URL is generated by the relay server and provided to the owner device. The owner device provides the sharing URL to the friend device to initiate the transfer of the DK. The sharing URL shall accept additional parameters. The version is determined by the relay server when generating the URL. The version is determined for use by the relay server and shall not have any impact on the owner or friend device.

18

11.2.1.2 Sharing Password

If supported by the Vehicle OEM, it is provided to the owner, which provides it to the friend via a dedicated channel. The friend then types this into the vehicle UI for key activation. Vehicle OEM verifies that the correct sharing password was entered. This specification includes sharing password as an *activation option* defined in Section 11.2.1.4 and Section 17.8.19. Hence the sharing password as defined in previous versions of this specification [32] is considered obsolete. Vehicle OEM decides whether to implement this option or not.

11.2.1.3 Device PIN

Can be set by the owner device during the sharing workflow if no sharing password is required by the vehicle OEM. The owner provides the PIN to the friend via a dedicated channel, separate from the channel used to communicate the sharing invitation. The friend then types it into their device UI to continue the sharing workflow. Device OEM verifies if the correct PIN was entered by friend. Device OEM decides whether to implement this option or not.

11.2.1.4 Activation Options

Can be defined by the Vehicle OEM to activate a friend key in the vehicle. The friend can then choose any supported activation option. May contain the sharingPassword as one option. Vehicle OEM decides whether to implement this option or not.

1 **11.2.1.5 Approved Sharing Methods**

2 These are Device OEM-proprietary sharing channels under full control of the device OEM that
3 have been approved by the vehicle OEM. The security level can be assessed and maintained.
4 When approved as secure by a vehicle OEM then the sharing mechanism doesn't require
5 activation options to be used for a key shared using that method. Vehicle OEMs shall have a
6 way to define the activation method policy for approved sharing methods.

7 **11.2.1.6 Sharing Method Group Identifier**

8 Identifier defined by OEM that implements a proprietary sharing method. Vehicle OEMs use this
9 identifier to define the activation options policy for the referenced sharing method.

10 **11.2.2 Principles**

11 The Digital Key system is based on signature verification using asymmetric cryptography. The
12 vehicle unlocks the doors or starts the engine only if a challenge has been signed with a private
13 key corresponding to a public key registered in the vehicle.

14 After the owner device and the vehicle have been paired as described in Section 6, the owner
15 possesses a private key for which the corresponding public key is stored in the vehicle.

16 To allow friends to access the vehicle, the owner uses his/her private key to sign friend public
17 keys. On presentation of the owner's signature over the friend's public key, the vehicle will in
18 turn accept the friend as a new vehicle user and store the friend's public key.

19 Digital key sharing consists of the owner sending an invitation URL to the friend device and a
20 stateful exchange of key creation, key signing, and data import requests. When a secure sharing
21 channel, such as a device OEM-controlled proprietary messaging channel, is used, then
22 invitation and stateful sharing messages can all be sent across this channel. No additional second
23 factor verification (i.e., *activation options* and *device PIN*) of the destination device should be
24 required. These secure sharing methods are named *approved sharing methods* throughout this
25 specification, which means that they are approved by the vehicle OEM for use without additional
26 activation options. The approval process is out of scope of this specification. The list of approved
27 sharing methods can be pre-agreed between vehicle OEM and device OEM, or it can be sent
28 using key tracking and event notifications.

29 Digital Key Sharing between devices of different device brands (known as cross-platform
30 sharing), consists of:

- 31 1. Sending an invitation over a messaging channel that the owner uses regularly with the friend.
32 This provides good insurance that the invitation is received by the intended person. The
33 channel to exchange the stateful sharing messages is defined by [38].
 - 34 a. One or more Vehicle OEM-defined *activation options*, such as the *sharing password*
35 along with more options (E.g., presence of a key fob during the first usage of the friend
36 key), shall be applied if required by the vehicle OEM. Vehicle OEM provides these in the
37 trackKey response or eventNotification sent to owner device. Available *activation*
38 *options* are listed in Section 17.8.19. The list of *activation options* for a given vehicle is
39 communicated to owner and friend device during key tracking. In addition, the owner
40 device provides the list of *activation options* to the friend device during key sharing. The
41 activation options shall be shown in the owner and friend device UIs. Friend can select
42 and use one option from the list of activation options.

- 1 i. If the sharing password is one of the supported activation options, then the owner
2 device shall make the password available in the device UI along with necessary
3 guidance for usage. In case of a vehicle software update that adds or changes
4 activation options, owner device can be updated using event notifications.
- 5 2. If a vehicle OEM decides to not support additional *activation options*, the owner device
6 should propose the use of a *device PIN* (and potentially other “silent” verification methods in
7 the future) to owners that want to add additional verification to their shares. The policy for
8 device OEM-proprietary sharing methods is defined by the owner device OEM.
9 In some cases, (e.g., when list of *approved sharing methods* has been modified by the vehicle
10 OEM) the owner device shall send a *sharing method attestation* for shares using an approved
11 sharing method to confirm that the owner device uses the updated list of *approved sharing*
12 *methods*.
13 The conditions of change of the sharing policy (i.e., adding or removing approved sharing
14 methods from the list) by the vehicle OEM are out of scope of this specification.
15 The vehicle OEM shall notify the device OEM of a change to the activation policy using either
16 an event notification “SHARING_POLICY_CHANGED” or another device OEM proprietary
17 method.

18 **11.3 Communication Channel**

19 The standardized communication channel between devices and the relay server used by devices
20 and vehicles that implement cross-platform sharing shall use the stateful workflow method
21 defined in [38].

22 The referenced sharing channel has the following properties:

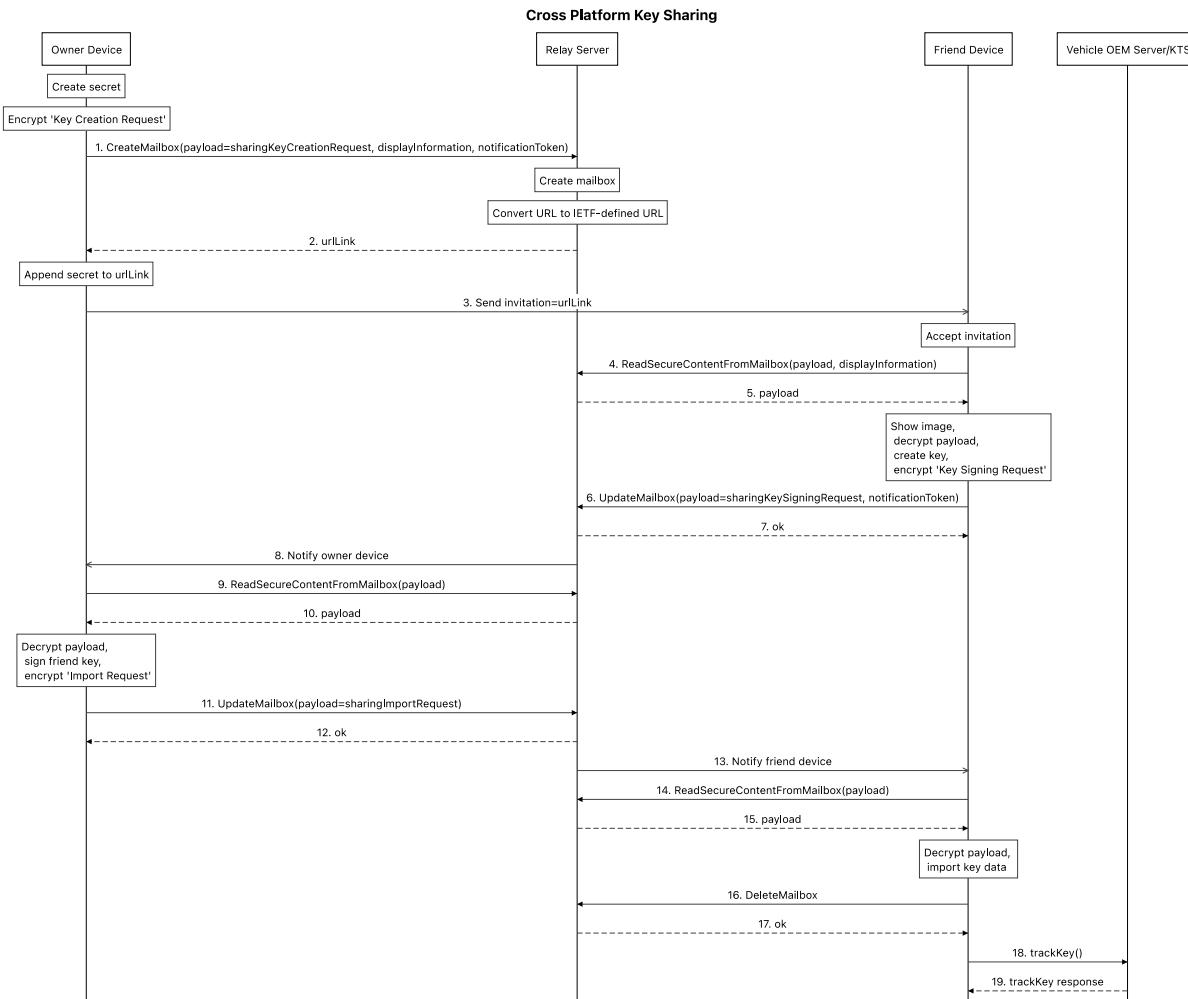
- 23 • Open standard for implementation on all Digital Key eligible devices and servers
- 24 • Sharing invitation can be sent over any messaging or chat channel
- 25 • Devices connect directly to relay server

26 In addition, the devices shall only send encrypted sharing data using the encryption key from the
27 sharing invitation.

28 The communication channel between an owner device and a friend device from the same OEM
29 is implementation-specific and out of scope of this specification but shall be listed as an
30 approved sharing method in Table 11-11, if key activation without vehicle OEM-defined second
31 factor activation option is intended. Figure 11-1 shows the detailed key sharing flow between the
32 owner device and friend device. The process follows the “Stateful Workflow” defined in [38]

33 Note: In Figure 11-1, the full head arrows represent synchronous requests, dashed arrows
34 represent responses to synchronous requests and open head arrows represent asynchronous
35 messages

1 Figure 11-1: Detail of Key Sharing Flow Between Owner and Friend Device After Channel is Established



2
3 Figure 11-1 does not show all steps or parameters, such as the full *payload* coding or the device
4 OEM server. It is meant to clarify the flow in principle and pointing to the relevant elements.
5 The exchanges between owner and friend device are described by the stateful workflow method
6 using the following HTTP access methods as defined in [38]. An application running on a device
7 may invoke the following APIs on Relay Server (see Section 11.3.4.1 to 11.3.4.6)

- 8 • CreateMailbox
- 9 • UpdateMailbox
- 10 • DeleteMailbox
- 11 • ReadDisplayInformationFromMailbox
- 12 • ReadSecureContentFromMailbox
- 13 • Relinquish Mailbox

14 The secure element is not involved in generating key material used for the sharing channel as
15 described in this section.

16 The key sharing flow and data is described in Section 11.8.

17

11.3.1 Notifications

2 The push notification feature shall be implemented by the Relay Server.
3 Notification tokens as described in [38] should be used by every device OEM.
4 If the notification tokens are not used, the devices shall implement a polling strategy for Relay
5 Server data. Devices may implement a (backup) polling strategy even if push notifications are
6 used.

7 The polling strategy should follow these recommendations:

- 8 - After sending the invitation, the owner device should poll every 10 seconds for a duration
9 of 3 minutes for the Signing Request (or other messages to be received as per Table
10 11-3), thereafter it should poll every 30 seconds for the next 10 minutes, thereafter it
11 should poll every 3 minutes for the next hour, thereafter every hour until the mailbox is
12 expired or deleted.
- 13 - After accepting the invitation and uploading the signing request to the mailbox, the friend
14 device should poll every 5 seconds for a duration of 1 minute for the Import Request (or
15 other messages to be received as per Table 11-3), thereafter it should poll every 30
16 seconds for the next 10 minutes, thereafter it should poll every 3 minutes for the next
17 hour, thereafter every hour until the mailbox is expired or deleted.

11.3.2 Cross-Platform Sharing Invitation

19 The owner device generates a Secret and encrypts the payload for the CreateMailbox API using
20 this Secret as described in [38].

21 The owner device then initiates the sharing session by calling the CreateMailbox API as
22 described in Section 11.3.4.1. The call returns the sharing invitation in form of a URL. The URL
23 is then sent to the friend over any communication channel (e.g., WhatsApp, etc.). The URL
24 formatting is described in [38]. The framework shall append the Secret as a fragment to the URL.

25 A relay server may be implemented by any CCC member. All CCC-approved relay server URLs
26 shall be listed in [35]. The approval process is defined in a separate document and is controlled
27 by the CCC. All Owner devices shall only use approved relay server URLs. Friend devices shall
28 only accept invitations containing an approved relay server URL. Friend devices shall support all
29 CCC-approved relay server URLs within a reasonable time frame as defined by the CCC in the
30 CCC Program Management Document [*PMD*].

31

11.3.3 General API Parameter Definitions

33 The *version* parameter shall be set as described in [38].

34 The *mailboxIdentifier* shall be used as described in [38].

35 The *deviceAttestation* should be used to authenticate the owner device if the Relay Server is not
36 hosted by the owner device OEM. In other cases, the device OEM might use proprietary methods
37 to authenticate the device. The friend device does not need to present a *deviceAttestation*.

38 The *deviceClaim* shall be provided and set by owner and friend device as described in [38].

39 The payload is encrypted as described in [38]. For Digital Keys payload shall contain the
40 provisioning information as the JSON structure with the keys *format* and *content* as per [38]. For

1 Digital Keys the format identifier is *digitalwallet.carkey.ccc*. The *content* data structure is
2 described in Table 11-1.

3 *Table 11-1: content in JSON Format*

Key	Type	Max Length (Bytes)	Description	Required
genericSharingData	Dictionary	4096	This is data shared between devices on different device OEM platforms.	Yes
<Device OEM>	Dictionary	4096	Device OEM proprietary provisioning information for the designated device OEM. Ignored by all other device OEMs.	Optional

- 4
- 5 <Device OEM> is defined in [35].
- 6 The owner device can add proprietary data (e.g., “apple”) that a friend device from the same
7 device OEM is able to use, e.g., to improve the user experience. If the friend device is from
8 another device OEM, then the friend device shall ignore the proprietary data. It shall be possible
9 for the owner device to add multiple <Device OEM> fields, each one for a different device
10 OEM, if required.
- 11 Note that all fields can be “seen” by friend devices of all platforms.
- 12 The genericSharingData is defined as the following JSON structure:

13 *Table 11-2: JSON formatted genericSharingData data structure*

Key	Type	Max Length (bytes)	Description	Required
sharingData	String	4096	Sharing data structure (TLV string) as listed in Table 11-3 based on the value of sharingDataType	Yes
sharingDataType	Integer	4	Enum for type of message. See sharingDataType in Table 11-3.	Yes
sharingId	String	64	ID of sharing invitation	Yes
friendKeyId	String	64	Identifier of key on friend device (this is used on owner side to track shared keys)	Conditional. Required for sharingDataType sharingKeySigningRequest.
authType	Array of strings	variable	“VehicleActivation” = activation options “DevicePIN” = PIN entered on friend device Other friend verification options can be added in the future.	Conditional. Required for sharingDataType sharingKeyCreationRequest and if an additional verification method is

Key	Type	Max Length (bytes)	Description	Required
				required by Owner Device or Vehicle OEM
activationOptions	Array of strings	variable	List of activation options as per section 17.8.19	Conditional. Required for sharingDataType sharingKeyCreationRequest if authType = “VehicleActivation”.
pinLength	Integer	4	Length of device PIN. Absent if device PIN is not used.	Conditional. Required for sharingDataType sharingKeyCreationRequest and if authType = “DevicePIN”.

- 1
- 2 *sharingData* contains the key sharing data elements as defined in Table 11-3, depending on the state of sharing.
- 3
- 4 *sharingDataType* is defined in Table 11-3.
- 5 *sharingId* is defined by the owner device in the first message to unambiguously identify the sharing session independently of the mailboxIdentifier. The value shall be unique per owner device and per session. Owner and friend device shall use the same value in all messages of the same sharing session. If the owner sends out multiple invitations (e.g., for friend phone and friend watch) then the values of *SharingId* shall be unique per friend device.
- 6 *friendKeyId* shall be sent by the friend device in the key signing request (format as defined in section 2.4 as “digital key identifier”)
- 7 *authType* indicates which of the following additional verification is required. If no method is used, then the field shall be absent. The additional verification methods are listed here:
- 8
- 9 • *VehicleActivation* usage shall be indicated if both of the following conditions are true:
 - 10 ○ At least one activation option has been provided to the owner device via uiBundle in trackKey response (see section 17.8.5) or eventNotification (see section 17.8.9)
 - 11 ○ A non-approved sharing method has been used
 - 12 • *DevicePIN* usage shall only be indicated if both of the following conditions are true:
 - 13 ○ No vehicle activation is required
 - 14 ○ The owner chooses to use the device PIN for a key sharing
- 15 It shall be possible to add other friend verification options, which are not part of the current list of activationOptions, in the future.
- 16 *activationOptions* are provided by the vehicle OEM in the trackKey response or eventNotification. Those options are specific to and implemented by the vehicle and may include the sharing password and/or alternatives to key activation via sharing password. Owner and friend device shall indicate them in the UI. If the field is absent, then no additional verification is

1 required by the vehicle OEM. In this case, where no activationOptions are provided by Vehicle
2 OEM, owner or device OEM can still require other verification methods not involving the
3 vehicle OEM, as described in authType.

4 *pinLength* shall be a value between 4 and 8 (including 4 and 8), chosen by the owner device. A
5 suitable combination of PIN length and allowed number of attempts to enter the device PIN in
6 the friend device UI shall be chosen by the owner device, following the recommendations in
7 [38].

8 *Table 11-3: sharingDataType enum Definition*

Key	Value
sharingKeyCreationRequest (Key Creation Request, see Table 11-5)	1
sharingKeySigningRequest (Key Signing Request, see Table 11-7)	2
sharingImportRequest (Import Request, See Table 11-16)	3
sharingOwnerCancel	4
sharingFriendCancel	5
sharingPinReEntryRequest (Table 11-8 in section 11.3.6.2)	6
sharingPinReEntryValue (Key Signing Request, see Table 11-7)	7
...unknown additional entries shall be ignored, reserved for forward compatibility.	

9
10 The sharing types *sharingOwnerCancel* and *sharingFriendCancel* shall be used on any error
11 condition instead of the next message of the sharing flow and shall contain an error message (see
12 Table 11-10) with the appropriate error code.

13 The sharing type *sharingPinReEntryRequest* signals the friend that the entered device PIN was
14 wrong. It contains the number of remaining attempts in the field *sharingData* in TLV format.

15 The sharing type *sharingPinReEntryValue* contains the key signing request with the new friend
16 device PIN.

17 Sample Provisioning Information in CreateMailbox Message is shown in Listing 11-1.

18 *Listing 11-1: Sample Provisioning Information in JSON Format*

```
1 {
2     "format": "digitalwallet.carkey.ccc",
3     "content": {
4         "genericSharingData": {
5             "sharingData": "7F31xxxx",
6             // key creation request, this field changes depending on the sharingDataType value
7             "sharingDataType": 1,
8             "sharingId": "123456789...",
9             "friendKeyId": "A2B686DFC...",
10            "authType": ["DevicePIN"]
11            "pinLength": 6
12        },
13        "apple": {
14            // ...
15        }
16    }
17 }
```

19

1 **11.3.4 API Parameter Usage**

2 The usage of API parameters from [38] that are relevant to Digital Key Cross Platform Sharing
3 are described in the following sections.

4 **11.3.4.1 CreateMailbox API Parameters**

5 The payload data contains *format* and *content* fields (see [38]). *content* contains
6 *genericSharingData* and optional device OEM-specific sharing data as described in Table 11-1
7 *sharingDataType* enum (Table 11-3) shall be set to 1 (*sharingKeyCreationRequest*) to indicate
8 that the content of *sharingData* is the key creation request as defined in Table 11-6.

9 The *displayInformation* is determined by the owner device. The format is described in [38].

10 The *notificationToken* is provided by owner device at the time of mailbox creation and is valid
11 for the lifetime of the mailbox. It should be used by the owner device to receive a notification
12 from the relay server when a Key Signing Request, sharing rejection or subsequent retry of
13 device PIN (in context of Key Signing Request) is received by the Relay Server from the friend
14 device. If notifications are not used, then the owner device shall perform regular polling as
15 described in Section 11.3.1 subsection “Notifications”.

16 The *mailboxConfiguration* shall be set as follows:

- 17 • *accessRights* = “RWD”
- 18 • *expiration* = <owner device OEM policy>

19 The *urlLink* is formatted as described in [37].

20 *isPushNotificationSupported* shall be set to “true” by all CCC relay servers, i.e., they shall be
21 supported by servers. Devices may choose to not use them although their use is strongly
22 recommended.

23 **11.3.4.2 UpdateMailbox API Parameters**

24 The payload contains *format* and *content*. *content* contains *genericSharingData* and device
25 OEM-specific sharing data as per Table 11-1.

26 *sharingData* from a friend device contains

- 27 • the Key Signing Request as per Table 11-7, or
- 28 • a *sharingFriendCancel* error message.

29 *sharingData* from an owner device contains one of the following:

- 30 • the Import Request as per Table 11-16, or
- 31 • a *sharingOwnerCancel* error message, or
- 32 • the Device PIN entry request message

33 The *notificationToken* should be provided by the friend device to be notified when data has been
34 received by the Relay Server from the owner. The parameters are defined by the friend device. If
35 notifications are not used, then the friend device shall perform regular polling as per Section
36 11.3.1 subsection “Notifications”.

37 **11.3.4.3 DeleteMailbox API Parameters**

38 The friend device shall delete the mailbox after successfully retrieving the Import Request.

- 1 If the owner receives a sharing cancellation from the friend (e.g., friend rejects the invitation),
2 the owner device shall delete the mailbox.
- 3 If the friend receives a sharing cancellation from the owner (e.g., owner does not sign the friend
4 key), the friend device shall delete the mailbox.
- 5 If the owner wants to cancel a sharing after the key has been signed, in addition to deleting the
6 mailbox a Remote Termination Request shall be used as outlined in Figure 13-4 [REV_140]. The
7 vehicle OEM shall add the friend key ID to a deny-list and reject key tracking (using substatus
8 code 50112), if the friend key is not yet registered.

9 *11.3.4.4 ReadDisplayInformationFromMailbox API Parameters*

10 The *displayInformation* posted during mailbox creation can be retrieved again if required.

11 *11.3.4.5 ReadSecureContentFromMailbox API Parameters*

12 The payload contains the payload uploaded during creation or update of the mailbox according to
13 the descriptions in the respective sections.

14 The *displayInformation* posted during mailbox creation is retrieved and used to display a
15 placeholder for the final key graphics on the friend device.

16 The *expiration* of the mailbox indicates the maximum lifetime of the mailbox if it is not deleted
17 before.

18 *11.3.4.6 RelinquishMailbox API Parameters*

19 This API might be called to allow a device (e.g., a tablet) that receives the sharing invitation but
20 cannot handle a digital key, to release the mailbox link and let another device connect to the
21 mailbox at outlined in [38]. This allows another recipient to claim the mailbox and use the
22 appropriate device to proceed with sharing procedures.

23 **11.3.5 Security Considerations**

24 The security considerations in [38] are generic. Some considerations how to apply them in the
25 context of Digital Key:

- 26 - The slot identifier ensures that a key creation request can lead to the creation and
27 acceptance of only one key based on this request
- 28 - If not using an approved sharing method, sharing should be protected by a second factor,
29 which can be either a vehicle OEM-defined second factor (i.e., activation option)
30 described in section 11.2.1.4 and later, if supported by the vehicle, or a device OEM-
31 defined second factor (e.g., the device PIN) as described in section 11.3.6.
- 32 - Sharing shall require explicit user authentication (as described in section 9.1 for RKE).
33 Explicit user authentication is the combination of user intent and user authentication.

34 **11.3.6 Device PIN**

35 *11.3.6.1 Concept*

36 All devices shall support Device PIN that is generated on the owner device and entered on the
37 friend device. Device PIN could be activated by the owner during the sharing workflow if no

1 other vehicle OEM-defined activation option is used, or if the sharing password is not part of the
2 activation options. Device UIs shall guide users to transfer the device PIN over a second channel
3 and guard against the user unintentionally re-using the same channel used to communicate the
4 sharing invitation. The friend then types the PIN into the friend device UI to continue the sharing
5 workflow. The owner device then verifies that the correct PIN was entered into the friend device
6 UI.

7 The owner device policy determines whether a sharing method requires the device PIN. The
8 owner device policy is determined by the owner device OEM and is out of scope. To ensure a
9 good user experience, it is recommended that multiple activation options that require user
10 interface interaction should not be activated at the same time. For example, sharing password
11 (controlled by vehicle OEM) and device PIN (controlled by device OEM)

12 Please see Section 11.2 entitled “Principles” about vehicle OEM-defined activation option and
13 device PIN policies.

14 11.3.6.2 Basic Flow

15 The owner device generates the reference device PIN (O_PIN). It sets the number of device PIN
16 entry attempts to a device OEM-defined value, considering the *pinLength* as described in Section
17 11.3.3.

18 The owner then sends the O_PIN over a second channel to the friend. This step shall be guided
19 by the device UI to ensure that the device PIN is transmitted while complying with restrictions in
20 11.3.6.1. The owner device decrements the number of attempts by one.

21 The owner device sends the initial number of attempts in the key creation request via tag 9F17h.
22 The presence of this tag indicates that the owner has requested the entry of device PIN to the
23 friend device.

24 *Table 11-4: Key Configuration*

Tag	Length (bytes)	Description	Field is	Domain Version
7F30 _h	variable	Key Configuration	Mandatory	V-OD-FW
D0 _h	1	Standard access profiles as defined in Table 11-5	Conditional, present if no proprietary profile is present. Either Tag D0 _h or D1 _h shall be present	V-OD-FW
D1 _h	variable	Proprietary access profile	Conditional, present if no standard profile is present. Either Tag D0 _h or D1 _h shall be present	V-OD-FW
51 _h	15 or 13	not_before, DER encoded GeneralizedTime (15 bytes in length) or UTCTime (13 bytes in length) as per RFC 5280 [2]	mandatory	V-OD-FW

Tag	Length (bytes)	Description	Field is	Domain Version
52 _h	15 or 13	not_after, DER encoded GeneralizedTime (15 bytes in length) or UTCTime (13 bytes in length) as per RFC 5280 [3]	mandatory	V-OD-FW
D3 _h	variable	Friendly name of the friend key. Limited to 30 characters	mandatory	V-OD-FW

1

2

Table 11-5: Standard Access Profiles

Profile	Type	Description
0	Full	Full access and Drive Capabilities
1	Access Only	Only car access, no drive capabilities
2	Access and Drive Restricted	Access and Drive with Restrictions
3	Car Delivery	Car Delivery Profile
4	Valet	Valet Parking Key
5	Vehicle Service	Vehicle Service Key
6 - 255	RFU	

3 Proprietary access profiles can be coded in a vehicle-specific method within tag D1_h.

4

Table 11-6: Key Creation Request

Tag	Length (bytes)	Description	Field is	Domain Version
7F31 _h	variable	Key Creation Request		N/A
		endpoint_configuration data field as described in Table 15-13 (without fields endpoint_identifier and instance_CA_identifier)	mandatory	V-OD-FW
		key_configuration data field as described in Table 11-4	mandatory	V-OD-FW
59 _h	variable	ROUTING_INFORMATION as generated during owner pairing described in Section 5.1.4.3	mandatory	V-OD-FW
5C _h	variable	DIGITAL_KEY_APPLET_PROTOCOL_VERSIONS as obtained during owner pairing described in Section 6 (This is the Digital Key applet protocol versions in which the owner received in SPAKE2+ REQUEST during owner pairing)	mandatory	V-OD-FW
7F60 _h	variable	Friend sharing configuration derived from owner SHARING_CONFIGURATION as generated during owner pairing described in Table 5-14	optional	N/A
DA _h	variable	If this tag is present, friend device shall obtain immobilizer token and/or slot identifier. The following values are assigned: If this tag is present and the value is empty or the value is 00 _h : immobilizer token and slot identifier from vehicle via owner device	optional	V-OD-FW

Tag	Length (bytes)	Description	Field is	Domain Version
		01 _h immobilizer token and slot identifier retrieved online 02 _h no immobilizer token required, slot identifier from vehicle via owner device 03 _h no immobilizer token required; slot identifier retrieved online		
DB _h	0	If this tag is present, friend device shall obtain a key tracking receipt as per Sections 11.8.5 and 11.8.6	optional	V-OD-FW
DC _h	0	If this tag is present, Vehicle OEM supports online attestation delivery feature as described in Section 11.10	optional	V-OD-FW
4A _h	3	2 bytes offset (unsigned big-endian) 1 byte length to be returned from confidential mailbox in AUTH1 response as obtained during owner pairing in Section 6	optional	V-OD-FW
4B _h	3	2 bytes offset (unsigned big-endian) 1 byte length to be returned from private mailbox in AUTH1 response as obtained during owner pairing in Section 6	optional	V-OD-FW
D8 _h	1	SHARING_PASSWORD_LENGTH as obtained during owner pairing in Section 6 (may be used by the friend device UI). If absent, no sharing password is required.	optional	V-OD-FW
54 _h	2 x m	m supported OD-FD-KS compatibility versions of owner device (ver.high ver.low), ordered highest to lowest	mandatory	OD-FD-KS
5E _h	Var.	V-D-BT DK Protocol Version lists of vehicle (as per Table 5-4)	conditional	V-D-BT
80 _h	2 x n	Supported_DK_Protocol_Version	conditional	V-D-BT
...unknown tags under 7F60 _h shall be ignored, reserved for forward compatibility				
9F17 _h	1	Number of device PIN entry attempts	Conditional. Required if device PIN is used	OD-FD-KS
9F18 _h	1	Device PIN length (in digits)	Conditional. Required if device PIN is used	OD-FD-KS
...unknown additional tags under 7F31 _h shall be ignored, reserved for forward compatibility				

1

2 If tag D8h is present, then the sharing password shall be used and tags 9F17h and 9F18h shall not be present. If tag D8h is absent, then tags 9F17h and 9F18h shall indicate the device PIN length in digits to the friend device, if device PIN is required.

- 1 If tag 7F60_h is absent, then the default friend sharing configuration is:

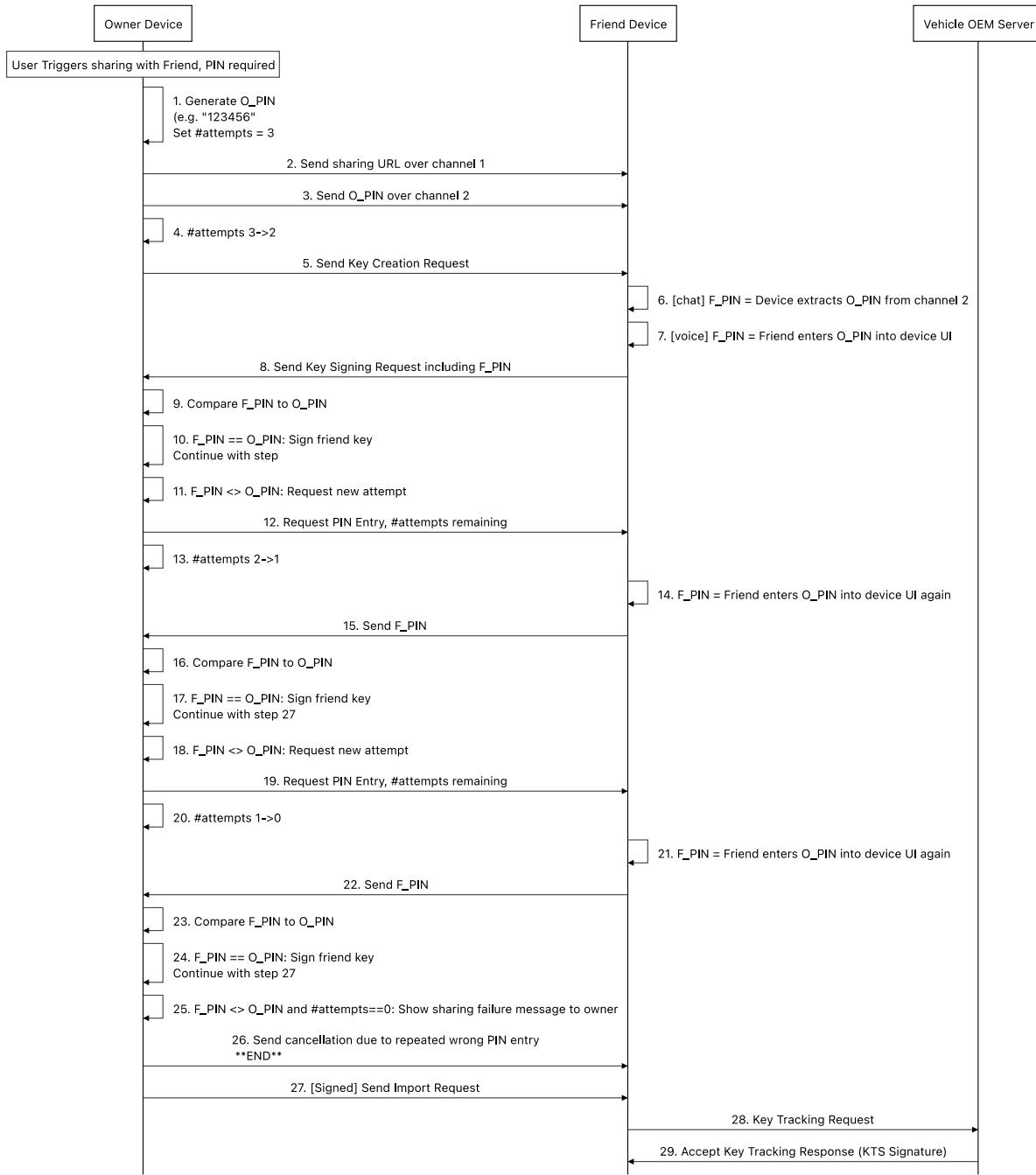
2 • DA_h: 00_h

3 • DB_h: present

4 • DC_h: not present

5 • D8_h: not present

Figure 11-2: Device PIN Flow



- 1 For a more focused view, Figure 11-2 does not show the device OEM server.
- 2 The friend extracts the O_PIN value from the second channel or receives it over a voice call.
- 3 This value is now named F_PIN. The friend enters the F_PIN into the friend device UI and the friend device adds the F_PIN to the key signing request.
- 4
- 5

Table 11-7: Key Signing Request

Tag	Length (bytes)	Description	Field is	Domain Version
7F36 _h	variable	Friend Key Signing Request		OD-FD-KS
		external CA Certificate container as per Table 11-12	mandatory	V-OD-FW
		instance CA Certificate container as per Table 11-13	mandatory	V-OD-FW
		endpoint certificate container as per Table 11-14 signed by instance CA private key	mandatory	V-OD-FW
		endpoint encryption key attestation as described in Table 15-47 signed by endpoint private key	conditional	V-OD-FW
55h	2 x n	n supported OD-FD-KS compatibility versions of friend device (ver.high ver.low), where n ≥ 16 for receivers and n ≤ 16 for transmitters. The agreed version is first in the list.	mandatory	OD-FD-KS
5F3F _h	variable	Friend-entered device PIN (F_PIN)	Conditional. Required if tags 9F17 _h and 9F18 _h have been sent in key creation request	OD-FD-KS

- 6
- 7 The owner device compares the received F_PIN with the O_PIN value. If both values are equal, then the owner device signs the friend key and sends the import request.
- 8
- 9 If the device PIN values are different, then the owner device sends a sharingPinReEntryRequest message (as per Table 11-8) to the friend. The sharingData structure is defined in Table 11-2.
- 10
- 11

Table 11-8: Device PIN Entry Request

Tag	Length (bytes)	Description	Field is
9F17 _h	1	Number of device PIN entry attempts	mandatory

- 12
- 13 The friend device allows the friend to enter the F_PIN again and sends the input back by sending a sharingPinReEntryValue message (as per Table 11-7) to the owner. The content is identical with the key signing request in Table 11-7, but the sharingDataType indicates that this is a repeated device PIN entry attempt.
- 14
- 15
- 16

1 The owner compares again F_PIN with O_PIN and, if equal, decides to sign or, if different, to
2 allow a new attempt to the friend device until either a correct F_PIN has been provided or the
3 number of remaining attempts is 0.

4
5 If the number of remaining attempts is 0, then the owner sends a cancel message (Table 11-3,
6 *sharingOwnerCancel*) indicating that no valid device PIN has been received in the defined
7 number of attempts.

8
9 If the friend device does not send back tag 5F3F_h (F_PIN) in the signing request, then the friend
10 device does not support the device PIN feature. The owner device policy for this case is defined
11 by the device OEM. Options could be to reject sharing to this device or propose to the owner to
12 share without device PIN, providing some security education.

13

14 **11.4 Certificate Chain**

15 The goal of the certificate chain is to allow the owner's SE to verify that a friend's endpoint is
16 authorized to participate in the sharing protocol. This verification is required to make sure that
17 the friend key pair has actually been generated on an approved SE, and to ensure that any secret
18 data transferred from owner's confidential mailbox to a friend's confidential mailbox shall be
19 handled as per the rules described in Section [4.3](#).

20 Each owner endpoint contains one or several authorized_PK set by the vehicle during the owner
21 pairing process as per Section [6](#). The authorized_PK list contains, at a minimum, the public key
22 of the Vehicle OEM CA. The authorized_PK list may only contain intermediate CA or root CA
23 but no leaf certificates. The External CA is typically the Device OEM CA used to issue the
24 Instance CA Certificate of Digital Key applet. The Instance CA in turn issues the endpoint
25 certificates.

26 In this version of the specification, only one single authorized_PK shall be set by the vehicle.
27 The verification of the friend endpoint is handled through the following verification chain. Each
28 element to the right of the arrow is attested by the element above it:

29 Authorized CA (Vehicle OEM CA)

30 —> External CA (Device OEM CA)

31 —> Instance CA

32 —> Endpoint

33 —> Endpoint Encryption Key

34 **11.5 Prerequisites**

- 35 1. The Digital Key applet helper methods described in Section 15.4 or equivalent
36 implementation are available on friend and owner device.
- 37 2. Owner and friend devices contain the necessary key material to authenticate remote
38 servers. This key material is stored on the Digital Key framework.

3. Owner and friend Digital Key applet or equivalent implementation possess an Instance CA dedicated to the Vehicle OEM brand involved in sharing. The issuance of Instance CA is described in Section 16.
4. Owner and friend devices have retrieved an External CA certificate as defined in Section 16.2.6. This certificate is stored in the Digital Key framework of both devices.
5. The owner pairing has been successfully executed on the owner device, as per Section 6.
6. If applicable, immobilizer tokens and slot identifiers are available on the owner device's confidential and private mailboxes, as per Section 6.
7. The following values have been obtained by the owner, as per Section 6 or equivalent preliminary operations. These values are stored on the Digital Key framework:
 - (a) IMMOBILIZER_TOKEN_LENGTH
 - (b) SIGNALING_BITMAP_OFFSET
 - (c) SLOT_IDENTIFIER_BITMAP_OFFSET
 - (d) SLOT_IDENTIFIERS_OFFSET
 - (e) VEHICLE_OEM_PROPRIETARY_DATA_OFFSET
 - (f) ATTESTATION_PACKAGE_OFFSET
 - (g) MAILBOX_CONTENT_END_OFFSET
 - (h) LONG_TERM_SHARED_SECRET
 - (i) SHARING_PASSWORD_LENGTH
 - (j) ROUTING_INFORMATION
 - (k) DIGITAL_KEY_APPLET_PROTOCOL VERSIONS
 - (l) SHARING_CONFIGURATION
8. Owner and friend OEM Server have implemented the Key Life Cycle Management functions especially key termination and deletion as described in section 13.

11.6 Key Configuration

A TLV encoded schema defines the Digital Key configuration with available access profiles and their encoding. This schema is described in Table 11-4. Note that prior versions of this specification used ASN.1 formatting for the Digital Key configuration.

11.7 Error Codes

The following codes may be sent from owner device to friend device or vice versa if an error occurs during the sharing process. The sharing procedure shall be aborted if an error is returned by friend or owner.

The error codes are returned in the TLV structure as shown in Table 11-9. All error codes that are unknown to the receiver shall be treated as FF_h (unspecified error).

Table 11-9: Error Code Message

Tag	Length (bytes)	Description	Field is	Domain Version
7F6C _h	variable	Error Message		OD-FD-KS
6D _h	1	Error code as per Table 11-10	mandatory	

1

Table 11-10: Error Codes

Error code	Description
00 _h	Missing mandatory field
01 _h	Invalid message structure
02 _h	Invalid message content
03 _h	Version not supported
04 _h	Certificate expired
05 _h	Invalid certificate structure
06 _h	Invalid certificate content
07 _h	Invalid certificate chain
08 _h	Request rejected
09 _h -1F _h	RFU
20 _h	Sharing cancelled
21 _h	Sharing cancelled by owner – no valid Pin received after max number of attempts
22 _h	Sharing cancelled by friend
23 _h -FE _h	RFU
FF _h	Unspecified error

- 2 **11.8 Key Sharing Flow: Steps**
- 3 Figure 11-1 shows the main key sharing flow and describes the content of the data packets
4 exchanged between the owner and friend devices independently of the communication channel
5 or transport layer used to convey these packets.
- 6 At the end of this flow, the friend's private mailbox contains an attestation package in which the
7 friend's public key is signed by the owner private key along with a set of entitlements. The
8 friend's confidential mailbox optionally contains an active immobilizer token.
- 9 The last step describes how the attestation package is delivered and verified by the vehicle
10 during the first friend transaction.

11 **11.8.1 Steps 1 and 2 (Owner): Sharing Invitation**

- 12 The owner device sends a Key Creation Request containing the required endpoint configuration
13 and the entitlements to be granted. The endpoint_identifier and instance_CA_identifier from
14 field endpoint_configuration shall not be included since their content is defined by the friend
15 device.
- 16
- 17

1

Table 11-11: Approved Sharing Methods

Tag	Length (bytes)	Description	Field is
7F10 _h	variable	list of approved sharing methods	
61 _h	variable	Tuple of first sharing method provider identifier and first sharing method group identifier	conditional
40 _h	2	Sharing method provider identifier as defined in XXX (e.g., 0x0000 = CCC, 0x0001 Apple, 0x0002 = Google, 0x0003 = OEM 3...)	mandatory
41 _h	variable	Sharing method group identifier defined by the sharing method provider	mandatory
61 _h	variable	Tuple of second sharing method provider identifier and second sharing method group identifier	optional
40 _h	2	Sharing method provider identifier as defined in XXX (e.g., 0x0000 = CCC, 0x0001 Apple, 0x0002 = Google, 0x0003 = BMW...)	optional
41 _h	variable	Sharing method group identifier defined by the sharing method provider	optional

2

3

Listing 11-2: Sharing Invitation

```

1 input: user action
2 output: key_creation_request
3 begin
4 generate a key_configuration field as per Table 11-4, owner might be prompted with a specific UI for this action.
5 key_configuration.updateCounter ← 0
6
7 if immobilizer token to be retrieved from the owner device (i.e. Tag DAh in Table 5-14 is not present, or if Tag DAh is present
8 and the corresponding value is empty or is set to 00h)
9 slot_identifier ← select the immobilizer token to be shared and save its corresponding slot identifier
10 else if no immobilizer tokens are used, and slot identifier is retrieved from the owner device (i.e., Tag DAh in Table 11-5 is
11 present and its value is set to 02h)
12 slot_identifier ← save slot identifier
13 generate an endpoint_configuration field as described in Table 15-13 with
14 endpoint_configuration ← copy owner's endpoint_configuration
15 endpoint_configuration ← remove endpoint_identifier and instance_CA_identifier fields
16 endpoint_configuration.option_group_1.bit4 ← 0
17 endpoint_configuration.option_group_1.bit5 ← 0
18 endpoint_configuration.endpoint_identifier ← see naming rules for friend key (see Section 4.2)
19
20 if confidential_mailbox_size field present in owner's endpoint configuration
21 endpoint_configuration.confidential_mailbox_size ← 1 × IMMobilizer_TOKEN_LENGTH
22 if slot_identifier to be retrieved from the owner device
23 endpoint_configuration.key_slot ← slot_identifier
24
25 generate key_creation_request as per Table 11-6 using key_configuration, endpoint_configuration,
26 ROUTING_INFORMATION (, friend sharing configuration)
27
28 if sharing password required by vehicle OEM policy or device policy (see Section 11.11)
29 sharing_password_seed ← generate random bytes
30 generate sharing_password as per Listing 11-12 using
31 LONG_TERM_SHARED_SECRET

```

```
30     SHARING_PASSWORD_LENGTH
31         sharing_password_seed
32         store the sharing_password and sharing_password_seed on owner device
33         the owner device can display the sharing_password with specific distribution instructions
34
35         send key_creation_request to friend over owner-friend-com
36     end
```

1 *11.8.1.1 Versioning*

2 If non-backward compatible changes in the key creation request (Table 11-6) are necessary, then
3 a new tag (e.g., 7F33_h) shall be defined and both data structures shall be sent to the friend device,
4 which can then pick the tag it is able to process.

5 Related applet commands:

6 - CREATE ENDPOINT (friend)

7 For forward compatibility, the friend device shall accept other tags from the owner device under
8 7F31_h in the Key Creation Request. If not known, these tags shall be ignored by the friend
9 device. In future versions, the owner device may send other tags.

10 If Tag 54_h in Table 11-6 is recognized, then the friend device shall determine a commonly
11 supported key sharing version (OD-FD-KS), send its supported version numbers back in the Key
12 Signing Request (see section 11.8.1.1) and continue the key sharing process according to the
13 determined version number. If the tag is not recognized, the friend device shall behave as prior to
14 version introduction.

15 The endpoint configuration data shall contain all data elements from older versions to be
16 backward compatible. New tag shall be used if format change or extension is needed.

17 The friend device shall reject the sharing if the ROUTING_INFORMATION format cannot be
18 handled by the device OEM server, which processes the data. This data format is determined
19 before the sharing starts, so its version or format cannot be changed.

20 The friend device shall reject the sharing or ask the user to update if
21 DIGITAL_KEY_APPLET_PROTOCOL VERSIONS are not compatible with the friend applet
22 protocol version. This version list is determined by the vehicle, the device needs to find a
23 matching version from that list.

24 The friend device shall reject the sharing or ask the user to update if any version of the set of
25 vehicle version lists in tag 5E_h is not compatible with the supported friend BT versions.

26 Note that tags 4A_h, 4B_h and D8_h are not likely to change their format. If this becomes necessary,
27 new tags shall be defined and sent in addition.

28 *11.8.2 Steps 3 and 4 (Friend): Key Signing Request*

29 The friend device creates an endpoint using the provided endpoint_configuration, then transfers
30 to the owner a Key Signing Request containing the endpoint certificate along with the associated
31 certificate chain.

32 The friend device uses the list of authorized_PK provided by the owner in the
33 endpoint_configuration field in order to produce a certificate chain starting from one of these
34 public keys. The created friend endpoint certificate shall not contain any authorized_PK.

1

Table 11-12: External CA Certificate Container			
Tag	Length (bytes)	Description	Field is
7F20 _h	variable	external CA certificate as per Listing 15-13 verifiable by an authorized_PK	mandatory

2

Table 11-13: Instance CA Certificate Container			
Tag	Length (bytes)	Description	Field is
7F22 _h	variable	instance CA Certificate as per Listing 15-16 Note: applet_version and platform_information are immutable values assigned at the time of the Instance CA certificate is issued. It might not reflect the current applet_version or platform_information (e.g., in case of applet or Secure Element platform upgrade). Listing 15-16 signed by external CA private key	mandatory

3

Table 11-14: Endpoint Certificate Container			
Tag	Length (bytes)	Description	Field is
7F24 _h	variable	endpoint certificate as Listing 15-5 signed by instance CA private key	mandatory

4

Listing 11-3: Key Creation Processing	
1	input: key_creation_request
2	output: friend_key_signing_request
3	begin
4	extract key_configuration, endpoint_configuration from key_creation_request
5	check the applet installed on platform supports at least one of the protocol versions present in DIGITAL_KEY_APPLET_PROTOCOL VERSIONS
6	check proposed entitlements received from owner comply with friend's policy, friend may be prompted with a specific UI for this action.
7	friend may decline sharing and stop the sharing process at this point.
8	check endpoint_configuration complies with friend's policy
9	check ability to produce a certificate chain starting from one of the authorized_PK
10	present in endpoint_configuration
11	
12	
13	execute framework.getInstance as described in Section 15.4.1.3
14	output: instance
15	
16	generate and add endpoint_identifier, instance_CA_identifier to endpoint_configuration as per friend device's local policy
17	Remove all element from authorizedPK list from endpoint_configuration
18	execute instance.createEndpoint as described in Section 15.4.1.7
19	input: endpoint_configuration,
20	bool_onlineCertificate = false
21	output: endpoint
22	
23	
24	execute endpoint.getCertificate as described in Section 15.4.1.12
25	input: bool_onlineCertificate = false
26	output: endpoint_certificate
27	

```

28 if tag DAh present in key_creation_request and set to 01h
29
30     execute endpoint.createEncryptionKey as described in Section 15.4.1.15
31     output: encryption_key_attestation
32
33     generate friend_key_signing_request as per Table 11-7
34     using external_ca_certificate,
35         instance_ca_certificate,
36         endpoint_certificate,
37         (encryption_key_attestation)
38
39 else
40     generate friend_key_signing_request as per Table 11-7
41     using external_ca_certificate,
42         instance_ca_certificate,
43         endpoint_certificate
44
45 send friend_key_signing_request to owner over the friend device to owner device link
46 end

```

11.8.2.1 Versioning

Related applet commands:

- AUTHORIZE ENDPOINT (owner)

The friend device generates the certificates and attestations in the key signing request (Table 11-7) using the highest version that is compliant with the agreed key sharing version (OD-FD-KS). It then sends the key signing request with the added list of supported friend versions. The selected, commonly supported version is first in the list.

11.8.3 Steps 5 and 6 (Owner): Generate Import Request

The owner generates an attestation over the friend's public key and optionally exports an immobilizer token from the owner endpoint's confidential mailbox. The owner sends an Import Request to the friend.

The owner shall generate a sharing method attestation and include it into the import request unless agreed otherwise with the vehicle OEM.

Table 11-15: Sharing Method Attestation Arbitrary Data to be Hashed and Signed

Tag	Length (bytes)	Description	Field is
61 _h	variable	Tuple of used sharing method provider identifier and used sharing method group identifier	conditional
40 _h	2	Sharing method provider identifier as defined in XXX (e.g., 0x0000 = CCC, 0x0001 Apple, 0x0002 = Google, 0x0003 = BMW...)	mandatory
41 _h	variable	Sharing method group identifier defined by the sharing method provider	mandatory
51 _h	20	Friend key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey	mandatory

1

Table 11-16: Import Request

Tag	Length (bytes)	Description	Field is	Domain Version
7F32 _h	variable	Mailbox Import Request		OD-FD-KS
		key_attestation as described in Table 15-60 (containing entitlement data as per Table 11-17, and arbitrary_data in Table 15-23)	mandatory	V-OD-FW
		the mailbox_mapping table referenced by tag 7F4D _h as described in Table 5-13	mandatory	V-OD-FW
4A _h	variable	Encrypted confidential mailbox data	conditional	V-OD-FW
97 _h	65	Encryption public key of the owner prepended by 04 _h	conditional	V-OD-FW
7F49 _h	variable	This tag, defined in Table 19-84, provides configuration data for the supported radios	Conditional. Shall be present if the vehicle supports WCC2/WCC3	V-OD-FW
7F2D _h	variable	SharingMethodAttestation Arbitrary data attestation (as per Table 15-61) with owner key using arbitrary data = SHA-256 hash value of Table 11-15	optional	
7F11 _h	variable	Content of Table 11-15 without tag 51 _h .	Conditional. Present when 7F2D _h is present.	

2

3

Table 11-17: Entitlement Data

Tag	Length (bytes)	Description	Field is	Domain Version
Key configuration as per Table 11-4			mandatory	Informative
4E _h	variable	slot identifier	Conditional: present if Tag DA _h in Table 11-6 is empty, or set to values 0x00 or 0x02	V-OD-FW
44 _h	8	sharing_password_seed	Conditional: present if second factor authentication is required by vehicle OEM	V-OD-FW
D7 _h	1	second_factor_activation_required (01 _h =required for non-approved sharing methods, other=RFU)	Conditional: present if second factor activation is required for non-approved	V-OD-FW

Tag	Length (bytes)	Description	Field is	Domain Version
			sharing methods by the vehicle OEM	
4D _h	8	vehicle_identifier	Conditional: present if Tag DA _h in Table 11-6 is set to values 0x01 or 0x03	V-OD-FW

- 1 If slotIdentifier is obtained from vehicle (via Owner Device) in keyCreationRequest, then slotIdentifier identified in Table 11-17 shall equal the value set during keyCreationRequest (Table 11-6) and vehicleIdentifier shall be absent.
- 4 If friend device obtains the slotIdentifier online in keyCreationRequest then the slotIdentifier shall be obtained later through the Key Tracking Response as per Section 11.10.1

6 *Listing 11-4: Import Request*

```

1   input: friend_key_signing_request
2   output: import_request
3   begin
4     extract from friend_key_signing_request
5       externalCACertificate,
6       instanceCACertificate,
7       endpointCertificate,
8       (encryptionKeyAttestation)
9
10    retrieve key_configuration sent in step1
11    verify endpointCertificate has been created as per step1 and step2 rules, otherwise abort procedure
12
13    generate an entitlement_data field as per Table 11-17
14    If key_slot present in key_creation_request sent in step1
15      entitlement_data.slotIdentifier ← key_slot
16      entitlement_data.vehicleIdentifier ← vehicle_identifier
17      entitlement_data.keyConfiguration ← key_configuration
18
19      if second_factor_authentication_required by Vehicle OEM policy or device policy (see Section 11.12)
20        entitlement_data.sharingPasswordSeed ← sharing_password_seed
21        entitlement_data.second_factor_authentication_required ← true
22      else
23        Entitlement_data.second_factor_authentication_required ← false
24
25      execute framework.getInstance as per Section 15.4.1.3
26      output: instance
27
28      execute instance.getEndpoint as per Section 15.4.1.9
29      output: endpoint
30
31      I ← index of the slot identifier selected in step 1
32      clearPrivateMailboxBit(endpoint, i, SLOT_IDENTIFIER_BITMAP) as per Listing 11-9
33      setPrivateMailboxBit(endpoint, SLOT_IDENTIFIER_BITMAP_BIT, SIGNALING_BITMAP_OFFSET) as per Listing 11-8
34
35      if immobilizer token required
36        immo_token_extraction_offset = i × IMMOBILIZER_TOKEN_LENGTH
37        immo_token_extraction_length = IMMOBILIZER_TOKEN_LENGTH
38      else
39        immo_token_extraction_offset = 0
40
41

```

```
42     immo_token_extraction_length = 0
43
44     execute endpoint.authorize as per Section 15.4.1.14
45     input:
46         u16_offset = immo_token_extraction_offset
47         u16_length = immo_token_extraction_length
48         bytes_arbitraryData = entitlement_data
49         bytes_externalCACertificate = externalCACertificate
50         bytes_instanceCACertificate = instanceCACertificate
51         (bytes_encryptionKeyAttestation = encryptionKeyAttestation)
52         bytes_endpointCreationCertificate = endpointCertificate
53         output: bytes_attestation(, bytes_encryptedData, bytes_encSenderePK)
54
55     prepare import_request buffer as per Table 11-16 using
56     bytes_attestation,
57     (bytes_encryptedData),
58     (bytes_encSenderePK)
59
60     send import_request to friend over the owner device to friend device link
61 end
```

1 *11.8.3.1 Versioning*

2 Related applet commands:

- 3 - CREATE ENCRYPTION KEY (owner)
- 4 - SET CONFIDENTIAL DATA (friend)

5 The import request includes data from different sources.

6 The key attestation format has been pre-agreed between owner device and vehicle (V-OD-FW).
7 Its version is not relevant for the friend device.

8 The mailbox mapping table is versioned by V-OD-FW, the friend device shall check if the
9 version of the mapping table can be processed.

10 The confidential mailbox data format and encryption public key is also pre-agreed by V-OD-FW.
11 The encryption algorithm is defined by the SET CONFIDENTIAL DATA API and shall be
12 considered when determining by the OD-FD-KS version.

13 The configuration data for the supported radios is based on Table 19-84, which supports adding
14 new tags for backward compatibility.

15 *11.8.4 Step 7 (Friend): Endpoint Data Import*

16 The key_attestation provided by the owner or the Vehicle OEM Server, shall be written into the
17 friend's private mailbox, and optionally the immobilizer token may be written into the friend's
18 confidential mailbox. The friend's private mailbox has the same mapping as the owner mailbox.
19 The friend's confidential mailbox contains the same or fewer immobilizer tokens than the owner
20 confidential mailbox. The key slot identifier shall only be included by the friend device when
21 slot identifiers are retrieved online.

1

Table 11-18: Attestation Package

Tag	Length (bytes)	Description	Field is	Domain Version
7F35 _h	variable	Attestation Package		V-OD-FW
41 _h	1	Certificate version = 02 _h	mandatory	V-OD-FW
92 _h	8	Random as described in Table 15-23	mandatory	V-OD-FW
5A _h	65	Public key from friend endpoint prepended by 04 _h	mandatory	V-OD-FW
78 _h	variable	Entitlements data as per Table 11-17	mandatory	V-OD-FW
9E _h	64	Signature with owner endpoint private key over Section Table 15-23 as per Listing 11-2	mandatory	V-OD-FW
4E _h	1-8	key_slot_identifier – only present if key slot identifiers are retrieved online (i.e., tag DA _h in Table 11-6 is present and set to 01 or 03)	conditional	V-OD-FW
45 _h	variable	key_tracking_receipt	conditional	V-OD-FW

2

Listing 11-5: Friend, Endpoint Data Import Processing

```

1 input: import_request (Table 11-16)
2 output: n/a
3 begin
4   extract tags content from import_request as separate buffers:
5     key_attestation
6     mailbox_mapping
7     (encrypted_confidential_mailbox_data),
8     (encryption_public_key),
9
10  extract friend private mailbox mapping offsets from mailbox_mapping
11
12  execute framework.getInstance as per Section 15.4.1.3
13    output: instance
14
15  execute instance.getEndpoint as per Section 15.4.1.9
16    output: endpoint
17
18  if key tracking required as per tag DBh from Table 11-6 or
19    online attestation delivery is supported as per tag DCh from Table 11-6
20    Execute Listing 11-7 from Section 11.10 to retrieve response
21
22  prepare attestationPackage buffer as per Table 11-18 using contents of:
23    key_attestation,
24    (key_tracking_receipt)
25
26  execute endpoint.setPrivateData as per Section 15.4.1.18
27    input: offset = ATTESTATION_PACKAGE_OFFSET, data = attestationPackage
28
29  setPrivateMailboxBit(endpoint, ATTESTATION_PACKAGE_BIT, SIGNALING_BITMAP) as per Listing 11-8
30
31  from that point a notification indicating success of the online attestation delivery can be received from server
32  on reception of such notification the following cleanup procedure is executed:
33    clearPrivateMailboxBit(endpoint, ATTESTATION_PACKAGE_BIT, SIGNALING_BITMAP) as per Listing 11-9
34    execute endpoint.setPrivateData as per Section 15.4.1.18

```

```
35     input: offset = ATTESTATION_PACKAGE_OFFSET, data = zeroes
36
37     if tag 4Bh was received in Step 1
38         execute endpoint.setParameters with the following parameters:
39             input: offset_private = offset extracted from tag 4Bh,
40                 length_private = length extracted from tag 4Bh
41
42     if tag DAh from Table 11-6 was present in Step 1, and is empty or equal to 00h or 01h
43         execute endpoint.setConfidentialData as per Section 15.4.1.16
44         input:
45             bytes_encsenderepk = encryption_public_key
46             bytes_encdata = encrypted_confidential_mailbox_data
47             u16_offset = 0
48             u16_length = IMMOBILIZER_TOKEN_LENGTH
49
50     if tag 4Ah was received in Step 1
51         execute endpoint.setParameters with the following parameters:
52             input: offset_confidential = offset extracted from tag 4Ah,
53                 length_confidential = length extracted from tag 4Ah
54         setPrivateMailboxBit(endpoint, 0, SLOT_IDENTIFIER_BITMAP) as per Listing 11-8
55
56 end
```

11.8.5 Steps 8 and 9 (Friend): Track Key

- 2 If key tracking is required by the Vehicle OEM, the friend device in Step 9 shall send trackKey
3 (see Section [17.7.1.2](#)) to the Vehicle OEM Server via the friend Device OEM Server to obtain a
4 key tracking receipt. Note that step 8 is shown for illustration purposes only.
5 If slot identifier is retrieved online, the friend device shall keep the key disabled on the NFC and
6 BLE interfaces until a valid slot identifier is available. The key shall be usable on the NFC and
7 BLE interfaces only after a successful trackKeyResponse(see Section 17.7.1.3) containing the
8 slot identifier.

11.8.6 Step 10 (Vehicle OEM Server): Track Key Response

- 10 In step 10, the Vehicle OEM Server shall send trackKeyResponse (see Section [17.7.1.3](#)) to the
11 friend Device OEM Server. Note that step 11 is shown for illustration purposes only.

11.8.7 Step 14 (Friend): First Transaction

- 13 This section describes the operations occurring between friend device and vehicle during the first
14 presentation of a friend endpoint on which operations described in Section [11.8.3.1](#) have been
15 successfully completed.

16 *Listing 11-6: First Friend Transaction*

```
1 begin
2     vehicle and device execute the standard transaction as described in Section 7 until
3         AUTH1 response from the device
4         device AUTH1 response contains: (as described in Section 15.3.2.10)
5             the key_slot
6             the endpoint_signature
7             (confidential_mailbox_subset: offset = 0, length = IMMOBILIZER_TOKEN_LENGTH)
8             private_mailbox_subset:offset = SIGNALING_BITMAP_OFFSET, length = 1
9             vehicle extracts signaling_bitmap ← private_mailbox_subset[0]
```

```
10  if immobilizer token required by vehicle
11    vehicle extracts immobilizer_token ← confidential_mailbox_subset[0]
12    vehicle verifies if ATTESTATION_PACKAGE_BIT in signaling_bitmap is not set
13      no attestation package is present, exit and process as regular transaction
14    vehicle computes attestation_package_length ← (END_OFFSET-- ATTESTATION_PACKAGE_OFFSET)
15    vehicle issues one or multiple EXCHANGE commands to retrieve the attestation package
16    if vehicle signature verification of attestation package with owner key is successful
17      vehicle verifies key has not been registered yet, otherwise abort
18      vehicle verifies value of slot_identifier is valid, otherwise abort (conditional, this step is not performed if slot identifier is
retrieved online)
19      vehicle verifies all elements of the attestation package are compliant with its policy, otherwise abort
20      vehicle registers endpoint.PK and entitlements
21      vehicle verifies the endpoint_signature received in AUTH1
22  else
23    abort procedure
24  vehicle clears ATTESTATION_PACKAGE_BIT in signaling_bitmap
25  vehicle clears the attestation package from device private mailbox using the one or multiple EXCHANGE commands
26
27  using value from attestationPackage.second_factor_authentication_required and information present in key tracking receipt
28  the vehicle may require the sharing_password generated and verified as
29    sharing_password_seed ← vehicle extracts from attestationPackage
30    vehicle generates sharing_password as per Listing 11-12 using
31      LONG_TERM_SHARED_SECRET
32      SHARING_PASSWORD_LENGTH
33      sharing_password_seed
34    vehicle verifies the sharing_password by requesting friend input on a specific vehicle UI
35    if vehicle supports feature from Section 11.11 and sharing_password is successfully presented
36      vehicle generates vehicleAttestation as per Section 11.11 and sends it to Vehicle OEM Server
37  End
```

1
2 The vehicle might require additional activation of a shared key to start the engine. In this case the
3 activation options have been provided to the owner in the key tracking request or via event
4 notification. The owner has provided the list of activation options to the friend during key
5 sharing.

6 11.9 Owner Device OEM Server Notification

7 Following a successful activation of the shared key, the Vehicle OEM Server shall send an
8 event-Notification SHARED_KEY_ADDED (see Section [17.7.3.2](#)) to the owner Device OEM
9 Server. This may occur before or after the first friend contact.

10 11.9.1 Step 12 (Vehicle OEM Server): eventNotification

11 In step 12, the Vehicle OEM Server shall send eventNotification (see Section [17.7.3.2](#)) to the
12 owner Device OEM Server.

13 11.9.2 Step 13 (Owner Device OEM Server): eventNotificationResponse

14 In step 13, the owner Device OEM Server shall send eventNotificationResponse (see Section
15 [17.7.3.3](#)) to the Vehicle OEM Server.

11.10 Key Tracking and Online Attestation Delivery

If required by the Vehicle OEM, a key tracking receipt computed over elements from [Table 11-18](#) shall be verified by the vehicle before accepting a new key. In that case, the key sharing flow requires an additional interaction with a Vehicle OEM Server.

The key tracking receipt may also contain proprietary instructions from the Vehicle OEM requesting the vehicle to verify the sharing password. The Vehicle OEM should consider the consistency for sharing password required in Digital Key creation and key tracking response.

If supported by the Vehicle OEM, the attestation package provided may also be sent directly to the vehicle via the Vehicle OEM telematic link (if the vehicle is reachable) in order to speed up the first friend transaction. This feature is called Online Attestation Delivery.

The cryptographic algorithm and final data elements used by the KTS to produce the receipt are specific to each Vehicle OEM and are out of scope of this specification.

Section [17.7.1](#) (API – Track Key) describes how the data elements highlighted in the paragraphs above are packaged and sent to the Vehicle OEM. This function assumes the Vehicle OEM Server possesses the owner's public key in order to verify the signature of the incoming packets.

Before the key tracking has taken place, a termination request may be received by the vehicle OEM server. In such a case, a key tracking receipt shall not be sent in response.

The Vehicle OEM Server may store the friend's endpoint public key and the Instance CA public key. The Device OEM may store an immutable unique device identifier for a given Instance CA public key, such that if the two parties cooperate, it is possible to link an endpoint public key with an immutable unique device identifier for legal and investigational purposes.

Table 11-19: Key Tracking and Online Attestation Delivery Request

Tag	Length (bytes)	Description	Field is	Domain Version
7F38 _h	variable	Friend Key Tracking and Online Attestation Delivery Request attestation package as described in Table 11-18 without tag 45 _h (key tracking receipt)	mandatory	V-OD-FW
		friend endpoint certificate container as per Table 11-14 signed by instance CA	mandatory	V-OD-FW
		friend Instance CA Certificate container as per Table 11-13 signed by external CA	mandatory	V-OD-FW
		endpoint encryption key attestation as per Table 15-47 signed by endpoint private key (present only if immobilizer token is required)	conditional	D-VS
5F49 _h	65	Device privacy encryption key (Device.Enc.PK)	mandatory	D-VS
DA _h	variable	Device privacy encryption version, Default "ECIES_v1" (ASCII)	mandatory	D-VS
7F2D _h	variable	SharingMethodAttestation Arbitrary data attestation (as per table 15-61) with owner key using arbitrary data = SHA-256 hash value of Table 11-15	Conditional. Friend device shall provide the attestation if it has been received from owner device	D-VS

Tag	Length (bytes)	Description	Field is	Domain Version
7F11 _h	variable	Content of Table 11-15 without tag 51 _h .	Conditional: present when 7F2D _h is present.	D-VS

1 *Table 11-20: Key Tracking Response*

Tag	Length (bytes)	Description	Field is	Domain Version
4E _h	1–8	Slot_identifier to be used by friend endpoint	conditional	V-OD-FW
45 _h	variable	Key tracking receipt	conditional	V-OD-FW
4A _h	variable	Encrypted confidential mailbox data (immobilizer token present only if immobilizer token is required)	conditional	D-VS
97 _h	65	Ephemeral encryption public key of the Vehicle OEM Server prepended by 04 _h (present only if immobilizer token is required)	conditional	D-VS

- 2 Endpoint encryption key attestation (Tag 7F26_h) shall be included in [Table 11-19](#) when Tag DA_h in tag 7F60_h in Table 11-6 is set to 01_h.
 3
 4 Friend slot identifier (Tag 4E_h) shall be included in [Table 11-20](#) when tag DA_h in Tag 7F60_h in Table 11-6 is set to 01_h or 03_h.
 5
 6 Friend immobilizer token (Tags 4A_h and 97_h) shall be included in [Table 11-20](#) when Tag DA_h in Tag 7F60_h in Table 11-6 is set to 01_h.
 7

8 *Listing 11-7: Key Tracking Receipt*

```

1 begin
2 device creates friend device to Device OEM Server link
3 device prepares message as per Table 11-19
4 device sends message over the Device OEM Server link to Vehicle OEM Server link as per Section 17 (API – trackKey)
5 Vehicle OEM Server verifies owner signature contained in attestation package
6 Vehicle OEM Server optionally tries to push the attestation package via vehicle telematic link
7 Vehicle OEM Server optionally computes key tracking receipt as
8   per vehicle OEM proprietary specification
9 device includes tag 45h and, if slot identifier is retrieved online 4Eh from Table 11-20 received from server into Table 11-18
10  response may be empty if vehicle OEM does not require key tracking receipt
11 device updates slot identifier as per Section 15.3.2.21 SETUP ENDPOINT command (if slot identifier is retrieved online)
12 device writes confidential mailbox using encrypted confidential mailbox data as per Section 15.3.2.20 SET CONFIDENTIAL
13 DATA command (if immobilizer token is retrieved online)
14 end
  
```

9 11.10.1 *Online immobilizer token and slot identifier retrieval*

- 10 As an optional feature, the immobilizer token and/or the slot identifier may be retrieved by the friend online, together with the Online Attestation package.
 11
 12 In this case, the owner shall not send a slot identifier in the Key Creation Request and shall set the tag DA_h (see Table 11-6) to 01_h to indicate the use of online immobilizer token, or to 03_h to indicate the use of online slot identifier without immobilizer token. After the key signing request, the owner will only use the key identifier to identify the friend key. The Key Tracking Response includes the immobilizer token and/or the slot identifier which has been assigned to this friend
 13
 14
 15
 16

1 key by the Vehicle OEM server. The friend shall store the immobilizer token in the confidential
2 mailbox using the SET CONFIDENTIAL DATA command and/or update the slot identifier
3 using the SETUP ENDPOINT command.

4 11.11 Vehicle Attestation

5 As an optional feature, the vehicle may be able to transmit an attestation to the owner device
6 when a friend enters a correct sharing password in the vehicle. Upon verification of this
7 attestation, the owner device may waive the sharing password requirement for subsequent
8 sharing with the same friend (identified by the same issuing Instance CA) if Device OEM and
9 Vehicle OEM policies allow it. Vehicle OEM should consider the consistency for sharing
10 password required in Digital Key creation and key tracking response.

11 The vehicle transmits the attestation to the Vehicle OEM Server via the telematic link. The
12 attestation is then transmitted to the owner device via the implemented communication channel.

13 *Table 11-21: Vehicle Attestation Data Fields*

Tag	Length (bytes)	Description	Field is
41 _h	1	version = 01 _h , version of the Vehicle Attestation Data Fields	mandatory
92 _h	8	random	mandatory
5A _h	65	friend public key	mandatory
44 _h	8	sharing_password_seed	mandatory
93 _h	4	usage = E670F31B _h	mandatory

14 *Table 11-22: Vehicle Attestation*

Tag	Length (bytes)	Description	Field is
7F34 _h	variable	Vehicle Attestation	
		key_attestation as described in Table 11-21	conditional
9E _h	variable	signature over fields in Table 11-21 using vehicle private key as per Table 11-19	optional

15 11.12 Algorithms

16 *Listing 11-8: setPrivateMailboxBit*

```

1 input: endpoint, bit_index, offset
2 output: n/a
3 begin
4   execute endpoint.getPrivateData as per Section 15.4.1.17
5   input: offset, length = 1
6   output: one_byte
7
8   mask = 1
9   mask = mask << bit_index
10  one_byte = one_byte | mask
11
12  execute endpoint.setPrivateData as per Section 15.4.1.18
13  input: offset, data = one_byte

```

14 | **end**

1 Listing 11-9: clearPrivateMailboxBit

```
1 input: endpoint, bit_index, offset
2 output: n/a
3 begin
4   execute endpoint.getPrivateData as per Section 15.4.1.17
5     input: offset, length = 1
6     output: one_byte
7
8   mask = 1
9   mask = mask << bit_index
10  mask = ~mask
11  one_byte = one_byte & mask
12
13 execute endpoint.setPrivateData as per Section 15.4.1.18
14   input: offset, data = one_byte
15 end
```

2 Listing 11-10: clearPrivateMailboxBytes

```
1 input: endpoint, offset, length
2 output: n/a
3 begin
4   generate empty_buffer using length
5   execute endpoint.setPrivateData as per Section 15.4.1.18
6     input: offset, data = empty_buffer
7 end
```

3 Listing 11-11: Generate Attestation Signature

```
1 input: data_fields, private_key
2 output: signature
3 begin
4   set curve_parameters ← "ECC NIST P-256" as per \[8\]
5   signature ← ECDSA(curve_parameters, private_key, data_fields) using SHA-256 as per \[8\]
6   return signature (64 bytes)
7 end
```

4 Listing 11-12: Generate Sharing Password

```
1 input: long_term_shared_secret, sharing_password_length, sharing_password_seed
2 output: sharing_password
3 begin
4   counter 0 (1 byte)
5   filtered_hash []
6   while length of filtered_hash < sharing_password_length
7     buffer ← counter || long_term_shared_secret || sharing_password_seed
8     hash ← SHA-256(buffer)
9     character_set_size ← 10 (digits from 0 to 9)
10    character_set_filter ← (256 - 256 mod character_set_size) - 1
11    for each byte in hash
12      if byte value <= character_set_filter
13        append byte to filtered_hash
14      if counter < 255
15        counter ← counter + 1
16      else
17        abort procedure
18    for i from 0 to (sharing_password_length - 1)
19      digit ← filtered_hash[i] mod character_set_size
```

```
20     sharing_password[i] ← toChar(digit)
21 return sharing_password (i+1 chars)
22 end
```

11.13 Versioning

- 2 Key sharing data is exchanged via framework APIs and is mostly consumed by the applet. The
3 version information shall combine framework and applet version into OD-FD-KS.
- 4 Version agreement does not add another roundtrip to the sharing flow. Instead, the version
5 information is sent by the owner device as part of the key creation request. This implies that the
6 provided key creation data can be consumed by the friend device independently of the supported
7 friend versions. To achieve this, the key creation data must contain all data required by all
8 possibly supported friend device versions.
- 9 If, for example, the format of the endpoint identifier changes in a future version, the new
10 endpoint identifier must be identified by a new tag and the owner device must send old and new
11 identifiers. The friend device can then pick the required format.
- 12 Mandatory tags shall not be removed from the key creation request, as the friend device might
13 only support a version where this tag is required to proceed.
- 14 The friend device determines the highest commonly supported version from the owner version
15 list and its own version list. It then sends the key signing request in a format supported by the
16 owner device.
- 17 The friend device uses the list of vehicle-supported owner pairing versions (V-OD-FW) to
18 determine the format of the key tracking request (Table 11-19). Note that content and format of
19 some fields of the key tracking request are determined by other domain versions.
- 20 If no common version can be determined, then both sides shall fall back to the sharing data
21 formats defined before versioning was introduced

12 DELEGATED KEY SHARING

- 2 Delegated key sharing feature is out of scope of this specification.

13 KEY TERMINATION AND DELETION [WCC1/WCC2]

2 Key termination may be triggered in the vehicle, in the Vehicle OEM Server (e.g., account), on
3 the device, and in the Device OEM Server (e.g., remote device wipe, if supported).
4 A termination attestation should be sent from the device to the KTS whenever the device
5 terminates the Digital Key. Along with the termination attestation, the device should send the
6 Vehicle OEM proprietary data subsection of the private mailbox. The termination attestation and
7 the Vehicle OEM proprietary data may not be sent in all cases due to technical constraints.
8 If the termination attestation is provided in the manageKey call from the Device OEM server to
9 the Vehicle OEM Server, then the vehicle OEM server should consider the key as “terminated in
10 the device” before attempting to terminate the key in the vehicle.
11 If the termination attestation is not provided in the manageKey call from the Device OEM server
12 to the Vehicle OEM Server, then the vehicle OEM server shall continue to terminate the key in
13 the vehicle while respecting the vehicle key termination conditions.
14 The Device OEM shall not terminate an active Digital Key unless one of the following
15 conditions are met:
16 • The terminate action is initiated by an authenticated user and user intent on the device is
17 confirmed, or
18 • After the vehicle OEM has signaled to the device OEM that the Digital Key has been
19 deleted in the vehicle
20 In the case of remote device wipe (REV_170/REV_420), the action shall only be initiated by an
21 authenticated user and Device OEM shall inform user that any keys on the device will not be
22 useable.
23 To address user safety, a fade-out period may be established by the Vehicle OEM server. The
24 fade out period is defined as the time between the IN_TERMINATION message and the actual
25 termination of the key on the vehicle. The Vehicle OEM server notifies the owner or friend
26 Device OEM server about the start of this period by sending an IN_TERMINATION
27 notification. Further requirements associated with the fade-out period (e.g. use, purpose and
28 duration of the period) are Vehicle OEM specific. A sample implementation of
29 IN_TERMINATION notification is shown in 17.7.7.6. After receiving the IN_TERMINATION
30 message, Device OEM server should notify the user. Further behaviors after reception of the
31 IN_TERMINATION message by the Device OEM server are specific to the Device OEM.
32 If a fade-out period is required, the Vehicle OEM should notify the device about the start of this
33 period.
34 If the fade-out period has ended and/or the conditions for a successful termination of the key in
35 the vehicle are met, the key is tracked as terminated in the KTS.
36 If online, owner and friend devices shall be notified about the successful termination. The device
37 should delete the associated Digital Keys prior to deletion of the instance CA triggered by
38 DELETE CA command.
39 The detailed termination and deletion flows in the vehicle and device are described in Sections
40 [13.2](#), [13.3](#), [13.4](#), [13.5](#), and [13.6](#).

13.1 Key Termination Scenarios

The termination and suspension use cases defined by CCC are listed in [Table 13-1](#), [Table 13-2](#), [Table 13-3](#), and [Table 13-4](#).

Table 13-1: Vehicle-triggered Key Termination

Use Case ID	Trigger	Owner Key Status	Friend Key Status
R-VT-1	Owner terminates owner key in vehicle	Terminated	-
R-VT-2	Owner terminates friend key in vehicle	-	Terminated
R-VT-3	Owner changes owner device via vehicle	Terminated	-
R-VT-4	Owner/user unpairs vehicle from owner account or resets Digital Key function	Terminated	Terminated

Table 13-2: Device-triggered Key Termination

Use Case ID	Trigger	Owner Key Status	Friend Key Status
R-DT-1	Owner terminates owner key in Vehicle OEM app	Terminated	-
R-DT-2	Owner terminates friend key in Vehicle OEM app	-	Terminated
R-DT-3	Owner terminates owner key in native app	Terminated	-
R-DT-4	Owner terminates friend key in native app	-	Terminated
R-DT-5	Friend terminates friend key in Vehicle OEM app	-	Terminated
R-DT-6	Friend terminates friend key in native app	-	Terminated
R-DT-7	Owner unpairs vehicle from owner account in Vehicle OEM app	Terminated	Terminated

Table 13-3: Vehicle OEM-triggered Key Termination

Use Case ID	Trigger	Owner Key Status	Friend Key Status
R-VOT-1	Owner terminates owner key in Vehicle OEM customer web portal, hotline, or support	Terminated	-
R-VOT-2	Owner terminates friend key in Vehicle OEM customer web portal, hotline, or support	n/a	Terminated
R-VOT-3	Friend terminates friend key in Vehicle OEM customer web portal, hotline, or technical support	n/a	Terminated
R-VOT-4	Owner Digital Key service license expires	Terminated	Terminated
R-VOT-5	Terminate friend key after expiry date (automatic)	n/a	Terminated
R-VOT-6	Owner/legal authorities report a stolen/destroyed vehicle	Terminated	Terminated
R-VOT-7	Garage service process	Terminated	Terminated
R-VOT-8	Security breach on vehicle is detected.	Terminated	Terminated
R-VOT-9	Owner triggers deletion of personal data(owner account)in Vehicle OEM Server	Terminated	Terminated

Use Case ID	Trigger	Owner Key Status	Friend Key Status
R-VOT-10	Friend triggers deletion of personal data (friend account) in Vehicle OEM Server	n/a	Terminated
R-VOT-11	Owner unpairs vehicle from owner account in Vehicle OEM Server	Terminated	Terminated

1

Table 13-4: Device OEM-triggered Key Termination

Use Case ID	Trigger	Owner Key Status	Friend Key Status
R-DOT-1	Owner/User wipes device (factory reset)	Terminated	-
R-DOT-2	Friend wipes friend device (factory reset)	n/a	Terminated
R-DOT-3a	Owner reports lost/stolen/rediscovered owner device: suspend/resume keys	Suspended/ Resumed	n/a
R-DOT-3b	Owner reports lost/stolen owner device: remote wipe keys	Terminated	n/a
R-DOT-4a	Friend reports lost/stolen/rediscovered friend device: suspend/resume keys	n/a	Suspended/ Resumed
R-DOT-4b	Friend reports lost/stolen friend device: remote wipe keys	n/a	Terminated
R-DOT-5a	Security breach on owner device is detected	Terminated	Terminated
R-DOT-5b	Security breach on friend device is detected	n/a	Terminated

2 The use cases in [Table 13-1](#) through [Table 13-4](#) are grouped into more detailed flow diagrams.
 3 [Table 13-6](#) provides the mapping. In all the flow diagrams described in Sections [13.2](#) through
 4 [13.6](#), the exchanges between vehicle and Vehicle OEM Server, device and Device OEM Server,
 5 and Vehicle OEM Server and KTS are out of scope of this specification. These exchanges are
 6 shown for illustration purposes only. The exchanges between Vehicle OEM Server and Device
 7 OEM Server are described by the relevant server APIs (see Section [17](#)). The following APIs are
 8 used in key termination procedures:

- **manageKey()** and **manageKeyResponse**: See Section [17.7.2](#)
- **eventNotification()** and **eventNotificationResponse**: See Section [17.7.3](#)

11 Note: Only relevant parameter(s) in the API that are specific to the particular step in the flow
 12 diagram are shown.

13 All keys are identified by their Digital Key identifier (key id) in the Vehicle OEM Server and
 14 Device OEM Server. Digital Key identifiers are not shown explicitly in the flow diagrams unless
 15 required for comprehension. Along with the terminationAttestation parameter, the
 16 vehicleOEMProprietaryData is also transferred if not explicitly shown in the flow diagrams.
 17 Interpretations of the keyID parameter inside of the manageKey() call for action=TERMINATE
 18 (as defined in section 17.7.2.2), are described in Table 13-5 below

19

1

2

Table 13-5 keyID usage for manageKey() with action=TERMINATE

Request by	ManageKey() Parameters	KeyID Usage
Owner / Friend Device	No RTR included, TA included	Key deleted on device, key to be deleted in vehicle
Device OEM Server	No RTR included, no TA included	key requested to be deleted in vehicle
Owner Device	Device RTR included	keyID = Owner keyID, multiple keys can be deleted in RTR
Vehicle OEM Server	Server RTR included	keyID = one of the keyIDs contained in the RTR

3 For full parameters of each API call, please refer to the corresponding sections.

4 Note: Friend key ID is used in all the figures in this section and that the term is synonymous
5 with Digital Key identifier of a friend.

6 Data elements requiring privacy encryption (See Section 14) are identified by “*” in the flow
7 diagrams. The exact data elements to be encrypted are defined in Section 17.

8 Messages requiring authentication (See Section 14) are identified by “+” in the flow diagrams.

9 The exact messages and API calls to be signed are defined in Section 17.

10

Table 13-6: Detailed Key Termination Use Cases

Flow ID	Description	Use Case ID
Friend Key Remote Termination		
REV_100	Friend key termination in vehicle	R-VT-2
REV_110	Friend key termination in owner Vehicle OEM account	R-VOT-2
REV_120	Friend key termination in friend Vehicle OEM account	R-VOT-3
REV_130	Friend key termination on owner device natively	R-DT-4
REV_140	Friend key termination on owner device in Vehicle OEM app	R-DT-2
REV_150	Friend key termination based on expiry date of the key	R-VOT-5
REV_160	Friend key termination by Device OEM (device security issue)	R-DOT-5b
REV_170	Friend key termination due to remote wipe of device	R-DOT-4b
REV_180	Friend key termination due to deletion of personal data in friend Vehicle OEM account	R-VOT-10
Friend Key Local Deletion		
REV_200	Friend key termination on friend device natively	R-DT-6
REV_210	Friend key termination on friend device in Vehicle OEM app	R-DT-5
REV_220	Friend key termination due to local wipe of device	R-DOT-2
Friend Key Remote Suspension		
REV_300	Friend key temporary suspension in device lost mode in Device OEM account	R-DOT-4a
REV_310	Friend key resume from device lost mode in Device OEM account	R-DOT-4a
Owner Key Remote Termination		

Flow ID	Description	Use Case ID
REV_400	Owner key deletion in vehicle UI (change device)	R-VT-1, R-VT-3
REV_410	Owner key termination by Device OEM (security issue)	R-DOT-5a
REV_420	Owner key termination due to remote wipe of device	R-DOT-3b
Owner Key Local Deletion		
REV_500	Owner key termination on owner device natively (deletion of pass for Digital Key)	R-DT-3
REV_510	Owner key termination on owner device in Vehicle OEM app	R-DT-1
REV_520	Owner key termination due to local wipe of device	R-DOT-1
Owner Key Remote Suspension		
REV_600	Owner key temporary suspension in device lost mode in Device OEM account	R-DOT-3a
REV_610	Owner key resume from device lost mode in Device OEM account	R-DOT-3a
Vehicle Unpairing		
REV_700	Unpairing in vehicle UI (sale of vehicle)	R-VT-4
REV_710	Unpairing on owner device in Vehicle OEM app	R-DT-7
REV_720	Unpairing in owner Vehicle OEM account (sale of vehicle)	R-VOT-11
REV_730	Unpairing based on cancellation or expiry date of the Digital Key service	R-VOT-4
REV_740	Unpairing by Vehicle OEM (vehicle stolen)	R-VOT-6
REV_750	Unpairing by Vehicle OEM (security breach in vehicle)	R-VOT-8
REV_760	Unpairing by Vehicle OEM (garage service process)	R-VOT-7
REV_770	Unpairing due to deletion of personal data in owner Vehicle OEM account	R-VOT-9
Vehicle Unbinding (leads to unpairing)		
REV_800	Unbinding of vehicle from owner account in vehicle UI (unpairing, owner sells vehicle)	R-VT-4
REV_810	Unbinding of vehicle from owner account in owner Vehicle OEM account (unpairing, owner sells vehicle)	R-VOT-11
REV_820	Unbinding of vehicle from owner account in Vehicle OEM app (unpairing, owner sells vehicle)	R-DT-7

13.2 Friend Key Remote Termination

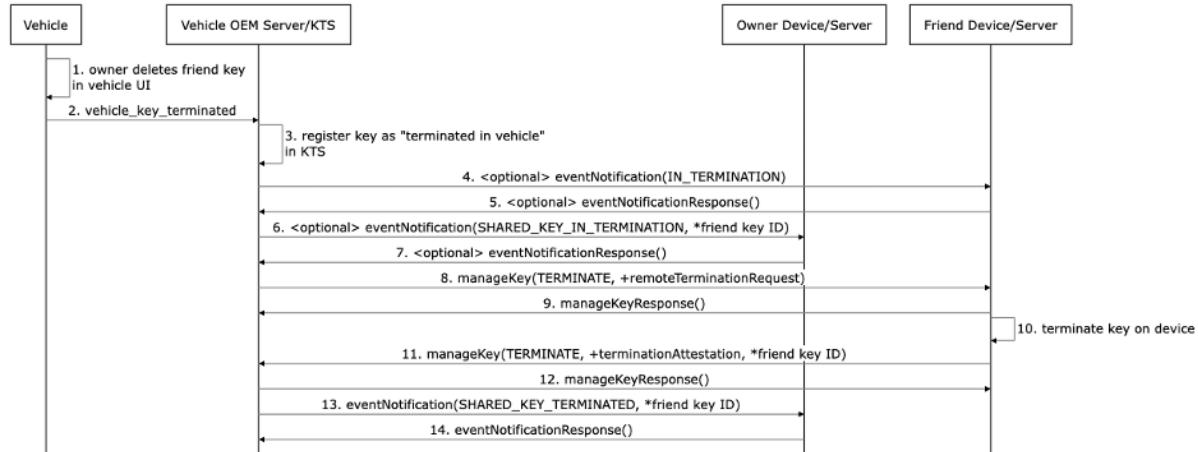
- 2 All scenarios in which the friend key termination process is started from a device other than the
 3 friend's own device are considered as remote termination cases. The common behavior is that
 4 the termination request is issued to the Vehicle OEM Server, which then starts the termination
 5 process on the vehicle side.

1 For remote termination, the key is considered as terminated when it is deleted in the vehicle.

2 13.2.1 REV_100: Friend key termination in vehicle

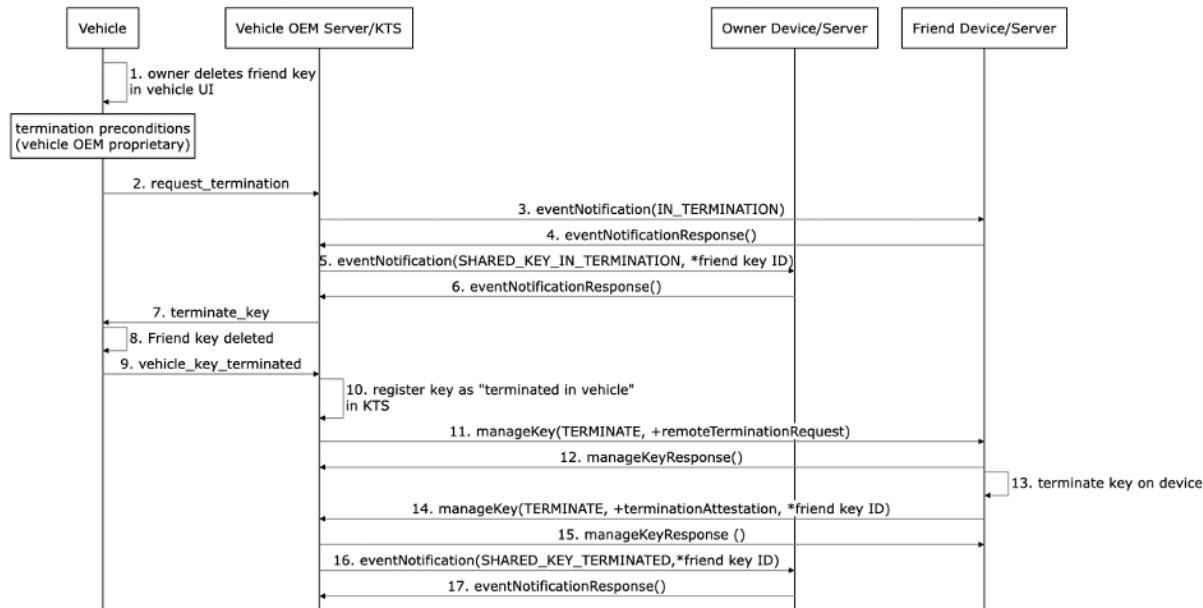
3 If the owner deletes a friend key in the vehicle, no fade-out is required. The vehicle notifies the
4 devices about the successful deletion via the Vehicle OEM Server (REV_100).

5 *Figure 13-1: REV_100: Friend Key Termination in Vehicle*



6
7 Alternatively, it may be required for the vehicle to request online deletion. In this case, the flow
8 is similar to remote termination from the Vehicle OEM Server. Devices are informed about the
9 deletion request and the successful deletion (REV_100a)

10 *Figure 13-2: REV_100a: Friend Key Termination in Vehicle (Vehicle Required to be Online)*



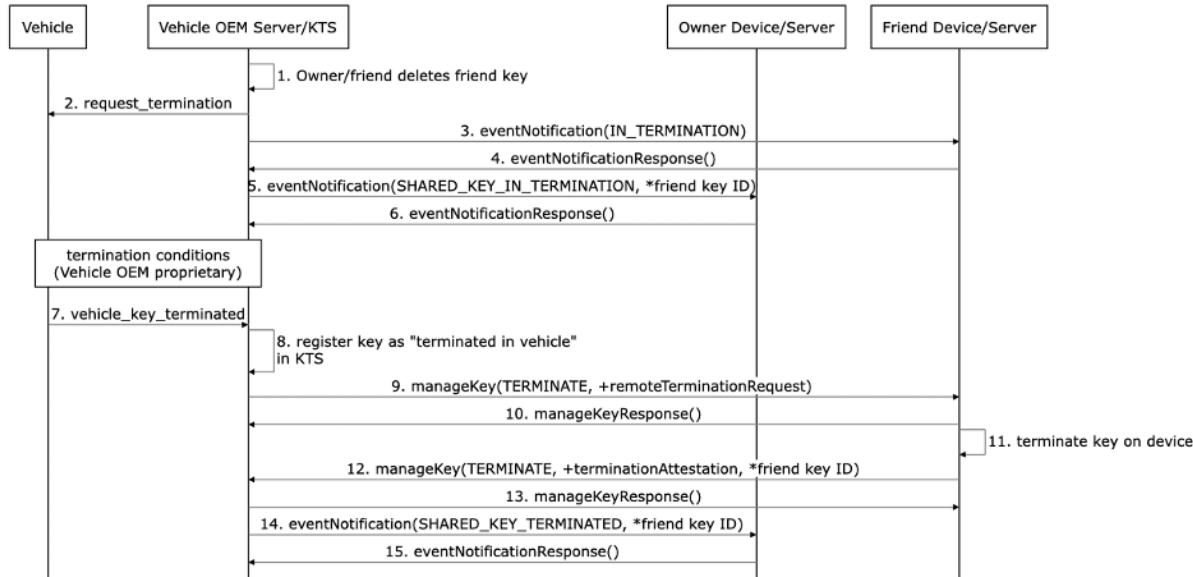
11
12 The eventNotification (IN_TERMINATION) should be sent in all cases so that the device side
13 does not see any difference between both options.

1 13.2.2 REV_110: Friend key termination in owner Vehicle OEM account

2 13.2.3 REV_120: Friend key termination in friend Vehicle OEM account

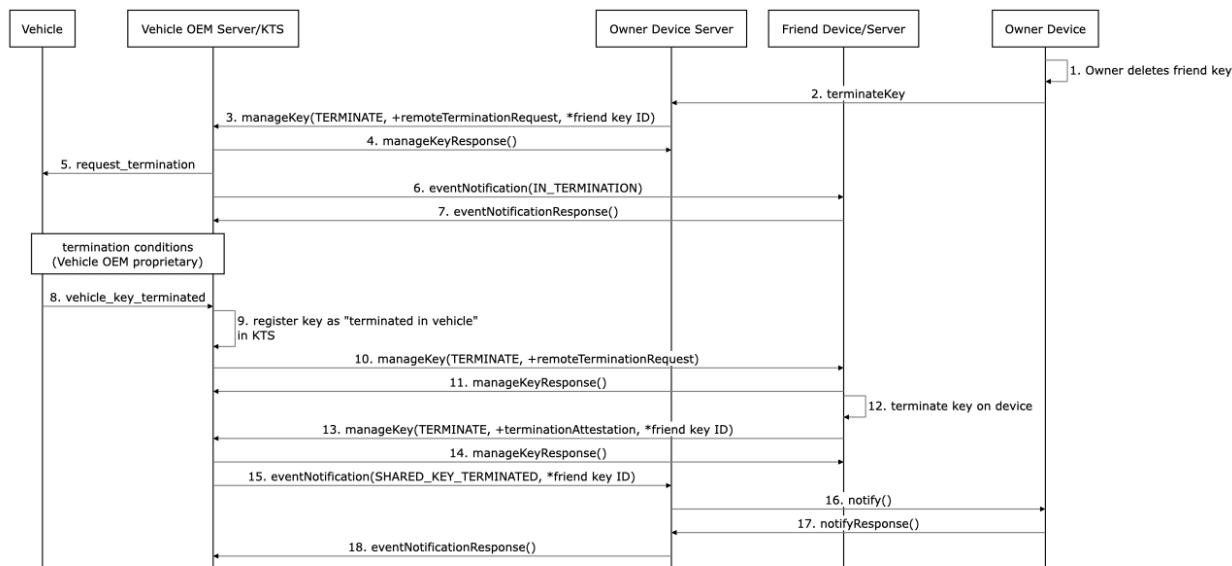
3 The same flow shall be executed when either the owner deletes the friend key from the owner's
4 Vehicle OEM account or the friends delete their own key from their Vehicle OEM account. This
5 is shown in [Figure 13-3](#).

6 *Figure 13-3: REV_110/120: Friend Key Termination in Owner/Friend Vehicle OEM Account*



7

- 1 **13.2.4 REV_130: Friend key termination on owner device natively**
- 2 **13.2.5 REV_140: Friend key termination on owner device in Vehicle OEM app**
- 3 *Figure 13-4: REV_130/140: Friend Key Termination on Owner Device Natively/in Vehicle OEM App*



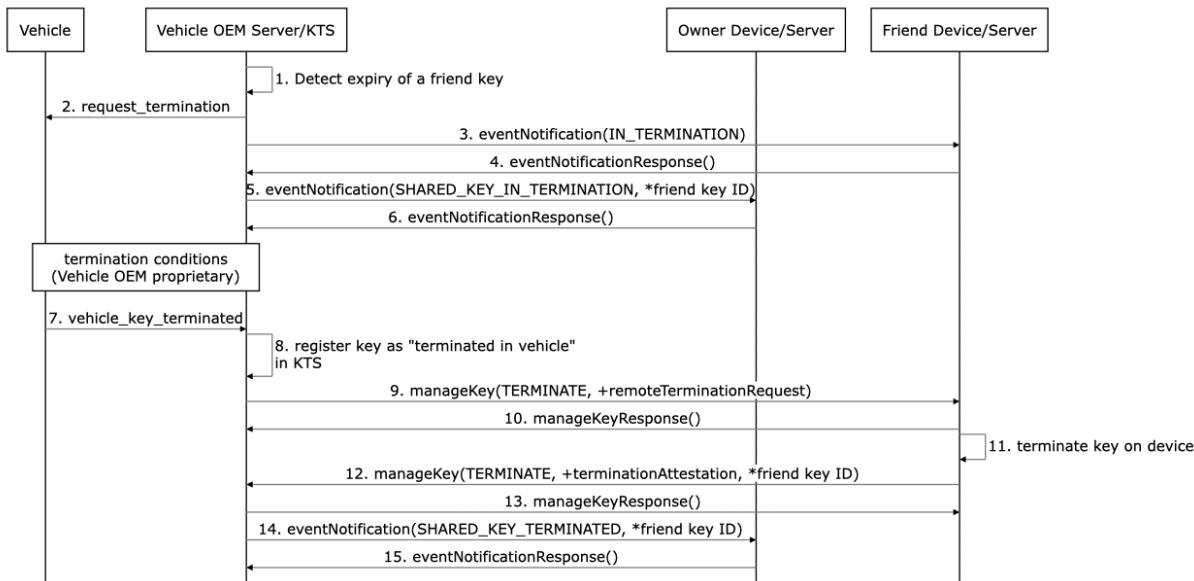
- 4
- 5 In some cases, the owner may terminate a friend key using one of the following methods:
 1. Natively through the device operating system (REV_130, see figure 13-4)
 2. Using vehicle OEM apps installed on the device which leverage one or more APIs offered by the device operating system (REV_140, see figure 13-4)
 3. Using vehicle OEM apps installed on the device which contact the Vehicle OEM server directly. (REV_110)

11 **13.2.6 REV_150: Friend key termination based on expiry date of the key**

- 12 If the vehicle or the Vehicle OEM Server detects that a friend key has expired, the key is deleted from the vehicle in the same way as if it were deleted in the vehicle or the Vehicle OEM Server by the owner.
- 15 The flow is similar to REV_120: Friend key termination in friend Vehicle OEM account (see Section [13.2.3](#)).

1

Figure 13-5: REV_150: Friend Key Termination Based on Expiry Date of the Key

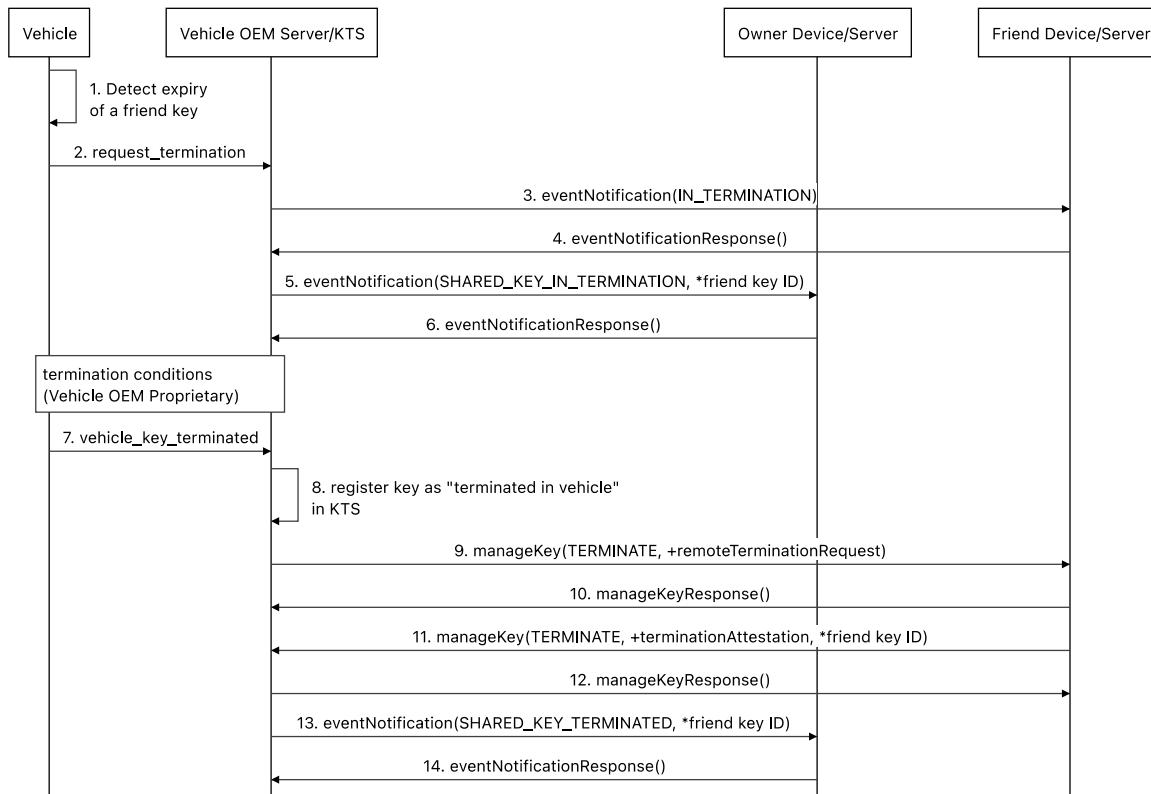


2

3 The flow depicting Vehicle detection of an expired friend key is shown in Figure 13-6 and is
4 similar to REV_100: Friend Key Termination in Vehicle (Figure 13-1)

5

Figure 13-6 Vehicle Detects Expiry of Friend Key



6

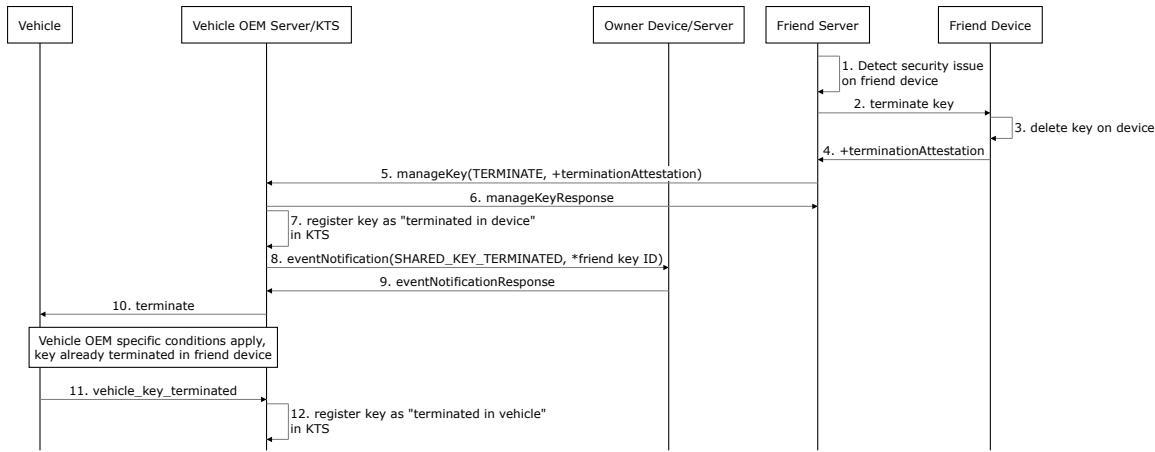
1 13.2.7 REV_160: *Friend key termination by Device OEM (device security issue)*

2 When a security issue on a device or group of devices is detected, the Device OEM removes the
3 Digital Keys based on the nature of the security issue. The details are out of scope of this
4 specification.

5 In those cases, the keys may be removed immediately from the device, depending on the security
6 issues.

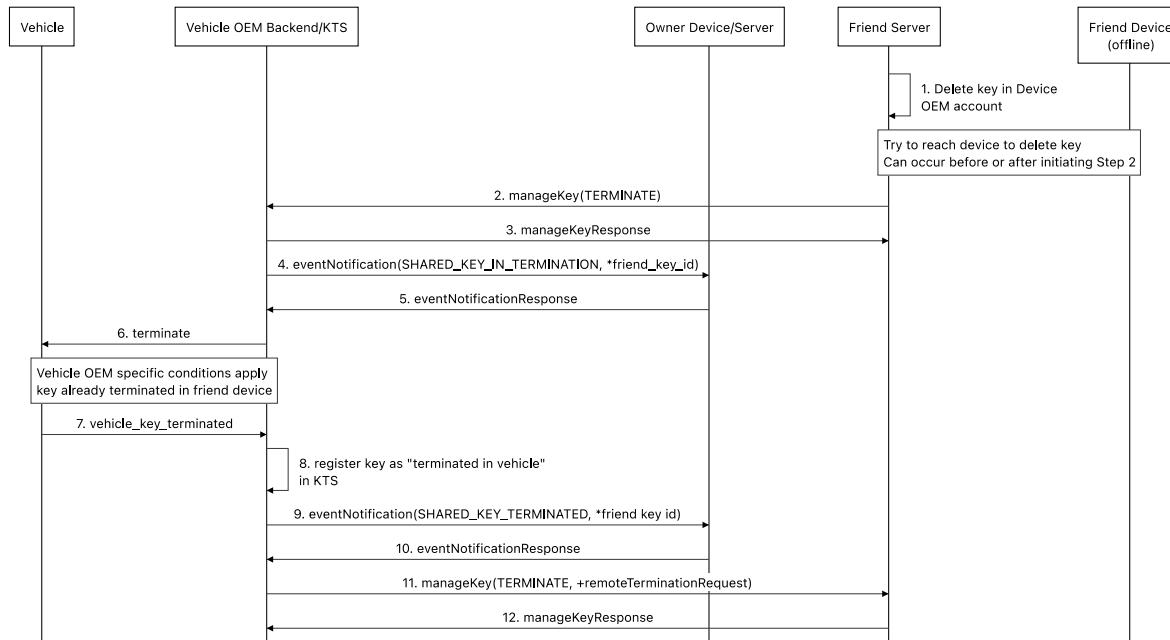
7 The termination attestation and the Vehicle OEM proprietary data from the device are provided
8 to the KTS.

9 *Figure 13-7: REV_160: Friend Key Termination by Device OEM (Device Security Issue)*



11 When the device cannot be reached, the Vehicle OEM Server shall still be notified immediately,
12 as shown in [Figure 13-8](#).

13 *Figure 13-8: REV_160a: Friend Key termination by Device OEM and Friend Device is Offline*



1

2 The termination attestation and the Vehicle OEM proprietary data from the device may be
3 provided to the KTS if the device is online.

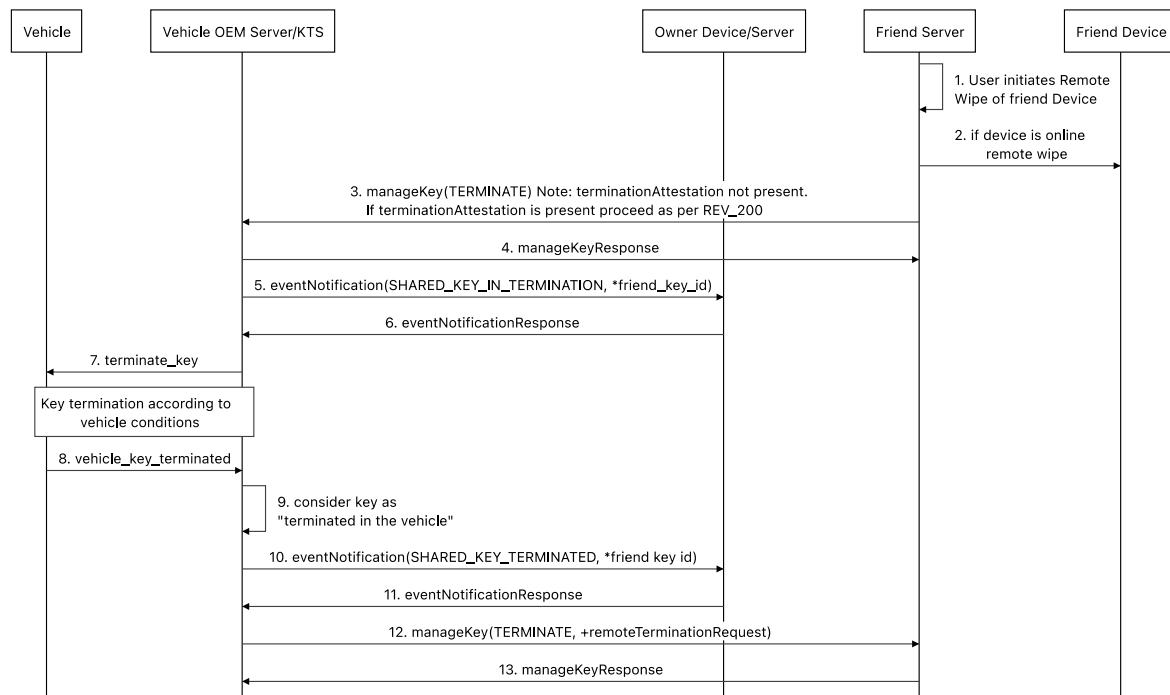
4 **13.2.8 REV_170: Friend key termination due to remote wipe of device**

5 When a friend decides to wipe his/her device remotely, the Device OEM deletes the Digital Keys
6 on the involved devices. The details are out of scope of this specification.

7 In those cases, the keys are deleted immediately from the friend's device. Vehicle and Vehicle
8 OEM termination conditions and fade-out do not apply.

9 *Note:* In the case of a remote wipe of a device, the device may include terminationAttestation
10 and vehicleOEMProprietaryData in manageKey() in Step 3, as shown in [Figure 13-9](#).

11 *Figure 13-9: REV_170: Friend Key Termination Due to Remote Wipe of Device*



12

13 In some cases, the friend may terminate the friend key using one of the following methods:
14 1. Natively through the device operating system (REV_200, see Figure 13-10)
15 2. Using vehicle OEM apps installed on the device which leverage one or more APIs
16 offered by the device operating system (REV_210, see Figure 13-10)
17 3. Using vehicle OEM apps installed on the device which contact the vehicle OEM server
18 directly (REV_120)

1 13.2.9 *REV_180: Friend key termination due to deletion of personal data in friend*
2 *Vehicle OEM account*

3 When the friend has a Vehicle OEM account, the shared key is linked with the account. A
4 request to delete the personal data from the account should lead to removal of the key from the
5 vehicle, similar to a key deletion as described in “REV_120: Friend key termination in friend
6 Vehicle OEM account” (see Section [13.2.3](#)).

7 **13.3 Friend Key Local Termination**

8 All scenarios in which the friend key termination process is started on the friend’s own device
9 are considered as local termination cases.

10 The common behavior is that the termination request should be executed immediately on the
11 friend device. Depending on the connectivity situation, the termination attestation and the
12 Vehicle OEM proprietary data may be sent immediately to the KTS, delayed, or not sent at all.

13 In cases where the Vehicle OEM Server is not aware of the termination, termination attempt, or
14 loss of the device, the owner or friend (if applicable) should remove the key in the vehicle.

15 Fade-out or termination preconditions do not apply when termination attestation and the Vehicle
16 OEM proprietary data will be transmitted to the Vehicle OEM Server.

17 The owner device should be notified about successful registration of the termination. The friend
18 and owner device UI should mark the corresponding key representation as “terminated.”

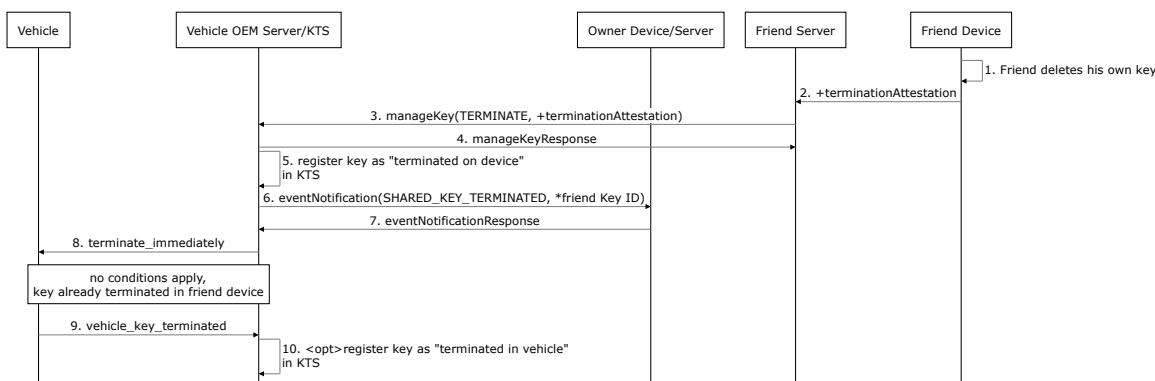
19 13.3.1 *REV_200: Friend key termination on friend device natively*

20 13.3.2 *REV_210: Friend key termination on friend device in Vehicle OEM app*

21 If the key is deleted on the friend device, the termination attestation and the Vehicle OEM
22 proprietary data shall be sent to the friend server as soon as the device is online.

23 The server flow is exactly the same for native and Vehicle OEM app-initiated deletion.

24 *Figure 13-10: REV_200/210: Friend Key Termination on Friend Device Natively/in Vehicle OEM App*

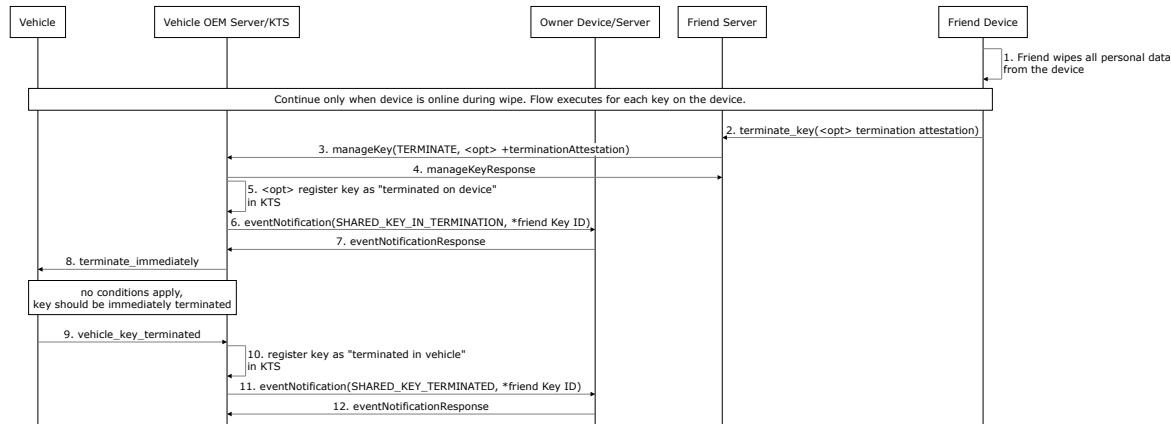


25
26 The Vehicle OEM indicates to the vehicle that the key can be terminated immediately when the
27 device key is registered in the KTS as “terminated on device.”

1 13.3.3 *REV_220: Friend key termination due to local wipe of device*

2 If the device is wiped locally, the termination attestation and the Vehicle OEM proprietary data
3 is sent to the friend server on a best-effort basis.

4 *Figure 13-11: REV_220: Friend Key Termination Due to Local Wipe of Device*



5

6 The Vehicle OEM indicates to the vehicle that the key can be terminated immediately when the
7 device key is registered in the KTS as “terminated on device.”

8 For each friend key present on the device, the flow shown in Figure 13-11 shall be executed.

9 The owner key termination due to device wipe is described in Section [13.5.6](#).

10 **13.4 Friend Key Remote Suspend/Resume**

11 Remote suspend and resume occur when the device is set to lost or stolen mode in the Device
12 OEM Server and when the lost/stolen mode is ended.

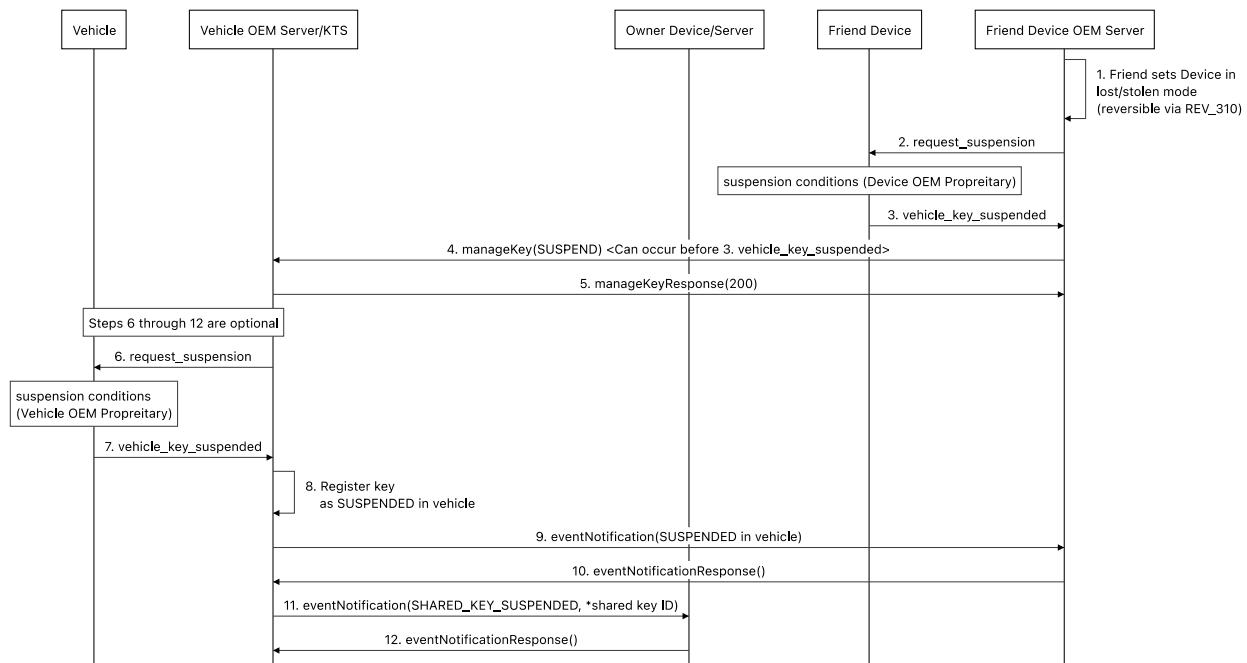
13 If the device is reachable, the keys are suspended immediately on the device. The suspended
14 status is reversible; keys may be resumed through the Device OEM Server or locally on the
15 device. No cryptographic attestation is available for a key suspended and resumed on the device.

16 **13.4.1 *REV_300: Friend key temporary suspension in Device OEM account***

17 Suspension of a Digital Key triggered by the Device OEM account is used when the device mode
18 is set to lost or stolen. The Device OEM Server attempts to suspend the key on the device and, in
19 parallel, the Vehicle OEM Server is notified to suspend the keys in the vehicle, as shown in
20 [Figure 13-12](#).

1

Figure 13-12: REV_300: Friend Key Suspension by Device OEM Account



2
3 The suspension on the device is not tracked in the KTS. The suspension of the key in the vehicle
4 may be tracked by the KTS.

5 manageKey SUSPEND command communicates the intent of the user to suspend the key to the
6 KTS, but not the suspension status in the device.

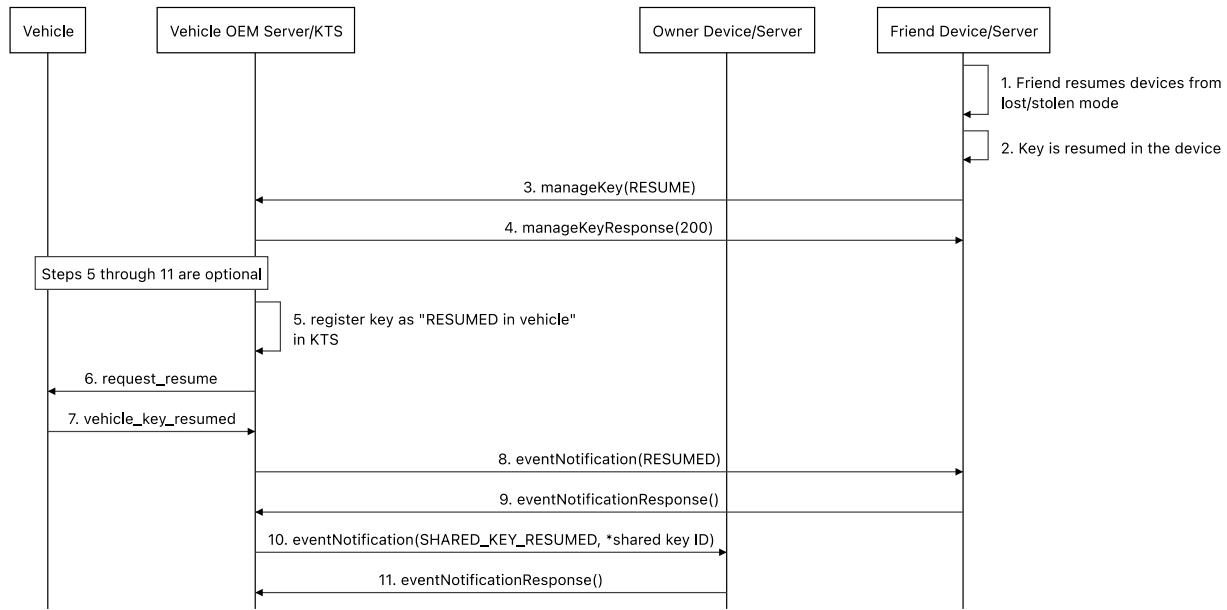
7 The Vehicle OEM notifies the owner and friend when the key is successfully suspended in the
8 vehicle.

9 Regardless of vehicle OEM server support of remote suspend/resume or not, the Vehicle OEM
10 server shall response with manageKeyReponse (statusCode 200) if it receives the
11 manageKeyRequest successfully.

12 13.4.2 REV_310: Friend key resume in Device OEM account

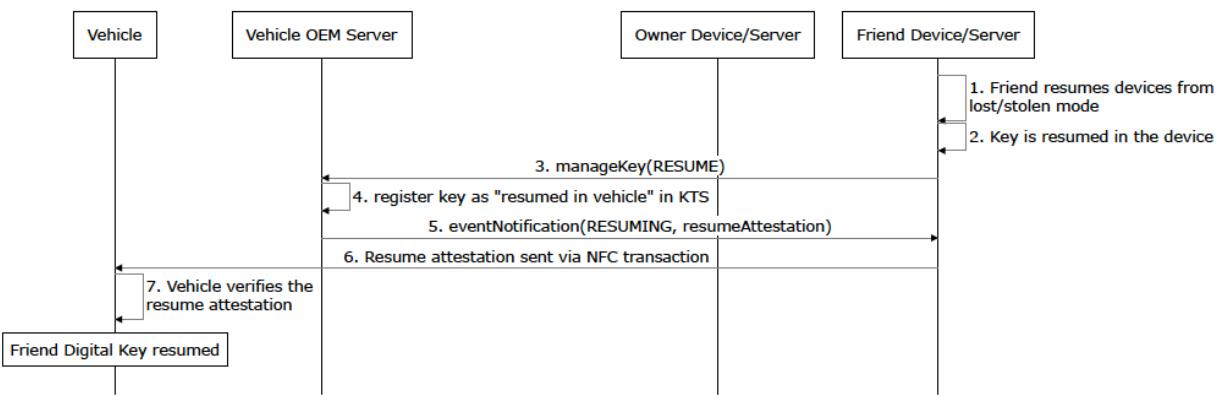
13 In the case where the Vehicle OEM Server supports remote suspend/resume, the suspended key
14 may be resumed in the Device OEM account or on the device itself if it has been recovered, as
15 shown in [Figure 13-13](#). The friend Device OEM Server shall only send manageKey(RESUME)
16 or transmit a resume attestation in the friend private mailbox if it has previously received
17 eventNotification(SUSPENDED).

1 Figure 13-13: REV_310: Friend Key Resume in Device OEM Account or on Device



- 2 3 The device shall be online to resume the key and notify the Vehicle OEM.
 4 If the key is previously tracked as “suspended in vehicle,” the key shall be tracked as “resumed
 5 in vehicle” before the vehicle resumes the key to avoid a state in which the key is resumed but
 6 still tracked as suspended.

7 Figure 13-14: REV_310: Friend Key Resume Using ResumeAttestation



- 8 9 When the vehicle is moved into an offline area between suspend and resume, the friend shall be
 10 able to provide a resume attestation via NFC transaction to the vehicle, as shown in [Figure
11 13-14](#).

12 13.4.3 Resume attestation

- 13 When a Digital Key has been suspended in a vehicle, it shall be possible to bring a Resume
 14 Attestation using the private mailbox of an owner’s or friend phone in a similar way as it is done
 15 for the first friend transaction.

- 1 The Resume Attestation may be used when the vehicle is offline and the owner Digital Key (see
2 Section [13.5.7](#)) or the friend Digital Key (see Section [13.4.1](#)) is in a suspended state.
3 The device shall add the tag 7F61_h and length fields to the resume-Attestation (see Table 13-7)
4 and store the TLV structure in the KeyAtt field of the private mailbox (see Table 4-2).The
5 Resume Attestation shall be written by the Digital Key framework in the private mailbox at
6 offset ATTESTATION_PACKAGE_OFFSET and the ATTESTATION_PACKAGE_BIT
7 signaling bit shall be set.
8 During a transaction, the vehicle detects the ATTESTATION_PACKAGE_BIT and consumes
9 the attestation. The vehicle then clears the ATTESTATION_PACKAGE_BIT and clears the
10 mailbox area at offset ATTESTATION_PACKAGE_OFFSET.
11 On reception of a valid Resume Attestation, the vehicle resumes the designated key or does
12 nothing if the Digital Key is already resumed.

13 **Implementation Notes:**

14 If the suspend/resume feature is supported by the Vehicle OEM, the Vehicle OEM must ensure
15 that the Resume Attestation cannot be replayed using a proprietary method.

16 *Table 13-7: Resume Attestation*

Tag	Length	Description	Field is	Domain Version
7F61 _h	variable	resumeAttestation (see Section 17.8.10)		V-OD-FW
vehicle OEM proprietary			mandatory	N/A

17 **13.5 Owner Key Remote Termination**

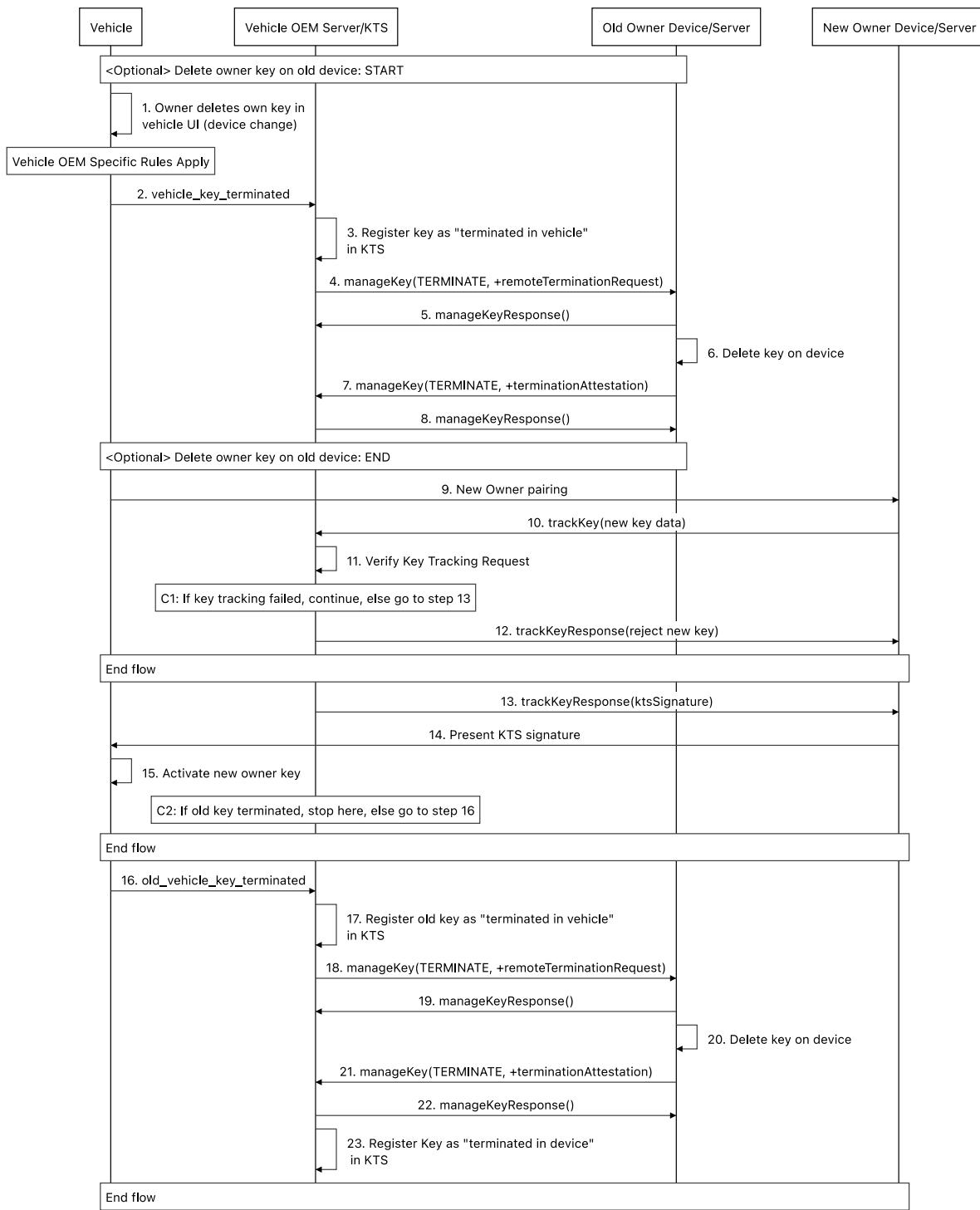
18 Owner key remote termination shall be handled very similar to friend key remote termination as
19 described in Section [13.2](#). The friend device is not informed about owner key termination.
20 Owner key termination does not affect the functionality of friend keys for the same vehicle.

21 **13.5.1 REV_400: Owner key deletion in vehicle UI (change device)**

22 If an owner device change is triggered in the vehicle, the vehicle deletes the owner key when the
23 necessary pre-conditions are fulfilled. Friend keys shall not be affected. Before the owner device
24 change, the old owner Digital Key may have been terminated.

1

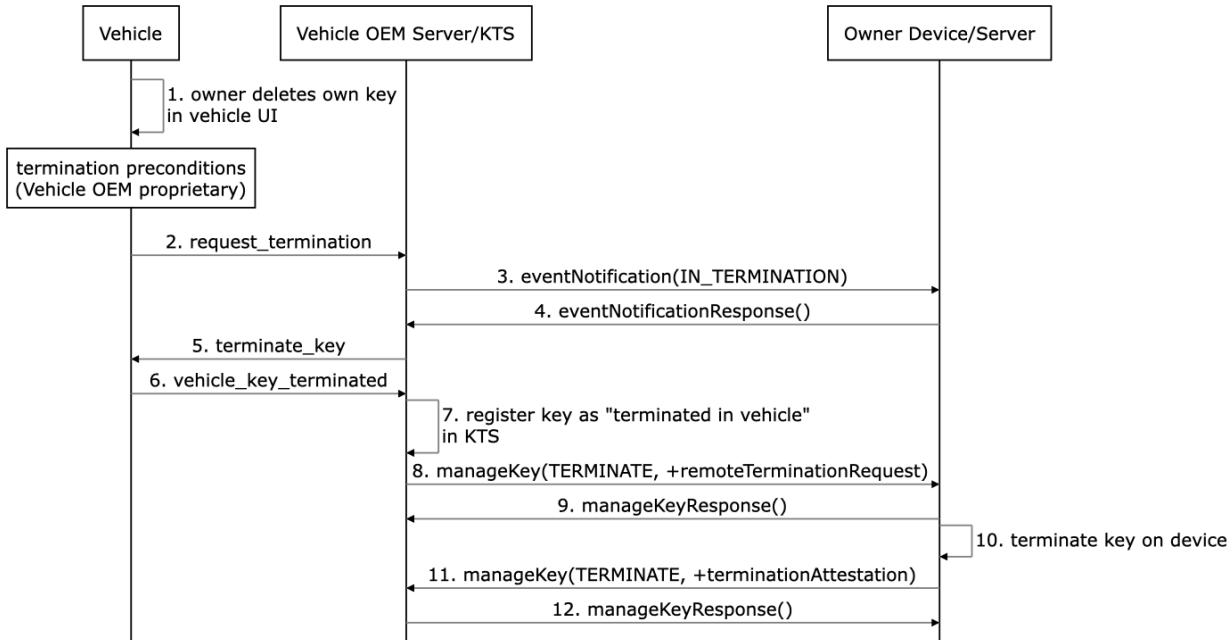
Figure 13-15: REV_400: Owner Key Deletion in Vehicle UI (Change of Device)



2

3 Alternatively, it may be required for the vehicle to request online deletion. In that case, devices
4 are informed about the termination request and the successful termination (See [Figure 13-16](#)).

1 Figure 13-16: REV_400a: Owner Key Deletion in Vehicle UI (Change of Device)



2

3 Note: REV_400 and REV_400a are similar to flow REV_100 and REV_100a, respectively, but
4 without involvement of the friend device.

5 13.5.1.1 *Transmission of the shared key information to the new owner device*

6 After the owner device change, information related to the shared keys associated with the owner
7 Digital Key is sent to the new owner device in Step 22 of [Figure 13-15](#) using trackKeyResponse
8 (see Section [17.7.1](#)).

9 13.5.2 *REV_410: Owner key termination by Device OEM (security issue)*

10 This flow corresponds to REV_160 with the owner device instead of the friend device. In
11 addition, all friend keys are terminated, and the friend devices are notified.

12 13.5.3 *REV_420: Owner key termination due to remote wipe of device*

13 This flow corresponds to REV_170 with the owner device instead of the friend device.

14 13.5.4 *REV_500: Owner key termination on owner device natively (delete pass for
15 Digital Key)*

16 This flow corresponds to REV_200 with the owner device instead of the friend device.

17 13.5.5 *REV_510: Owner key termination on owner device in Vehicle OEM app*

18 This flow corresponds to REV_210 with the owner device instead of the friend device.

19 13.5.6 *REV_520: Owner key termination due to local wipe of device*

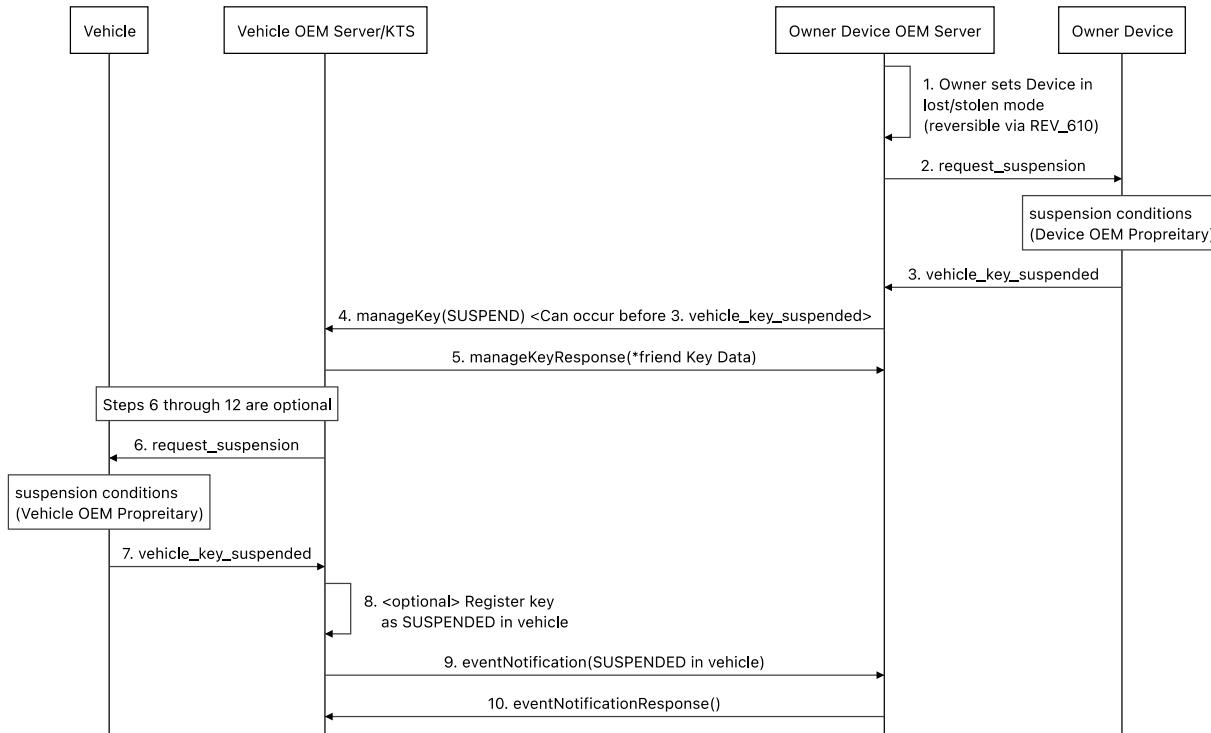
20 This flow corresponds to REV_220 with the owner device instead of the friend device.

1 13.5.7 *REV_600/610: Owner key temporary suspension in device lost mode in Device*
2 *OEM account*

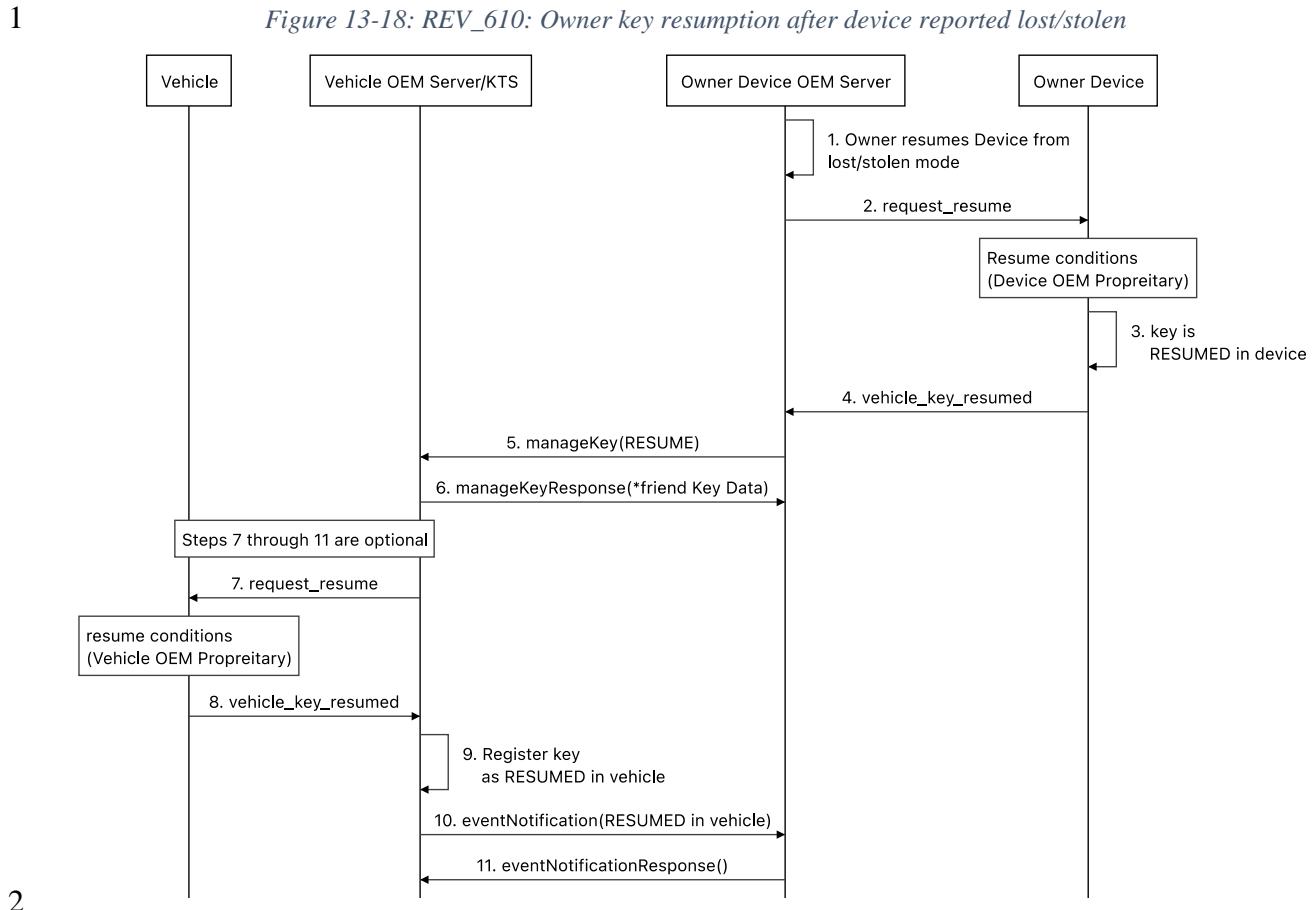
3 This flow corresponds to REV_300 and REV_310 with the owner device instead of the friend
4 device.

5 On owner key SUSPEND/RESUME, manageKeyResponse from Vehicle OEM Server shall
6 contain friend Key Data for all active friend keys associated with the Vehicle.

7 *Figure 13-17: REV_600: Owner key suspension due to device reported lost/stolen*



8
9



3 13.6 Owner Device Unpairing

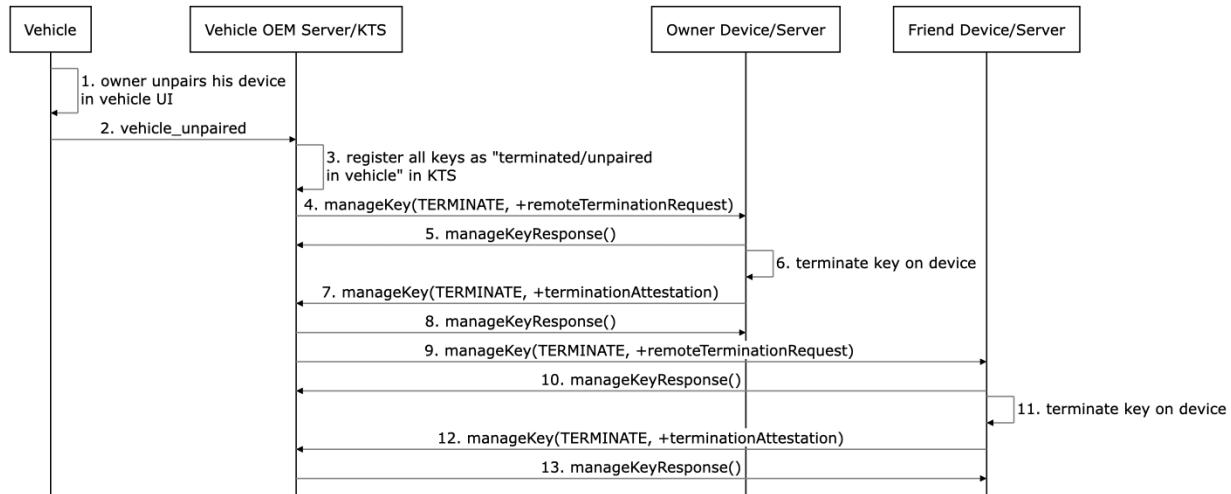
- 4 If the owner device is unpaired from the vehicle, all friend keys shall be deleted in the vehicle
5 and the friend devices should be notified.
6 The unpaired state shall be registered in the KTS. Key deletion on the owner and friend devices
7 is optional but recommended.

8 13.6.1 REV_700: Unpairing in vehicle UI (sell vehicle)

- 9 If unpairing is started from the vehicle, key deletion is executed under Vehicle OEM-defined
10 conditions, and the owner device is notified.

1

Figure 13-19: REV_700: Unpairing in Vehicle UI (Sale of Device)

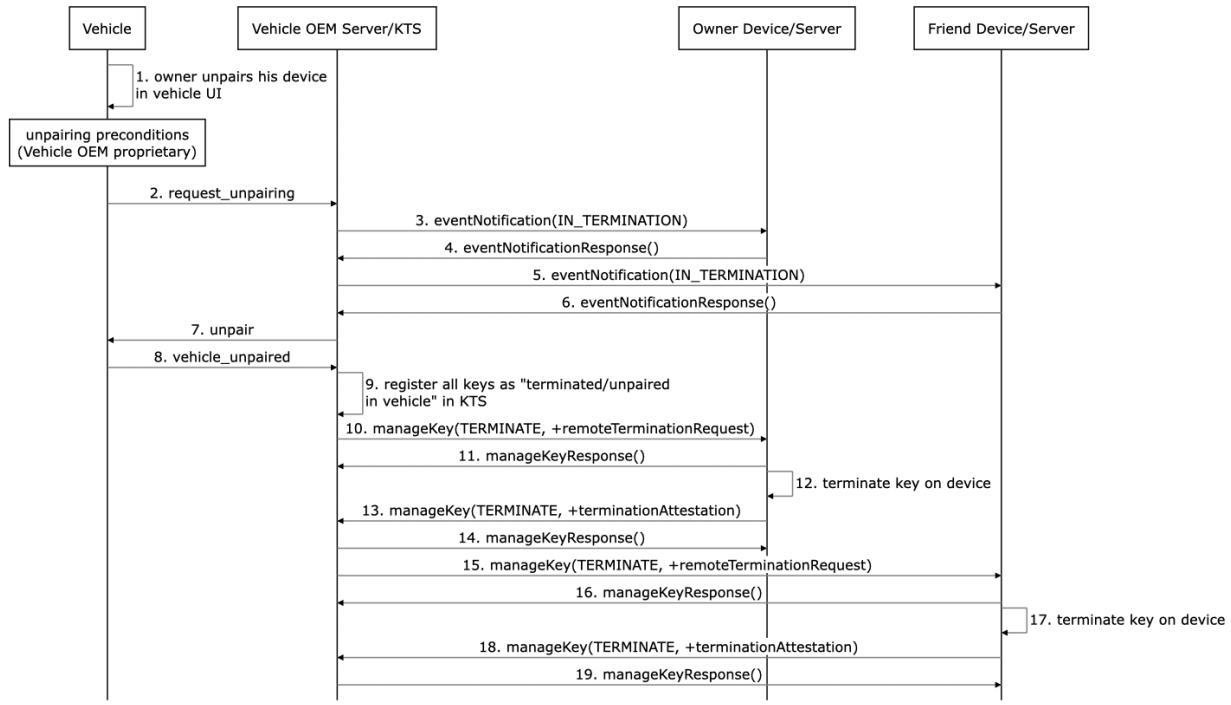


2

- 3 Alternatively, it may be required for the vehicle to request online unpairing. In that case, devices are informed about the termination request and the successful termination (See [Figure 13-20](#)).
 4 Note: Inability to reach owner or friend device in Steps 3 through 6 does not stop the process

6

Figure 13-20: REV_700a: Unpairing in Vehicle UI (Vehicle Required to be Online)

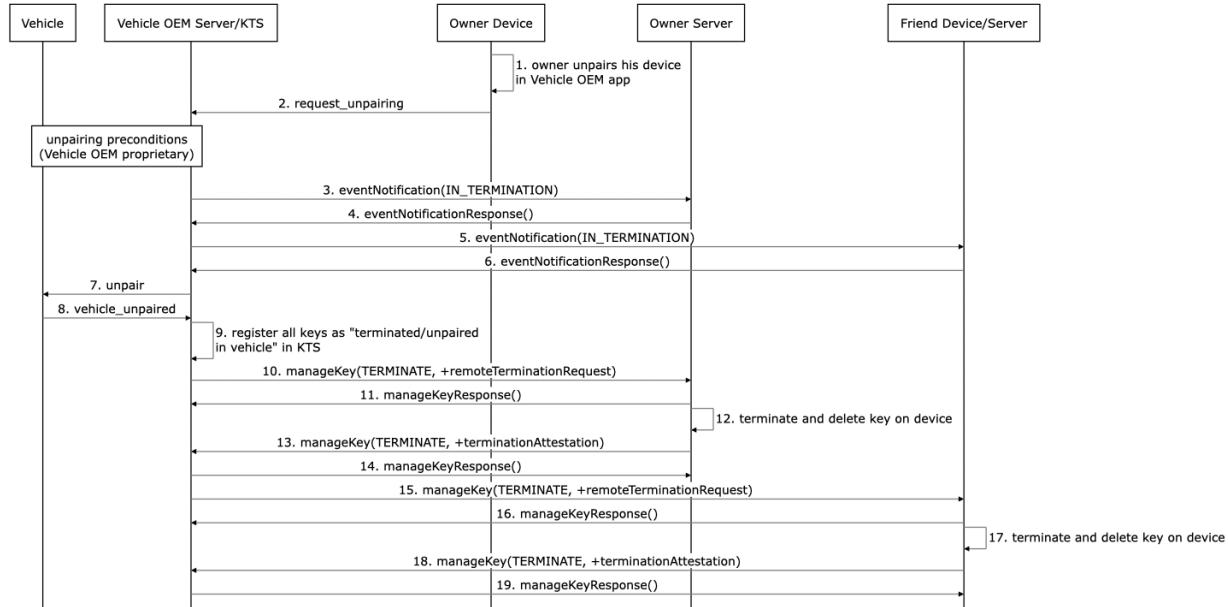


7

- 8 13.6.2 *REV_710: Unpairing on owner device in Vehicle OEM app*
 9 If the owner device is unpaired through the Vehicle OEM app, the app shall not start the unpairing process on the device. The app shall notify the Vehicle OEM Server about the

- 1 unpairing request, and the Vehicle OEM Server shall execute the unpairing process. Note:
2 Inability to reach friend device in Steps 5 through 6 does not stop the process.

3 *Figure 13-21: REV_710: Unpairing in Vehicle OEM App on Owner Device*



- 4
- 5 13.6.3 *REV_720: Unpairing in owner Vehicle OEM account (sale of vehicle)*
6 This flow is similar to REV_710 with the Vehicle OEM Server triggering the unpairing process.
- 7 13.6.4 *REV_730: Unpairing based on cancellation or expiry date of the Digital Key service*
8 This flow is similar to REV_710 with the Vehicle OEM Server triggering the unpairing process.
- 9 13.6.5 *REV_740: Unpairing by Vehicle OEM (vehicle stolen)*
10 This flow is similar to REV_710 with the Vehicle OEM Server triggering the unpairing process.
- 11 13.6.6 *REV_750: Unpairing by Vehicle OEM (security breach in vehicle)*
12 This flow is similar to REV_710 with the Vehicle OEM Server triggering the unpairing process.
- 13 13.6.7 *REV_760: Unpairing by Vehicle OEM (garage service process)*
14 This flow is similar to REV_710 with the Vehicle OEM Server triggering the unpairing process.
- 15 13.6.8 *REV_770: Unpairing due to deletion of personal data in owner Vehicle OEM account*
16 This flow is similar to REV_710 with the Vehicle OEM Server (account) triggering the unpairing process.

13.6.9 REV_800: Unbinding of vehicle from owner account in vehicle UI (unpairing, owner sale of vehicle)

3 This flow is similar to REV_710 with the vehicle UI triggering the unpairing process.

13.6.10 *REV_810: Unbinding of vehicle from owner account in owner Vehicle OEM account (unpairing, owner sale of vehicle)*

6 This flow is similar to REV_710 with the Vehicle OEM account triggering the unpairing
7 process.

13.6.11 *REV_820: Unbinding of vehicle from owner account in Vehicle OEM app (unpairing, owner sale of vehicle)*

10 This flow is similar to REV_710 with the Vehicle OEM app triggering the unpairing process.

13.7 Termination in Vehicle

13.7.1 *Rules*

13 Digital Key termination on the vehicle side requires protection against replaying all or a part of
14 an earlier key sharing process. The following preconditions shall apply:

- **C1:** The initial slot identifier (SlotID in the example below) values in the private mailbox shall be set to all bytes FF_h.

- **C2:** The initial Slot Identifier Bitmap (SlotID Bitmap in the example below) value is 00h.

18 The vehicle shall implement the following rules if Tag DA_h (as defined in Table 5-14) is set to
19 00_h (slot identifiers and immobilizer tokens) or 02_h (slot identifiers only) are provided by the
20 vehicle:

- **Rule V1:** Accept a key addition only if the friend slot identifier (as described in Section 4.3.1) is verified to be valid for an available key storage in the vehicle.

- **Rule V2:** When a key is deleted in the vehicle (from any interface), delete the data (device PK, etc.) in the corresponding key storage and assign a new slot identifier. The vehicle OEM shall implement appropriate means to ensure replay protection. This new slot identifier is available for future Digital Key sharing. When all bits in the Slot Identifier Bitmap are set to 1, the vehicle shall set the corresponding bit in the signaling bitmap to 0 to indicate that all slot identifiers are written. To avoid inconsistent states, either all of the described operations shall be done in one atomic transaction, or the vehicle shall verify the consistency of all bits and values after each console reader transaction and update accordingly.

- **Rule V3:** On each console reader transaction, the vehicle refills slot identifiers in each entry with bit=0 and sets the bit to 1 in an atomic transaction. The vehicle ensures the uniqueness of the slot identifiers and memorizes the next slot identifier to be written to the device.

36 All devices shall implement the following rules:

- **Rule D1** (only applies to owner device): Provide a matching pair of slot identifier and, if applicable, immobilizer token from the confidential mailbox to the friend when sharing a

1 Digital Key. Set the value in the corresponding bit of the Slot Identifier Bitmap in the
2 private mailbox to 0. Set the corresponding bit in the signaling bitmap to 1 if it was set to
3 0 before sharing. The device shall always use the lowest value of all slot identifiers for
4 key sharing.

- 5 • **Rule D2:** When a key is terminated in the device locally, send the termination attestation
6 and the Vehicle OEM proprietary data to the Vehicle OEM Server to trigger the deletion
7 of the Digital Key in the vehicle.

8 13.7.2 Example

9 The example below shows the principle of key termination and slot identifier refill as per the
10 above rules. In this example, the following notes apply:

- 11 1. Immobilizer tokens are not shown.
- 12 2. The vehicle has a number of key stores (Key Storage) in memory (0–10; see [Figure 13-22](#)).
- 13 3. Key storage 0 is reserved for the owner key.
- 14 4. The slot identifiers are refilled by the vehicle starting with “the highest slot identifier
15 value in the private mailbox of the device + 1.”
- 16 5. For all steps described in the example, the vehicle shall update the signaling bitmap after
17 writing the slot identifier bitmap and the slot identifiers to signal an update to the owner
18 device.
- 19 6. Slot identifiers and immobilizer tokens are provided by the vehicle.

20 [Figure 13-22](#) shows the device and vehicle memory after owner pairing and before the vehicle
21 has identified the immobilizer tokens and the corresponding slot identifiers for friend sharing (if
22 the immobilizer tokens are retrieved from the vehicle).

23 Initially all device-side key storages are set to FF_h FF_h (Rule C1). The signaling bitmap shows
24 that slot identifiers need to be refilled.

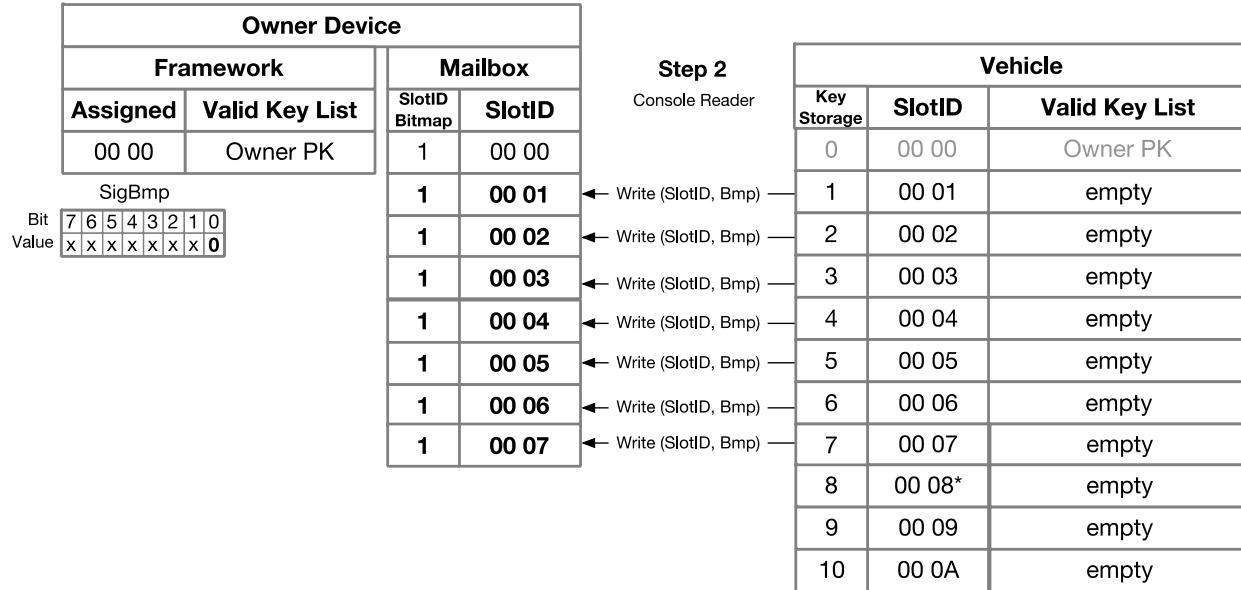
26 *Figure 13-22: Example Step 1: After Owner Pairing*

Owner Device		Step 1		Vehicle		
Framework		Mailbox		Key Storage	SlotID	Valid Key List
Assigned	Valid Key List	SlotID Bitmap	SlotID			
00 00	Owner PK	1	00 00	0	00 00	Owner PK
SigBmp		0	FF FF	1	00 01*	empty
Bit Value	7 6 5 4 3 2 1 0 x x x x x x x 1	0	FF FF	2	00 02	empty
		0	FF FF	3	00 03	empty
		0	FF FF	4	00 04	empty
		0	FF FF	5	00 05	empty
		0	FF FF	6	00 06	empty
		0	FF FF	7	00 07	empty
		0	FF FF	8	00 08	empty
		0	FF FF	9	00 09	empty
		0	FF FF	10	00 0A	empty

27 * next SlotID to be given to the owner

1 [Figure 13-23](#) shows device and vehicle memory after the vehicle has provisioned the slot
2 identifiers. All entries on the device containing FF_h FF_h have been filled with slot identifier
3 values, and the signaling bitmap has been updated to show that no slot identifier entries are
4 missing in SlotIdentLst (**Rule V3**).
5 The vehicle memorizes the next slot identifier to be provisioned into the device (*).
6

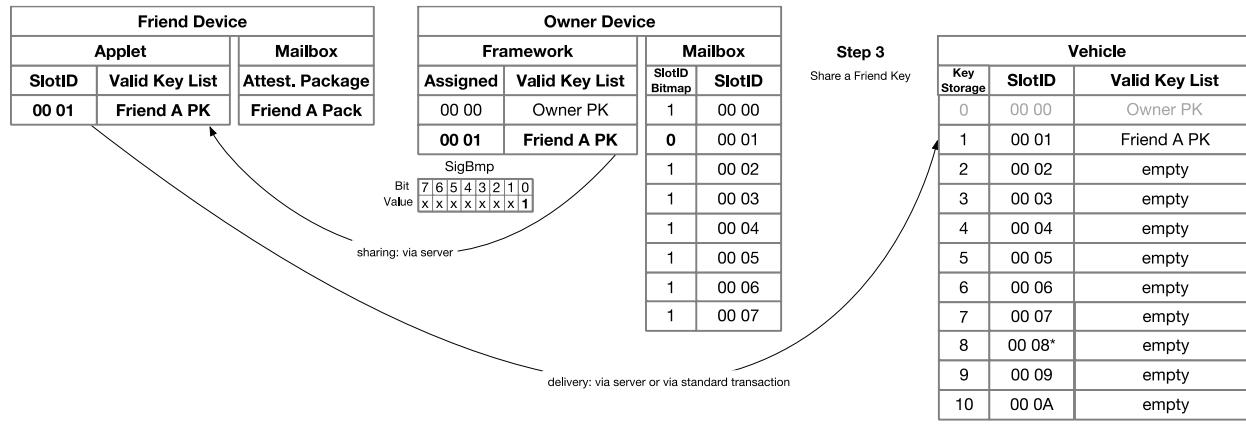
Figure 13-23: Example Step 2: Device Fully Refilled



* next SlotID to be given to the owner

7
8 In [Figure 13-24](#), the owner has shared one key with a friend, and the slot identifier bitmap and
9 signaling bitmap have been updated accordingly (**Rule D1**). The friend key is accepted by the
10 vehicle, either via server link or via a standard transaction (**Rule V1**). The slot identifier 00_h 01_h
11 has been used for the friend key (SlotID Bitmap bit= 0). The framework of the owner device
12 memorizes the assigned friend slot identifiers.

Figure 13-24: Example Step 3: Key Shared with Friend

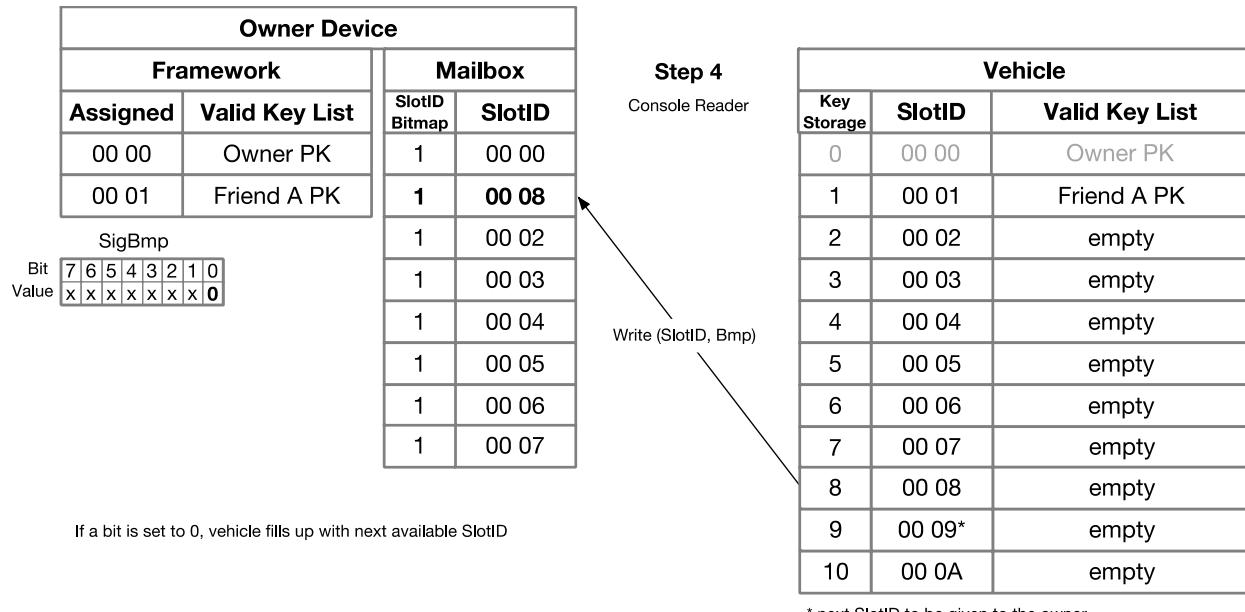


* next SlotID to be given to the owner

13
14

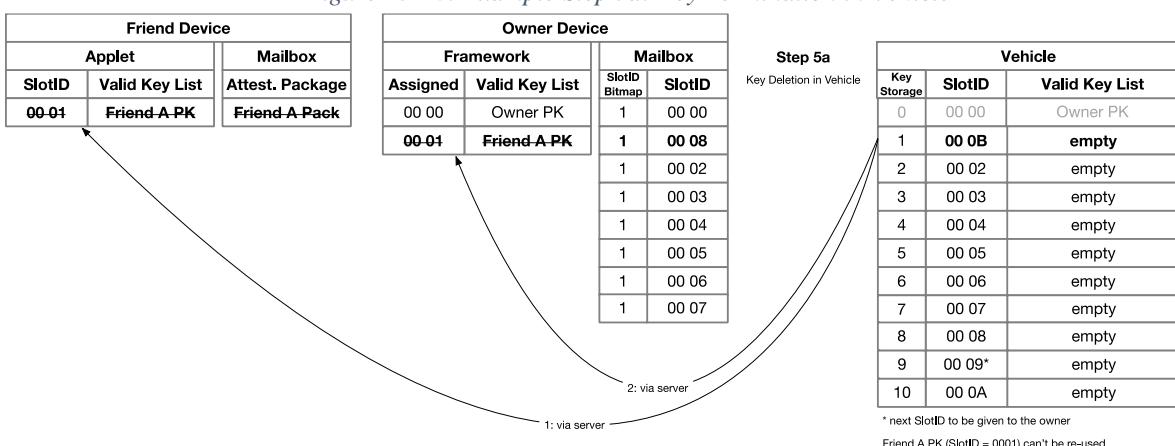
- 1 In [Figure 13-25](#), the vehicle has refilled the empty entry in the mailbox with the next available
2 slot identifier (00_h 08_h) and memorizes 00_h 09_h as the next available slot identifier. The
3 corresponding bit in the slot identifier bitmap has been set to 1 and the one in the signaling
4 bitmap has been set to 0 by the vehicle.

5 *Figure 13-25: Example Step 4: Refill of Shared Slot Identifier*



- 6
- 7 In [Figure 13-26](#), the friend key has been deleted in the vehicle (**Rule V2**). The vehicle has
8 emptied the key storage and has assigned slot identifier value 00_h 0B_h. When the device presents
9 the Digital Key (attestation package) of friend A to the vehicle again, the vehicle rejects the key
10 (**Rule V1**).

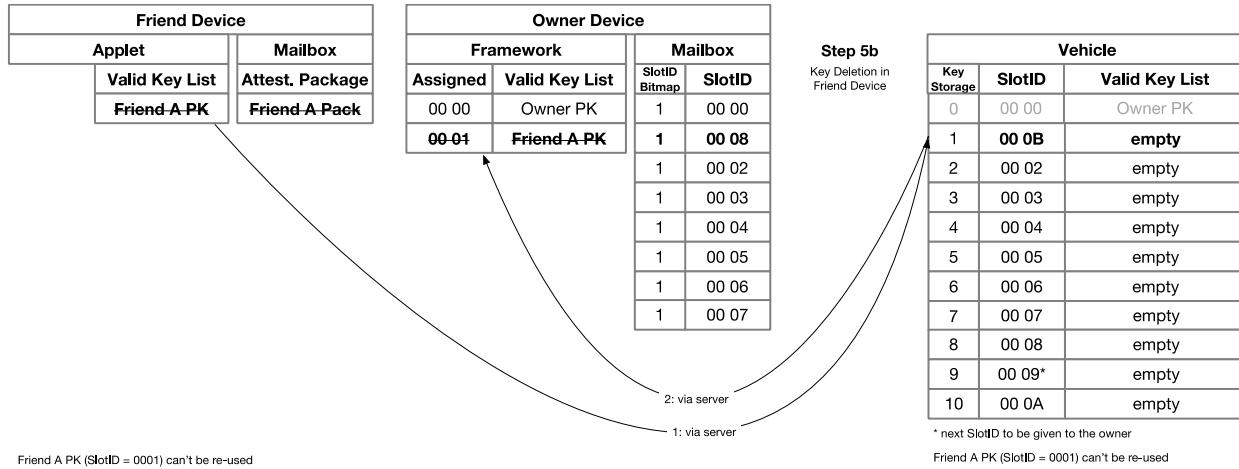
11 *Figure 13-26: Example Step 5a: Key Termination in Vehicle*



- 12
- 13 Key termination in the vehicle may be originated in the vehicle, from the Vehicle OEM Server,
14 from the Device OEM Server, or from the owner device. The key termination is propagated to
15 the friend device via the Vehicle OEM Server, which causes remote key deletion on the friend
16 device.

- 1 In an alternative, [Figure 13-27](#), the friend key has been deleted in the device (**Rule D2**). The
2 device deletes the key locally and sends the termination attestation and the Vehicle OEM
3 proprietary data to the Vehicle OEM Server to trigger the deletion of the Digital Key in the
4 vehicle. The vehicle removes the key as for a remote deletion (**Rule V2**).

5 *Figure 13-27: Example Step 5b: Key Termination in Device*



14 AUTHENTICATION AND PRIVACY

14.1 Authentication and Privacy Keys

14.1.1 Usage

Sensitive data exchanged between the device and the Vehicle OEM shall be encrypted. The currently identified data includes:

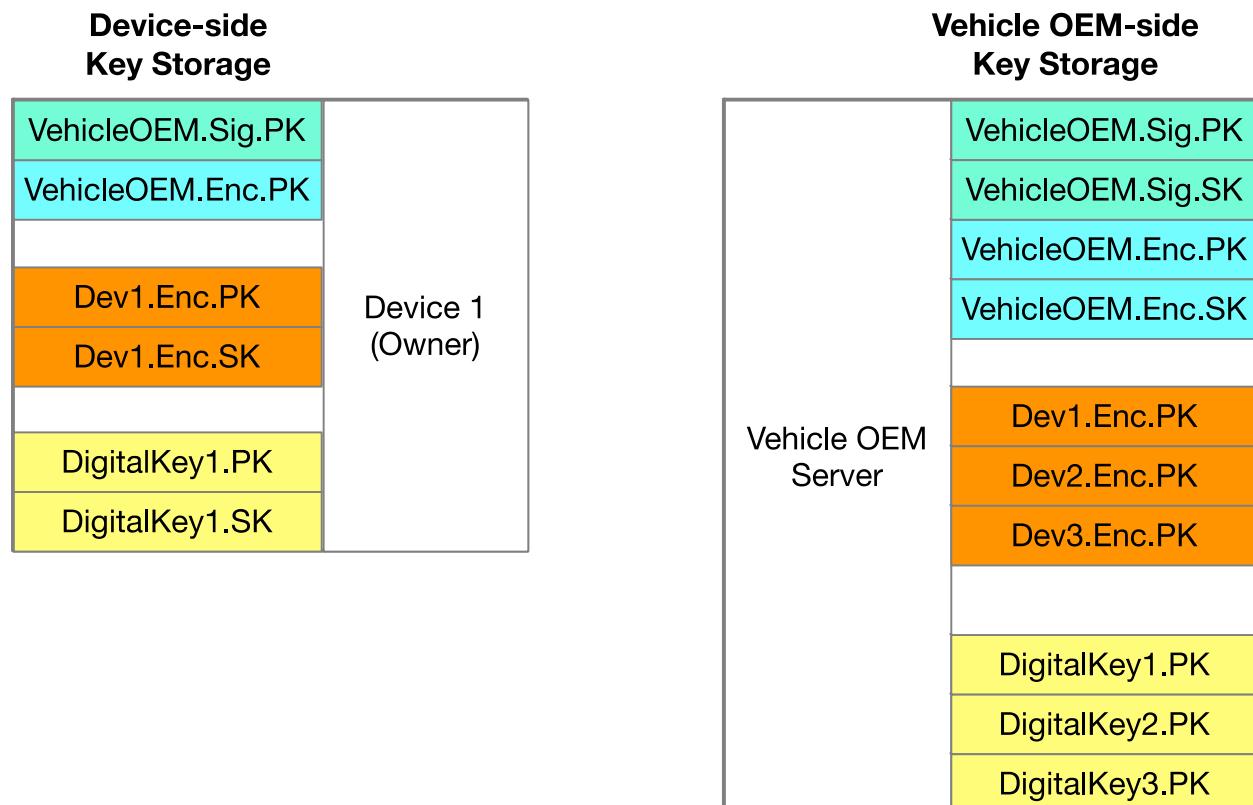
- Key tracking request data (content: attestation package, device encryption public key)
- Notification messages to the owner device that contain friend key information (e.g., the Digital Key identifier of a friend)
- Remote key deletion requests from the owner device that contain friend key information (e.g., the Digital Key identifier of a friend)

Remote key deletion requests that are sent from a device to the Vehicle OEM Server shall be authenticated through a signature.

Key deletion commands that are sent from the Vehicle OEM Server to the target device shall be authenticated through a signature.

Figure 14-1 shows the keys that are used for encryption/decryption and signature verification.

16 *Figure 14-1: Message Authentication and Privacy Encryption*



18 See Table 14-1 for a complete list of involved keys.

1

Table 14-1: Authentication and Privacy Keys

Key	Created by	Purpose	Description
VehicleOEM.Sig.SK	Vehicle OEM Server	Vehicle OEM Server signs the data sent to the device	Safely stored in Vehicle OEM Server
VehicleOEM.Sig.PK	Vehicle OEM Server	Device authenticates the data from the Vehicle OEM Server	Certificate pre-stored in device OS
VehicleOEM.Enc.SK	Vehicle OEM Server	Vehicle OEM Server decrypts the data sent from the device	Safely stored in Vehicle OEM Server
VehicleOEM.Enc.PK	Vehicle OEM Server	Device encrypts the data sent to the Vehicle OEM Server	Certificate pre-stored in device OS
Devx.Enc.SK	Device	Device decrypts the data sent from the Vehicle OEM Server	Created at owner pairing/key sharing, safely stored in the device OS if not already existing
Devx.Enc.PK	Device	Vehicle OEM Server encrypts the data sent to the device	Created at owner pairing/key sharing, sent in key tracking request
DigitalKeyx.SK	Device	Device signs the data sent to the Vehicle OEM Server	This is the Digital Key SK, safely stored in the SE
DigitalKeyx.PK	Device	Vehicle OEM Server authenticates the data sent from the device	This is the Digital Key PK

2 The privacy and authentication keys are used in the following cases, described in detail in
3 Section [17](#):

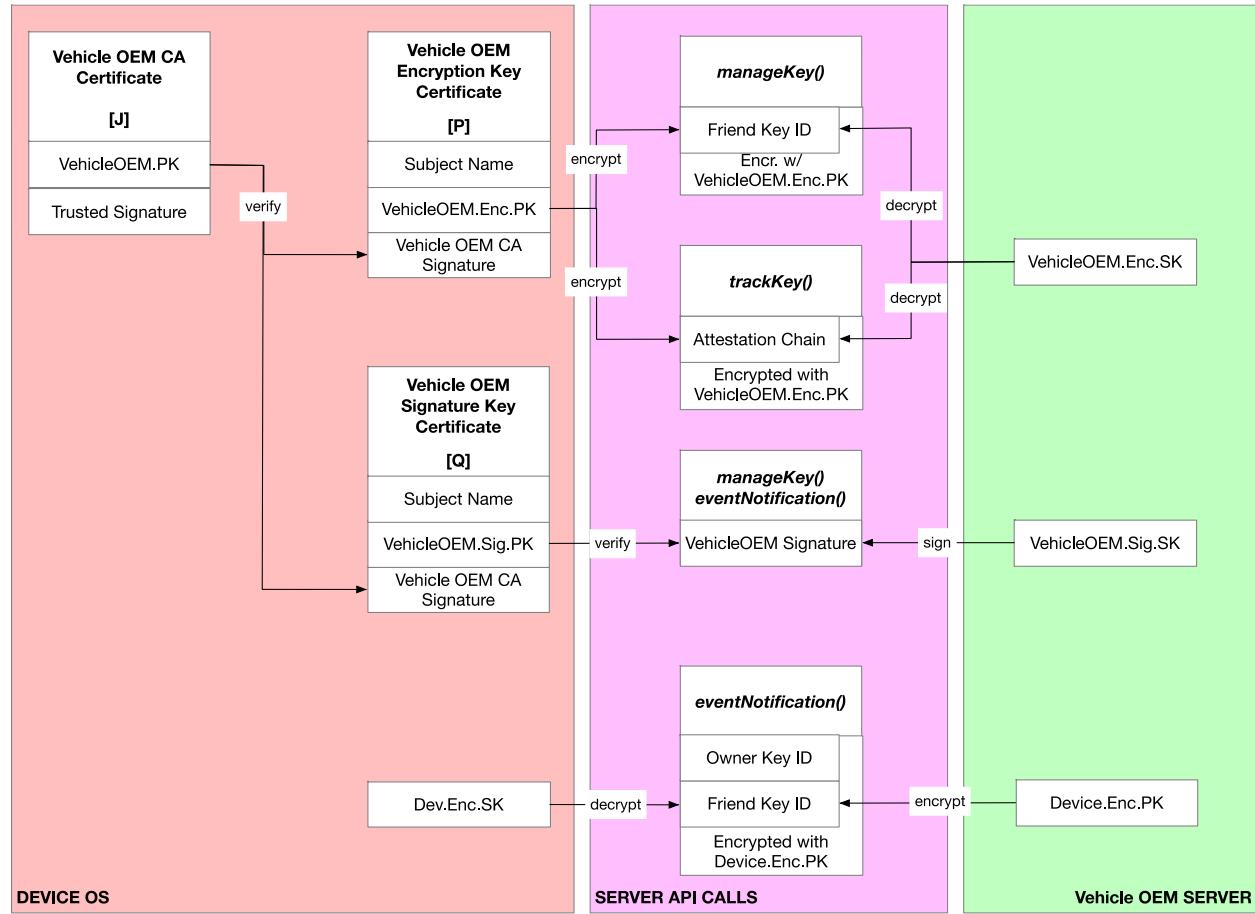
- 4 1. The key tracking request contains Devx.Enc.PK as part of the encrypted payload. The
5 payload is encrypted using VehicleOEM.Enc.PK with the encryption scheme described in
6 Section [14.3](#). The Vehicle OEM Server stores Devx.Enc.PK associated with the Digital
7 Key identifier provided in the attestation chain.
- 8 2. The remote (Digital Key) deletion request sent from the owner device to the Vehicle
9 OEM Server contains the Digital Key identifiers of owner and friend in a structure
10 described in Section [14.2](#). The request data is signed using the owner private key
11 (DigitalKeyx.SK) and encrypted using VehicleOEM.Enc.PK, as it contains the Digital
12 Key identifier of a friend. The signature scheme is described in Section [14.3](#).
- 13 3. The key deletion request sent from the Vehicle OEM Server to the target device is signed
14 by the Vehicle OEM Server using the VehicleOEM.Sig.SK. The signature is verified by
15 the target device using VehicleOEM.Sig.PK.
- 16 4. The notification which is sent from the Vehicle OEM Server to the owner device is
17 encrypted using Devx.Enc.PK, as it contains the Digital Key identifier of a friend.

18 14.1.2 Certificate Chains

19 All public keys used for authentication and privacy are embedded in X.509 certificates except
20 the Device Privacy Encryption Key, and are chained to the trusted root as shown in [Figure 14-2](#).
21 The trust chain of all certificates shall be verified before using them.

- 1 The usage of each certificate's public key is described in [Table 14-1](#) and in Section [14.1.1](#). All
2 private keys are assumed to be stored safely where they have been created.

3 *Figure 14-2: Authentication and Privacy Encryption Certificate Chain*



- 5 *14.1.2.1 [H] – Digital Key Creation Attestation*
6 See Section [16.2.8](#) for a description of this certificate.

- 7 *14.1.2.2 [J] – Vehicle OEM CA Certificate*
8 See Section [16.2.9](#) for a description of this certificate.

- 9 *14.1.2.3 [P] – Vehicle OEM Privacy Encryption Certificate*
10 This certificate is generated by the Vehicle OEM Server and provided to the Device OEM. Each
11 Device OEM shall receive a different public key in the certificate. The transfer of the certificate
12 from the Vehicle OEM Server via the Device OEM Server to the device is out of scope of this
13 specification. The certificate should have a lifetime defined by Vehicle OEM policy. Revocation
14 may be achieved by stopping the renewal process or through revocation methods (CRL, OCSP),
15 which are out of scope of this specification.
16 The certificate shall have proprietary extension using the assigned OID listed in Appendix [B.2](#).
17 The extension is defined as shown in [Listing 14-1](#) and [Listing 14-2](#).

1 *Listing 14-1: Vehicle OEM Privacy Encryption Certificate Extension Schema*

```
1 VehicleOEMPrivacyEncryptionCertificateExtensionSchema ::= SEQUENCE
2 {
3   extension_version INTEGER (1 ..255)
4 }
```

2 *Listing 14-2: Vehicle OEM Privacy Encryption Certificate Extension Data*

```
1 vehicle-oem-privacy-encryption-cert-extension-data VehicleOEMPrivacyEncryptionCertificateExtensionSchema :=
2 {
3   extension_version 1 --value shall be 1
4 }
```

3 *14.1.2.4 VehicleOEM Signature Key Certificate*

4 This certificate is generated by the Vehicle OEM Server and provided to the Device OEM. Each
5 Device OEM shall receive a different public key in the certificate. The transfer of the certificate
6 from the Vehicle OEM Server via the Device OEM Server to the device is out of scope of this
7 specification. The certificate should have a lifetime defined by Vehicle OEM policy. Revocation
8 may be achieved by stopping the renewal process or through revocation methods (CRL, OCSP),
9 which are out of scope of this specification

10 The certificate shall have a proprietary extension using the assigned OID listed in Appendix
11 **B.2.2**. The extension is defined as shown in [Listing 14-3](#) and [Listing 14-4](#).

12 *Listing 14-3: Vehicle OEM Signature Certificate Extension Schema*

```
1 VehicleOEMSSignatureCertificateExtensionSchema ::= SEQUENCE
2 {
3   extension_version INTEGER (1 ..255)
4 }
```

13 *Listing 14-4: Vehicle OEM Signature Certificate Extension Data*

```
1 vehicle-oem-signature-cert-extension-data VehicleOEMSSignatureCertificateExtensionSchema :=
2 {
3   extension_version 1 --value shall be 1
4 }
```

14 *14.1.2.5 Device Privacy Encryption Key*

15 This key is generated by the device and provided to the Vehicle OEM Server in the TrackKey()
16 API call (see Section [17.7](#)). Each device shall generate a unique device encryption key pair per
17 Digital Key [H].

18 **14.2 Remote Termination Requests**

19 The Remote Termination Request shall be sent when either the owner device or the Vehicle
20 OEM Server requests the deletion of a Digital Key on a target device.

21 If the owner device originates the Remote Termination Request, the request structure in [Table](#)
22 [14-3](#) shall be used.

23 If the Vehicle OEM Server or the vehicle itself originate the Remote Termination Request, the
24 request structure in [Table 14-4](#) shall be used. A specifically defined source key id indicates
25 whether the request is originated by the vehicle or the Vehicle OEM Server (owner or friend
26 account, subscription ended, etc.).

- When the key id of the target key is not known to the source device, the target key id tag shall be absent.
- If the key id of the target key is known to the source device but not to the server and vehicle, the slot identifier tag shall be used to identify the key to terminate. The corresponding slot identifier shall be deleted in the server and vehicle. When the target device attempts to track the key after its deletion, the track key request shall be rejected.
- In some cases, the slot identifier might not be known. In those cases, the key identifier shall be used to identify the key to be deleted.

[Table 14-2](#) describes the data fields of the Remote Termination Request.

Table 14-2: Remote Termination Request Data Fields

Tag	Length (bytes)	Description	Field is	Domain Version
5F21 _h	20	subject key identifier (key id) of the source Digital Key	mandatory	D-VS
7F23 _h	variable	list of key identifier and slot identifier pairs of the target Digital Key(s)	mandatory	D-VS
61 _h	variable	pair of key identifier and slot identifier of 1 st Digital Key	mandatory	D-VS
50 _h	20	subject key identifier	conditional	D-VS
57 _h	variable	slot identifier	conditional	D-VS
61 _h	variable	pair of key identifier and slot identifier of 2 nd Digital Key	conditional	
50 _h	20	subject key identifier	conditional	
57 _h	variable	slot identifier	conditional	
...	

- The request shall contain either subject key identifier or slot identifier in element (tag 61_h). It should contain both if they are available.
- The request may contain one or more pairs of target slot identifier and target key identifier. The maximum number of keys in the request may be defined by the Vehicle OEM; otherwise it is limited to 16.
- The input data fields for signature creation for the remote termination request for the owner device and the Vehicle OEM Server are described in [Table 14-3](#) and [Table 14-4](#), respectively.
- The signature creation for the remote termination request is described in [Table 14-3](#).

Table 14-3: Remote Termination Request Owner Device

Tag	Length (bytes)	Description	Field is	Domain Version
7F40 _h	variable	Remote Termination Request owner device	mandatory	D-VS
content of Table 14-2			mandatory	D-VS

Tag	Length (bytes)	Description	Field is	Domain Version
		SIG-DAT: content of Table 15-56 (Signature Data fields) with: arbitrary_data_ = SHA-256 hash value of Table 14-2	mandatory	V-OD-FW
9E _h	64	signature with the private key of the owner Digital Key over fields from SIG-DAT	mandatory	V-OD-FW

1 Note: The counter_value field (deprecated) in [Table 15-56](#) is not present for SIG-DAT
2 computation.

3 *Table 14-4: Remote Termination Request from Vehicle OEM Server*

Tag	Length (bytes)	Description	Field is	Domain Version
7F39 _h	variable	Remote Termination Request from Vehicle OEM Server	mandatory	D-VS
		content of Table 14-2 with: source key id = <vehicle key id>, if originated in vehicle source key id = <vehicle OEM key id>, if originated in Vehicle OEM Server source key id = <owner key id>, if originated by owner device via Vehicle OEM Server		D-VS
5F22 _h	20	Subject key identifier of the certificate to be used for signature verification (VehicleOEM.Sig.Cert)	Mandatory	D-VS
9E _h	64	signature with the private key of the Vehicle OEM Server (VehicleOEM.Sig.SK) over fields from Table 14-2	mandatory	D-VS

- 4 The <vehicle OEM key id> is defined as 20 bytes of FF_h. It is used when the key was deleted by
5 the Vehicle OEM (e.g., subscription ended).
- 6 The <vehicle key id> is defined as 20 bytes of EE_h. It is used when the key was deleted in the
7 vehicle.
- 8 <owner key id> is provided in the remote termination request coming from the owner. It is used
9 when the key was deleted by the owner device.
- 10 The <device OEM key id> is defined as 20 bytes of DD_h. It is used when the key was deleted by
11 the Device OEM (e.g. remote device wipe).
- 12 The signature covers tags 5F21_h and 7F23_h, inclusive of tags and lengths, as defined in [Table 14-2](#).
- 14 When receiving a remote termination request from the owner device ([Table 14-3](#)), the Vehicle
15 OEM Server shall verify the following before executing the termination request:
- 16 • Value of the hash of [Table 14-2](#) matches the arbitrary_data value provided in SIG-DAT
17 • Signature is correct
- 18 When receiving a remote termination request from the Vehicle OEM Server ([Table 14-4](#)), the
19 friend device shall verify the following before executing the termination request:
- 20 • Source key ID corresponds to the key ID of the originator
21 • Signature is correct
- 22 The Device OEM Server shall ensure the following security properties:

- Each termination request from a mobile device to the Device OEM Server shall contain a signed terminationAttestation (owner or friend device deletes its own key) or remoteTerminationRequest (owner deletes friend key).
- Each termination request and each termination response from a mobile device to the Device OEM Server shall contain the Vehicle OEM proprietary data subsection of the private mailbox.
- The user can delete only his/her own keys in the device portal, and not the shared keys.
- When keys are deleted in the Device OEM portal (see [Table 13-4](#)) and the device could not be reached, the Device OEM Server is not required to provide a remoteTerminationRequest to the Vehicle OEM Server.

14.2.1 Example of RTR Signature Verification

RTR –

7F397C5F2114FFFFFFFFFFFFFF7F2320611E5014444DF
CECAA57A75B4C42D60A7EB1868B793BEDE05706000F000000019E40CD846C5BCFAB51
41CE129A7C197129220EB5747B9A80C1726DEA200FFA42B2178235452B60DD60AA8D7F
9EB43EA486A87C91B9BC9886D83DAE9583F1112D183A

SIG –

CD846C5BCFAB5141CE129A7C197129220EB5747B9A80C1726DEA200FFA42B217823545
2B60DD60AA8D7F9EB43EA486A87C91B9BC9886D83DAE9583F1112D183A

TBS –

5F2114FFFFFFFFFFFFFF7F2320611E5014444DFCECAA
57A75B4C42D60A7EB1868B793BEDE05706000F00000001

PK -

043517d1a4043c6738a2fc0458fecd4ae94c833b7fac59fbf697ee65979bcd64cbc3ce7323a6c5f0c6
6e9500734e33f0bd93c280b7299fc85674bbc64766492e25

>>> pubkey = bytes.from-
hex('3517d1a4043c6738a2fc0458fecd4ae94c833b7fac59fbf697ee65979bcd64cbc3ce7323a6c5f0
c66e9500734e33f0bd93c280b7299fc85674bbc64766492e25')

>>> data = bytes.from-
hex('5F2114FFFFFFFFFFFFFF7F2320611E5014444DFCE
CAA57A75B4C42D60A7EB1868B793BEDE05706000F00000001')

>>> sig = bytes.from-
hex('CD846C5BCFAB5141CE129A7C197129220EB5747B9A80C1726DEA200FFA42B2178
235452B60DD60AA8D7F9EB43EA486A87C91B9BC9886D83DAE9583F1112D183A')

>>> vk = ecdsa.VerifyingKey.from_string(pubkey, curve = curve)

>>> vk.verify(sig, data, hashfunc = hashfunc)

True

14.3 Encryption and Signature Verification Schemes

The data to be privacy-encrypted is described in Section 17. The encryption scheme to be used is described in Section 17.11. Signatures of the commands shall use ANSI X9.62 ECDSA algorithm with SHA-256 (see [23] and [24]).

14.4 OEM App Data Attestation

OEM App Data Attestation shall be used to obtain a signature over data provided by the OEM App, using the endpoint.SK. In response, all fields of the returned attestation shall be provided to the Vehicle OEM App. For OEM App Data Attestation, a request for User Consent shall be enforced. Enforcement of User Authentication is optional.

Table 14-5 describes the input fields of the OEM App Data Attestation.

Table 14-5: OEM App Data Attestation Input

ASN.1 Tag	Length (bytes)	Description	Field is
04 _h	variable	App Bundle Identifier	mandatory
04 _h	variable	OEM App Data	mandatory
04 _h	variable	Nonce, defined by framework	mandatory

Attestation is generated as described in Listing 14-5.

Listing 14-5: OEM App Data Attestation Processing

```
Input: Table 14-5
output: attestation
begin
    perform user consent
        return error if user did not consent
    perform user authentication if required
    arbitrary_data = SHA-256 hash value of Table 14-5
    send SIGN command to retrieve attestation as described in Table 15-61
    return attestation
end
```

15 DIGITAL KEY APPLET

15.1 Introduction

The Digital Key applet is aimed at providing multi-purpose SE-based transaction mechanisms combined with peer-to-peer key distribution and a data storage system with strong security and privacy properties. Three types of contactless transaction may be used: standard transaction (see Section 7), fast transaction (see Section 8), and check presence transaction (see Section 10).

In this specification, two applet implementation models are provided, depending on the Device OEM's implementation or the Digital Key service deployment model.

- **SE-centric applet model:** For this model, the Device OEM CA Certificate's corresponding public key is protected by the SE, and the non-SE endpoints (e.g., vehicle, server, etc.) are verified by the SE.
- **Framework-centric applet model:** For this model, the Device OEM CA Certificate's corresponding public key is protected by the device OS native key store, and the non-SE endpoints (e.g., vehicle, server, etc.) are verified by the framework.

15.2 Keys and Data

This section defines some of the key and data elements used later in Section 15 to help understand the flow diagrams.

Cryptogram: MAC value calculated over unique public transaction data. The cryptogram proves that the device is in possession of the same symmetric key as the vehicle.

Kpersistent: Symmetric long-term key that is used to derive encryption and MAC session keys. It is stored in NVM on both vehicle and device sides.

Kenc: Derived symmetric key used to encrypt confidential commands and responses payloads

Kmac: Derived symmetric key used to calculate command MACs

Krmac: Derived symmetric key used to calculate response MACs

Kcmac: Derived symmetric key used to calculate cryptograms

Keseed: Symmetric that is used to derive keys for confidential data encryption and MAC

Keenc: Derived symmetric key used to encrypt confidential data

Kemac: Derived symmetric key used to compute MAC on encrypted confidential data

Kdh: Symmetric key generated by a Diffie-Hellman operation

***ePK/*eSK:** denotes ephemeral public and private keys

***PK/*SK:** denotes public and private keys

transaction_identifier: randomly generated nonce value and used on vehicle and device sides for symmetric key derivation and MAC/signature generation/verification

vehicle_identifier: unique identifier of the vehicle. On device side, it is used to look up the correct endpoint

15.3 Applet Implementation

15.3.1 Introduction

The following subsections describe internal data structures and APDU commands for the applet.
This specification supports only one Digital Key applet protocol version. The Digital Key applet protocol version is set to 0100_h.

15.3.1.1 TLV Field

The various Tag Length Value fields presented in this document shall comply with the BER-TLV format as defined in ISO 7816-4 [1]. The TLV fields shall be ordered as described in this specification; a different field order is considered invalid unless specified otherwise. The nesting level is represented by indentation of Tag values in the Tag column.

15.3.1.2 Applet Memory Types

The naming conventions for the memory types used in the applet are as follows:

- Variable prefixed with “nvm.” Indicates the memory used is persistent after power off.
- Variable prefixed with “cod.” indicates the memory used is volatile and erased after applet deselection (caused by contactless field off or explicit deselection).
- Variable noted as (nvm/cod).object.variable indicates the variable is an object member.
- Variable with neither memory prefix nor object prefix indicates a local volatile variable erased after exiting the current programming context.
- All variables and objects are globally available at the instance level except the local variables defined above.
- Variable suffixed with [] indicates a table of variables.
- Variable suffixed with * indicates all the variables names starting with the string present before *.

15.3.1.3 Endpoint Life Cycle States

state_endpoint_active

In this state, the endpoint accepts all commands allowed by its configuration.

State_endpoint_terminated

In this state, the endpoint only accepts the TERMINATE ENDPOINT and DELETE ENDPOINT commands.

15.3.1.4 Communication Interfaces

The wired interface connects the SE to the device’s Digital Key framework through physical wires inside the device. The contactless interface connects the SE to the NFC reader in the vehicle through radio interface. The notifications sent from the applet to the Digital Key framework (denoted as notify_*) may be implemented as described in Section [15.3.1.9](#) or using a proprietary method.

1

Table 15-1: Command Availability on Interfaces

Command	Wired Interface	Contactless Interface	Comment	Section
SELECT	Yes	Yes		15.3.2.1
CREATE CA	Yes	No		15.3.2.2
DELETE CA	Yes	No		15.3.2.3
CREATE ENDPOINT	Yes	No		15.3.2.4
TERMINATE ENDPOINT	Yes	No		15.3.2.5
DELETE ENDPOINT	Yes	No		15.3.2.6
GET PRIVATE DATA	Yes	No		15.3.2.18
SET PRIVATE DATA	Yes	No		15.3.2.19
AUTHORIZE ENDPOINT	Conditional	No	Availability configured in Section 15.3.2.4	15.3.2.7
SET CONFIDENTIAL DATA	Yes	No		15.3.2.20
CREATE ENCRYPTION KEY	Yes	No		15.3.2.17
VIEW	Yes	No		15.3.2.8
SETUP ENDPOINT	Yes	No		15.3.2.21
SETUP INSTANCE	Yes	No		15.3.2.22
AUTH0	Conditional	Mandatory	Availability configured in 15.3.2.4	15.3.2.9
AUTH1	Conditional	Mandatory	Availability configured in Section 15.3.2.4	15.3.2.10
PRESENCE0	No	Yes		15.3.2.11
PRESENCE1	No	Yes		15.3.2.12
READ BUFFER	Yes	No		15.3.2.13
WRITE BUFFER	Yes	No		15.3.2.14
EXCHANGE	Conditional	Yes	Availability configured in Section 15.3.2.4	15.3.2.15
CONTROL FLOW	Yes	Yes		15.3.2.16
SIGN	Conditional	No	Availability configured in Section 15.3.2.4	15.3.2.23
MANAGE UA	Conditional	No	Availability configured in Table 15-5	15.3.2.24
CREATE RANGING KEY	Conditional	No	Availability depends on platform	15.3.2.26
DELETE RANGING KEYS	Conditional	No	Availability depends on platform	15.3.2.27
GET NOTIFICATION	Optional	No	Availability depends on the implementation	15.3.2.28

2

1 Implementations may require the use of a SCP such as SCP03 [7] or SCP11 [28] between the
2 Digital Key applet and the Digital Key framework. In these cases, the Digital Key applet shall
3 support usage of a SCP for the following commands: CREATE ENDPOINT, TERMINATE
4 ENDPOINT, DELETE ENDPOINT, GET PRIVATE DATA, SET PRIVATE DATA, VIEW,
5 SETUP ENDPOINT, SETUP INSTANCE, READ BUFFER, WRITE BUFFER, SIGN,
6 AUTHORIZED ENDPOINT, SET CONFIDENTIAL DATA, CREATE ENCRYPTION KEY
7 and MANAGE UA-. This option of the Digital Key applet implementation is called Option A
8 (i.e., DK applet that supports SCP) in this section. The provisioning of secure channel keys is out
9 of scope of this specification.

10 The commands set over the contactless interface are standardized commands. SELECT, AUTH0,
11 AUTH1, PRESENCE0, PRESENCE1, EXCHANGE, and CONTROL FLOW commands are
12 mandatory to be supported; all other commands are optional.

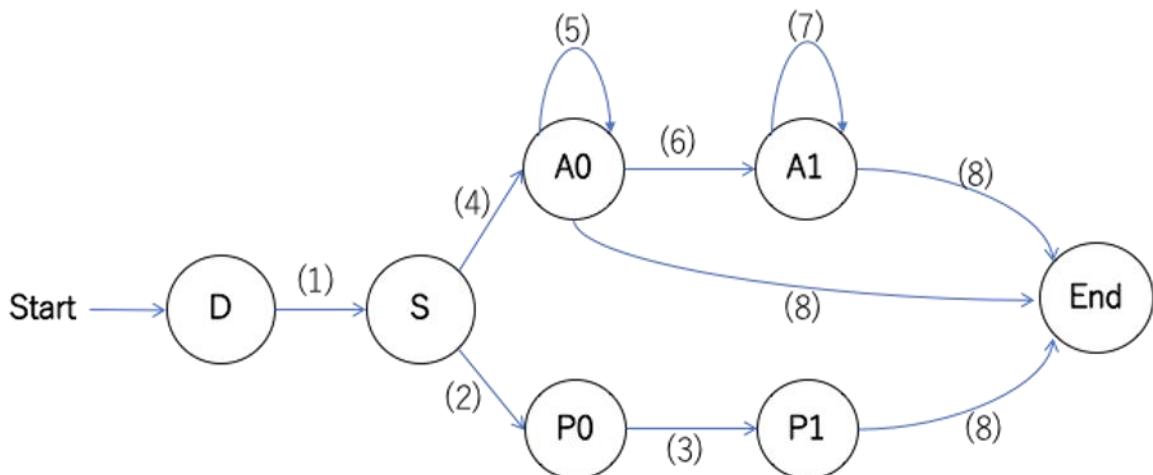
13 Over the wired interface, implementations may require the support of supplementary logical
14 channels in addition to support for the basic logical channel. This option of the Digital Key
15 applet implementation is called Option B (i.e., DK applet that supports supplementary logical
16 channels) in this section.

17 *15.3.1.5 Command Flows*

18 [Figure 15-1](#) describes the allowed command flows over the contactless interface; other command
19 flows are rejected. The CONTROL FLOW command (which can be used anywhere after the
20 SELECT command) is not represented in this figure.

1

Figure 15-1: Authentication Command Flows Over Contactless Interface



State

D: Applet Deselected
S: Applet Selected
A0: AUTH0 done
A1: AUTH1 done
P0: PRESENCE0 done
P1: PRESENCE1 done

Action

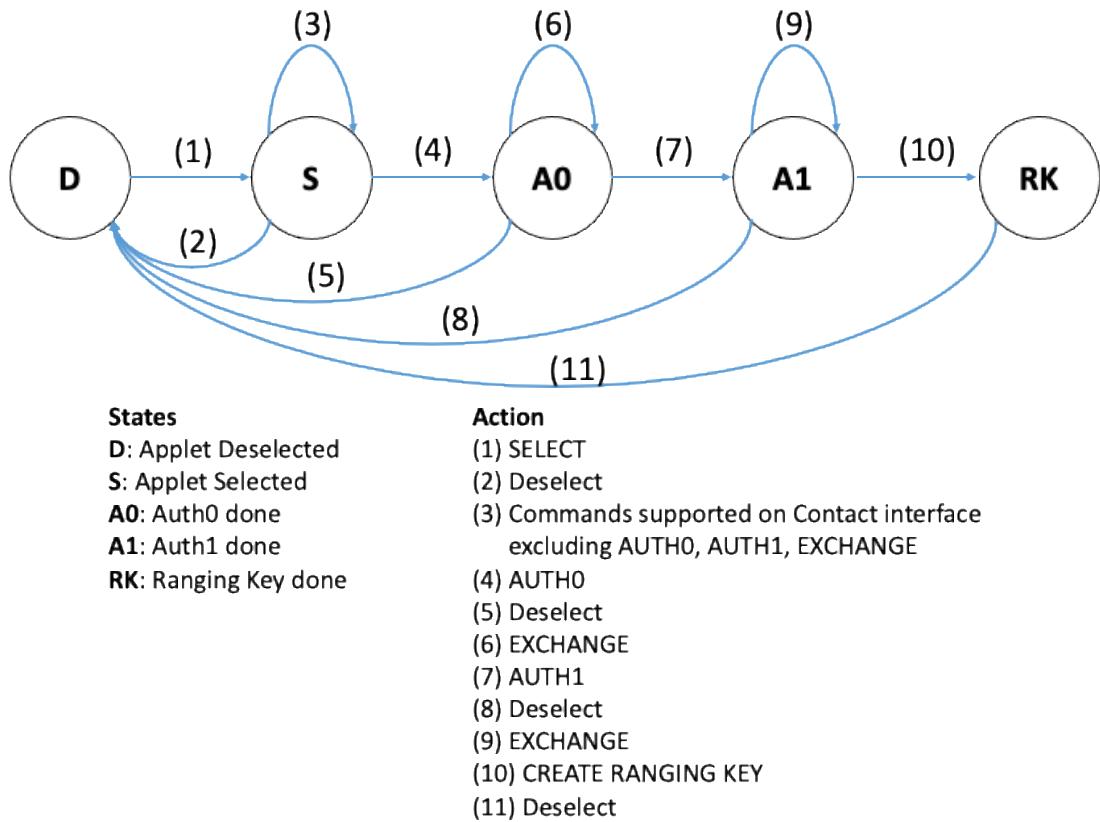
(1) SELECT
(2) PRESENCE0
(3) PRESENCE1
(4) AUTH0
(5) EXCHANGE
(6) AUTH1
(7) EXCHANGE
(8) NFC Link Teardown or
NFC Reset Procedures

2

3 Figure 15-2 describes the allowed command flows over the wired interface. The CONTROL
4 FLOW command (which can be used anywhere after the SELECT command) is not represented
5 in this figure.

6 *Figure 15-2: Authentication Command Flows Over Wired Interface*

7



In state A0, only AUTH1 and EXCHANGE commands are accepted. In state A1, only EXCHANGE and CREATE RANGING KEY commands are accepted.

GET NOTIFICATION command, if implemented, may be sent at any time after the execution of a command. It has no impact on the state

- 1
- 2 *15.3.1.6 Generic Error Handling*
- 3 This section applies to all listed commands. It describes the generic status words to be retrieved in case of error during basic input command checking.
- 4
- 5 The basic input command checking includes checking that an INS is allowed on a given interface, that the CLA is consistent, that P1/P2 bytes have valid values, that Lc is in valid range, and checking the format of the payload.
- 6
- 7
- 8 The basic input command checking is executed before the steps described in the Listings below and can result in the error status words described in [Table 15-2](#).
- 9
- 10 This specification does not describe all other possible errors that can occur during a command processing when not specified in the command listing. In such cases, the choice of the status word is left to the implementer. However, usage of status word 6400_h is highly recommended to ensure interoperability.
- 11
- 12
- 13
- 14 Digital Key applet should return an error if an APDU command includes unexpected or unknown parameters or tags
- 15

1

Table 15-2: Generic Status Words

Status Word	Comment
6400 _h	No specific diagnostic
6A80 _h	Wrong payload format
6A84 _h	Not enough memory
6A86 _h	Wrong P1 or P2
6A88 _h	Reference data not found
6700 _h	Wrong command payload length
6D00 _h	Wrong INS code
6E00 _h	Wrong CLA code
9000 _h	Command successfully executed

2 15.3.1.7 Class byte coding

3 The class byte coding of all commands shall conform to Section 5.4 of [1]. The value of the class
4 byte depends on which logical channel (basic or supplementary) the command is addressed for
5 and on whether or not secure messaging is used. As a result:

- 6 • When the implementation supports Option B, a range of class byte values is applicable to
7 all the commands available on the wired interface. For commands addressed to the basic
8 logical channel or supplementary logical channels 1, 2, and 3, the class byte coding is
9 defined in Table 2 of [1]. For commands addressed to supplementary logical channels 4
10 through 19, the class byte coding is defined in Table 3 of [1].
11 • When the implementation supports Option A, a range of class byte values is applicable to
12 the subset of commands defined in Section 15.3.1.4 when a secure channel is used
13 between the Digital Key applet and the Digital Key framework.

14 For commands addressed to the basic logical channel or to supplementary logical channels 1, 2,
15 and 3:

- 16 • Bit b8 is set to 0 for the SELECT command and to 1 for all other commands defined in
17 this specification.
18 • Bits b7 to b5 are set to 0.
19 • When a secure channel is used, bits b4 and b3 are set to 01; otherwise they are set to 00.
20 • Bits b2 and b1 indicate the logical channel number from 0 to 3.

21 For commands addressed to the supplementary logical channel 4 through 19:

- 22 • Bit b8 is set to 0 for the SELECT command and to 1 for all other commands defined in
23 this specification.
24 • Bit b7 is set to 1.
25 • When a secure channel is used, bit b6 is set to 1; otherwise it is set to 0.
26 • Bit b5 is set to 0.
27 • Bits b4 to b1 indicate the logical channel number from 4 through 19 (i.e., 0000 for logical
28 channel 4 through 1111 for logical channel 19)

Table 15-3 lists the four different ranges of class byte values to be supported for the different commands, depending on the options implemented by the Digital Key applet. The determination of whether a command shall use CLA1, CLA2, CLA3, or CLA4 range is defined in the command description provided in Section [15.3.2](#).

Table 15-3: Coding of the Different Ranges of Class Byte Values

Range of values of the class byte (CLA)	Over the contactless interface (hex)	Over the wired interface (hex)			
		DK applet implementation does not support Options A or B.	DK applet implementation supports Option A but does not support Option B.	DK applet implementation supports Option B but does not support Option A.	DK applet implementation supports Options A and B.
CLA1	00	00	00	00 – 03 or 40 – 4F	00 – 03 or 40 – 4F
CLA2	N/A	80	80 or 84	80 – 83 or C0 – CF	80 – 87 or C0 – CF or E0 – EF
CLA3	80	80	80 or 84	80 – 83 or C0 – CF	80 – 83 or C0 – CF
CLA4	84	84	84	84 – 87 or E0 – EF	84 – 87 or E0 – EF

15.3.1.8 INSTALL for INSTALL parameters

This command defined in GlobalPlatform specification [20] creates a Digital Key applet instance and allocates the needed memory.

The following values are provided in tag C9_h during the INSTALL for INSTALL command. Refer to GlobalPlatform specification [20] for formatting of this command.

The INSTALL for INSTALL command might include additional parameters, depending on the Device OEM policy for managing the non-volatile memory available for the Digital Key applet (e.g., using the Non-volatile Memory Quota parameters defined in GlobalPlatform [\[20\]](#)).

Table 15-4: INSTALL for INSTALL Content of Tag C9

Tag	Length (bytes)	Description	Field is
A0 _h	1	install_param_endpoint_count, in range 1–255	mandatory
A1 _h	1	install_param_instance_CA_count, in range 1–255	mandatory
A2 _h	2	install_param_internal_buffer_size (MSB LSB), in range 1024–max (where, “max” is defined by the Device OEM)	mandatory
A3 _h	2	install_param_max_allocatable_private_mailbox_size (MSB LSB), in range 0–2048	mandatory

Tag	Length (bytes)	Description	Field is
A4 _h	2	install_param_max_allocatable_confidential_mailbox_size (MSB LSB), in range 0–2048	mandatory
A6 _h	variable	install_param_oid_external_ca	mandatory
A7 _h	variable	install_param_oid_instance_ca	mandatory
A8 _h	variable	install_param_oid_endpoint	mandatory
63 _h	1	applet_implementation_options	optional

1 **Install_param_endpoint_count**

2 The maximum number of endpoints that can be created on an instance of the applet.

3 **Install_param_instance_CA_count**

4 The maximum number of Instance CA that can be created on an instance of the applet.

5 **Install_param_internal_buffer_size**

6 The size in bytes of the internal buffer described in Section [15.3.2.13](#). This value is set by the
7 Device OEM according to Device OEM service policy, which is out of scope of this
8 specification.

9 **Install_param_max_allocatable_private_mailbox_size**

10 The maximum size for private mailbox per endpoint. This value is set by the Device OEM
11 according to its service policy, which is out of scope of this specification.

12 **Install_param_max_allocatable_confidential_mailbox_size**

13 The maximum size for confidential mailbox per endpoint. This value is set by the Device OEM
14 according to its service policy, which is out of scope of this specification.

15 **Applet_implementation_options**

16 The applet_implementation_options is encoded as defined in Table 15-5.

17 If this field is not present, the applet implementation model is the framework-centric applet
18 model, and the SE root of trust certificate chain model is Variant 1, as depicted in [Figure 16-1](#),
19 and the MANAGE UA command is not implemented or disabled.

20 *Note:* Tag A6_h, A7_h and A8_h have fixed values as defined in Appendix [B.2](#), however they are
21 included here to allow for potential evolutions in the future.

22

Table 15-5: Applet Implementation Options

b8	b7	b6	b5	b4	b3	b2	b1	Description
-	-	-	-	-	-	-	0	applet_implementation_model is framework-centric applet model
-	-	-	-	-	-	-	1	applet_implementation_model is SE-centric applet model
-	-	-	-	-	-	0	-	SE Root of Trust certificate chain models is Variant 1
-	-	-	-	-	-	1	-	SE Root of Trust certificate chain models is Variant 2
-	-	-	-	-	0	-	-	MANAGE UA command is not available
-	-	-	-	-	1	-	-	MANAGE UA command is available

b8	b7	b6	b5	b4	b3	b2	b1	Description
0	0	0	0	0	-	-	-	RFU (0)

1

Listing 15-1: INSTALL for INSTALL Processing

```

1 input : install_param *
2 output : none
3 begin
4     check install_parameters
5     atomic start
6         allocate cod.internal_buffer using the size defined by install_param_internal_buffer_size
7         If not enough memory
8             return 6A84h
9         reserve space using the number of endpoints defined by
10            install_param_endpoint_count,
11            install_param_instance_CA_count
12        create dummy_endpoint
13            generate key pair endpoint_PK, endpoint_SK according to Listing 15-40
14            nvmwrite nvm. Dummy_endpoint.SK ← endpoint_SK
15            nvmwrite nvm. Dummy_endpoint . Kpersistent , fill with random
16            nvmwrite nvm. Dummy_endpoint . kpersistent_established ← true
17            nvmwrite nvm.dummy_endpoint.terminated ← false
18            nvmwrite nvm.dummy_endpoint.option_group_1 ← 0Fh (b4–b7 are set to zero)
19            nvmwrite nvm.dummy_endpoint.option_group_2 ← 00h (b0,b1,b7 are set to zero)
20            nvmwrite nvm.supported_protocol_versions_tlv ← TLV as per Table 15-12
21        atomic commit
22    end

```

2 The following values define the Protocol Data Type A (Card Emulation Mode) tag (86_h) and the
3 Protocol Data Type B (Card Emulation Mode) tag (87_h), which are provided in tag A0_h (Contact-
4 less Protocol Parameters tag) of tag EF_h (System Specific Parameters tag) during the INSTALL
5 for INSTALL command. Refer to GlobalPlatform Card Specification Amendment C [\[20\]](#) for
6 formatting of these parameters.

7 *Table 15-6: Values for Protocol Data Type A (tag '86')*

Tag	Length (bytes)	Value Description	Value in Protocol Parameter Data (Tag 'A0')	Value in Protocol Parameter Mandatory Mask (Tag 'A1')
80 _h	1	Unique Identifier LV structure	00 _h	00 _h
81 _h	1	SAK	20 _h	24 _h
82 _h	2	ATQA	0400 _h	39F0 _h
83 _h	1	ATS LV structure	00 _h	00 _h
84 _h	1	FWI, SFGI	72 _h	FF _h
85 _h	1	CID Support	00 _h	00 _h
86 _h	3	DATA_RATE_MAX	000000 _h	FFFF00 _h

8 For Type A SFGI value, coexistence issues with legacy NFC services may appear. In such cases,
9 increasing SFGI value could be allowed, with impact only on transaction performance. In any
10 case, this value shall not exceed 8.

11 *Note:* LSB byte of the two-byte ATQA value is transmitted first.

1

Table 15-7: Values for Protocol Data Type B (tag 87_h)

Tag	Length (bytes)	Value Description	Value in Protocol Parameter Data (Tag 'A0')	Value in Protocol Parameter Mandatory Mask (Tag 'A1')
80 _h	1	PUPI LV structure LV structure	00 _h	00 _h
81 _h	1	AFI	00 _h	00 _h
82 _h	4	ATQB	00000071 _h	000000F1 _h
83 _h	1	Higher Layer Response in response to ATTRIB	00 _h	00 _h
84 _h	3	Maximum data rate	000000 _h	FFFF00 _h

2 NOTE: For some areas, certain contactless services require fixed parameter values. In such
3 cases, corresponding mask values as defined in [Table 15-6](#) and [Table 15-7](#) may be adapted to
4 allow availability of these services concurrently with the DK service.

5 *15.3.1.9 Notification of the Digital Key framework*

6 **Notification for APDU commands processed over the contactless interface**

7 This notification mechanism may be used in case the APDU command is processed over the
8 contactless interface. The DK applet uses the Connectivity Service interface defined in ETSI TS
9 102 705 [\[25\]](#) to request the NFC Controller to create a Host Controller Interface (HCI) event
10 EVT_TRANSACTION (as defined in ETSI TS 102 622 [\[26\]](#)) as follows:

- 11 • The field AID (Tag 81_h) contains the AID of the DK applet.
- 12 • The field PARAMETERS(Tag 82_h) contains a list of notifications. The coding of each
13 notification is as defined in [Table 15-9](#). These notifications are appended to the list in the
14 sequential order they are generated.

15 Using this notification mechanism is referred to as Option C in this specification.

16 **Notification for APDU commands processed over the wired interface**

17 This notification mechanism may be used in case the APDU command is processed over the
18 wired interface. The DK framework may issue a GET NOTIFICATION command described in
19 Section [15.3.2.28](#) to retrieve the notification(s) potentially generated by the DK applet during the
20 processing of the previous APDU command.

21 Using this notification mechanism is known as Option D in this specification.

22

23 *Table 15-8: Status Word - Command successfully executed*

SW as defined in the listing in Section 15.3.2	SW sent by the Digital Key applet to the Digital Key framework	Behavior of Digital Key framework
SW 9000 _h	SW 9000 _h if no notification is pending	Forward SW 9000 _h to the vehicle
	SW 9100 _h if a notification is pending	Forward SW 9000 _h to the vehicle

SW as defined in the listing in Section 15.3.2	SW sent by the Digital Key applet to the Digital Key framework	Behavior of Digital Key framework
		Issue a GET NOTIFICATION command to retrieve the notification(s)
All other SWs	SW as defined in the listing (notification pending or not)	Forward the received SW to the vehicle Issue a GET NOTIFICATION command to retrieve potentially generated notification(s)

1

2 Coding of the notifications

3 *Table 15-9: Coding of the notification*

Tag	Length (bytes)	Description	Field is
7F60 _h	variable	Notification	mandatory
81 _h	1	Notification code as defined in Table 15-11	mandatory
A0 _h	variable	Notification context as defined in Table 15-10	mandatory
8E _h	variable	Additional data as defined in Table 15-11	conditional

4 The notification context is defined in [Table 15-10](#).

5 The different values of the notification code corresponding to the different notification notify_* are defined in [Table 15-11](#).

6 The presence and content of the additional data depends on the notification code (see [Table 15-11](#)).

9 *Table 15-10: Notification Context*

Tag	Length (bytes)	Description	Field is
A0 _h	variable	Notification context	mandatory
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	conditional
4D _h	8	vehicle_identifier	conditional

10 The presence of the key_identifier and of the vehicle_identifier depends on the notification code (see [Table 15-11](#)).

12 *Table 15-11: Value and Presence of the Different Fields in the HCI Notification Event*

Notification	Code	Field key_identifier is	Field vehicle_identifier is	Field additional data is
notify_start_of_transaction	00 _h	NA	NA	NA
notify_endpoint_not_found_fast	01 _h	NA	mandatory	Length: 2

Notification	Code	Field key_identifier is	Field vehicle_identifier is	Field additional_data is
				Value: P1, P2 of AUTH0 command
notify_endpoint_not_found_standard	02 _h	NA	mandatory	Length: 2 Value: P1, P2 of AUTH0 command
notify_user_authentication_not_performed	03 _h	NA	mandatory	Length: 2 Value: P1, P2 of AUTH0 command
notify_standard with nvm.endpoint.identifier	04 _h	mandatory	mandatory	Length: 2 Value: P1, P2 of AUTH0 command
notify_fast_kpersistent_established with nvm.endpoint.identifier	05 _h	mandatory	mandatory	Length: 2 Value: P1, P2 of AUTH0 command
notify_fast_kpersistent_not_established with nvm.endpoint.identifier	06 _h	mandatory	mandatory	Length: 2 Value: P1, P2 of AUTH0 command
notify_vehicle_authentication_failed	07 _h	optional	optional	NA
notify_vehicle_authentication_success	08 _h	optional	optional	NA
notify_endpoint_not_found_presence	09 _h	NA	mandatory	NA
notify_presence with nvm.endpoint.identifier	0A _h	mandatory	mandatory	NA
notify_mailbox_written	0C _h	optional	optional	NA
notify_deselect with reason	0D _h	NA	NA	Length: variable Value: Reason
notify_end_of_transaction with P1, P2 during a fast or standard transaction	0E _h	optional	optional	Length: 4 Value: P1, P2 of CONTROL FLOW command, P1, P2 of AUTH0 command
notify_end_of_transaction with P1, P2 outside a fast or standard transaction	0E _h	optional	optional	Length: 4

Notification	Code	Field key_identifier is	Field vehicle_identifier is	Field additional data is
				Value: P1, P2 of CONTROL FLOW command, FFFF _h
notify_application_specific with P1, P2 during a fast or standard transaction	0F _h	optional	optional	Length: 4 Value: P1, P2 of CONTROL FLOW command, P1, P2 of AUTH0 command
notify_application_specific with P1, P2 outside a fast or standard transaction	0F _h	optional	optional	Length: 4 Value: P1, P2 of CONTROL FLOW command, FFFF _h
notify_message_in_exchange	10 _h	optional	optional	Length: variable Value: Data present in the value part of tag 8E _h of the EXCHANGE command
notify_URSK_created	11 _h	optional	NA	NA
RFU	12 _h -FF _h	-	-	-

1 15.3.2 *Commands*

2 15.3.2.1 *SELECT command*

3 The AID of the Digital Key applet shall be A0000008094343444B417631_h.

4 This command selects the Digital Key applet instance to be used for the transaction. The instance AID parameter is defined during applet installation (see the INSTALL for INSTALL command in [20]).

7 The SE should respond with the status word 6985_h if the Digital Key applet is temporarily unavailable.

9 **command: CLA1 A4 04 00 Lc [instance AID] 00**

10 **response: [Table 15-12] 90 00**

11 The CLA1 is as defined in [Table 15-3](#).

1

Table 15-12: SELECT Response Fields

Tag	Length (bytes)	Description	Field is	Domain Version
5Ch	variable	A list of supported Digital Key applet transaction (V-D-TX) protocol versions (ver.high ver.low) ordered from highest to lowest. Each version number is concatenated and encoded on 2 bytes in big-endian notation.	mandatory	V-D-TX

2

Listing 15-2: SELECT Processing

```

1 input: none
2 output: supported_protocol_versions_tlv
3 begin
4   cod.transaction_state ← select_done
5   cod.atomic_session ← stopped
6   if contactless interface
7     sent to digital key framework notify_start_of_transaction
7   return nvm .supported_protocol_versions_tlv
8 end

```

3 *15.3.2.2 CREATE CA command*

4 This command creates an onboard signing authority called Instance CA used for endpoint
5 issuance. The generation and provisioning of this entity is out of scope of this specification. The
6 certificate containing the Instance CA public key shall comply with Listing 15-16. The INS
7 value for CREATE CA command shall be 38h.

8 *15.3.2.3 DELETE CA command*

9 This command deletes an Instance CA. The method of deletion of this entity is out of scope of
10 this specification. The INS value for DELETE CA command shall be 3Ah.

11 *15.3.2.4 CREATE ENDPOINT command*

12 The command creates a communication endpoint and provides the corresponding endpoint
13 certificate. A communication endpoint distributes signatures and mailbox content over a secure
14 channel to authenticated parties only.

15 The endpoint configuration fields described in [Table 15-13](#) may be provided either in the
16 command payload or in the internal buffer using the WRITE BUFFER command described in
17 Section [15.3.2.14](#). The endpoint creation certificate described in [Listing 15-5](#) is always provided
18 in the internal buffer and is accessible using the READ BUFFER command described in Section
19 [15.3.2.13](#).

20 **command: CLA2 70 00 00 Lc ([Table 15-13] [Table 15-16] 00)**

21 **response: [response_length] 9000**

22 The CLA2 is as defined in [Table 15-3](#).

23

Table 15-13: Endpoint Configuration

Tag	Length (bytes)	Description	Field is	Domain Version
7F27h	variable	Endpoint configuration	mandatory	V-OD-FW (Owner Pairing) OD-FD-KS (Key Sharing)
4Dh	8	vehicle_identifier	mandatory	V-OD-FW

Tag	Length (bytes)	Description	Field is	Domain Version
5F20 _h	1–30	endpoint_identifier, DER encoded PrintableString as per RFC 5280 [3] (without PrintableString tag and length)	mandatory	V-OD-FW
42 _h	1–30	instance_CA_identifier, DER encoded PrintableString as per RFC 5280 [3] (without PrintableString tag and length)	mandatory	V-OD-FW
46 _h	1	option_group_1 (on bits 0–7): bit0 : enable(1)/disable(0) Standard transaction allowed on contactless interface bit1 : enable(1)/disable(0) Fast transaction allowed on contactless interface bit2 : enable(1)/disable(0) Standard transaction allowed on wired interface bit3 : enable(1)/disable(0) Fast transaction allowed on wired interface bit4 : enable(1)/disable(0) AUTHORIZE ENDPOINT command allowed bit5 : enable(1)/disable(0) Authorizing endpoints which have the AUTHORIZE ENDPOINT command allowed bit6 : enable(1)/disable(0) EXCHANGE command allowed on wired interface bit7 : enable(1)/disable(0) EXCHANGE command allowed after AUTH0	mandatory	V-OD-FW
47 _h	1	option_group_2 (on bits 0–7): bit0 : enable(1)/disable(0) SIGN command allowed bit1 : enable(1)/disable(0) Exported data from confidential mailbox are replaced with random values bit2 : RFU (shall be set to 0) bit3 : RFU (shall be set to 0) bit4 : RFU (shall be set to 0) bit5 : RFU (shall be set to 0) bit6 : RFU (shall be set to 0) bit7 : enable(1)/disable(0) SET CONFIDENTIAL DATA command allowed	mandatory	V-OD-FW
5C _h	2	protocol_version	mandatory	V-OD-FW
5B _h	65	vehicle_PK prepended by 04 _h	mandatory	V-OD-FW
51 _h	15 or 13	not_before, DER encoded GeneralizedTime (15 bytes in length) or UTCTime (13 bytes in length) as per RFC 5280 [2]	mandatory	V-OD-FW OD-FD-KS

Tag	Length (bytes)	Description	Field is	Domain Version
52 _h	15 or 13	not_after, DER encoded GeneralizedTime (15 bytes in length) or UTCTime (13 bytes in length) as per RFC 5280 [3]	mandatory	V-OD-FW OD-FD-KS
49 _h	variable	authorized_PK[], list of up to 5 public keys prepended by 04 _h of 65 bytes length	optional	V-OD-FW
4A _h	2	confidential_mailbox_size	optional	V-OD-FW
4B _h	2	private_mailbox_size	optional	V-OD-FW
4E _h	1–8	key_slot	optional	D-VS
57 _h	4	counter_limit, values in range 00000001 _h –FFFFFFFFFF _h	deprecated*	N/A

1 * This field has been deprecated and shall no longer be used.

2 **vehicle_identifier:** Field provided by the endpoint creator; this field is used by the applet for fast endpoint lookup during transaction.

4 **endpoint_identifier:** Arbitrary field provided by the endpoint creator, describing the subject of the certificate. The endpoint_identifier field typically identifies the entity associated with the public key present in the certificate.

7 **instance_CA_identifier:** This field allows for selection of the instance CA used by the SE to sign the endpoint creation certificate. This value is present in the field “subject CommonName” of the corresponding instance certificate.

10 **option_group_1 and option_group_2:** For Owner Pairing (Owner Pairing Phase 2), the vehicle shall set these two fields according to [Table 15-14](#).

12 *Table 15-14: Setting of Tag 46_h and Tag 47_h by vehicle for various wireless capability combinations.*

Tag	Value	Description
Vehicle supporting WCC1		
46 _h	bit0 set to 1	Standard transaction allowed on contactless interface
	bit1 set to 1	Fast transaction allowed on contactless interface
	bit2 set to 0	Standard transaction not allowed on wired interface
	bit3 set to 0	Fast transaction not allowed on wired interface
	bit4 set to 1	AUTHORIZE ENDPOINT command allowed
	bit5 set to 0	Authorizing endpoints which have the AUTHORIZE ENDPOINT command not allowed
	bit6 set to 0	EXCHANGE command not allowed on wired interface
	bit7 set to 0/1	EXCHANGE command allowed (1) or not allowed (0) after AUTH0
Vehicle supporting WCC2 or WCC3		
46 _h	bit0 set to 1	Standard transaction allowed on contactless interface
	bit1 set to 1	Fast transaction allowed on contactless interface
	bit2 set to 1	Standard transaction allowed on wired interface

Tag	Value	Description
	bit3 set to 0	Fast transaction not allowed on wired interface
	bit4 set to 1	AUTHORIZE ENDPOINT command allowed
	bit5 set to 0	Authorizing endpoints which have the AUTHORIZE ENDPOINT command not allowed
	bit6 set to 1	EXCHANGE command allowed on wired interface
	bit7 set to 0/1	EXCHANGE command allowed (1) or not allowed (0) after AUTH0
47 _h	bit0 set to 1	SIGN command allowed
	bit1 set to 1	Exported data from confidential mailbox are replaced with random values
	bit2 set to 0	RFU
	bit3 set to 0	RFU
	bit4 set to 0	RFU
	bit5 set to 0	RFU
	bit6 set to 0	RFU
	bit7 set to 0/1	SET CONFIDENTIAL DATA command not allowed (0) if immobilizer tokens are not used or allowed (1) if immobilizer tokens are used

- 1
2 The bit7 of tag 46_h, shall not be set to 1 if sensitive data such as immobilizer tokens are exchanged using the EXCHANGE command.
3
4 For friend sharing (Key Sharing flow step 1 and 2), the owner device shall set these two fields which is within Key Creation Request according to Table 15-15.
5
6

Table 15-15: Setting of Tag 46_h and Tag 47_h by the owner device for the key sharing.

Tag	Value	Description
46 _h	bit0 to bit3	As defined in the endpoint configuration (owner device)
	bit4 set to 0	AUTHORIZE ENDPOINT not command allowed
	bit5 to bit7	As defined in the endpoint configuration (owner device)
47 _h	bit0 to bit7	As defined in the endpoint configuration (owner device)

- 7
8 **protocol_version:** Digital Key applet protocol transaction (V-D-TX) version assigned at the time of the endpoint creation. This field is informative and shall not be used by the applet to enforce usage of a specific protocol version. Thus, this shall not prevent the endpoint to be used with other versions of the Digital Key applet protocol version that may be supported by the vehicle and the Digital Key applet later.
9
10
11
12
13 **vehicle_PK:** The public key of the vehicle.
14 **Not_before:** The issued certificate start of validity date. The validity date is not required to be verified by the applet.
15
16 **Not_after:** The issued certificate end of validity date. The validity date is not required to be verified by the applet.
17

1 **Authorized_PK[]**: The public keys used by the device to verify other SE's endpoint creation
2 certificates.

3 **Confidential_mailbox_size**: If field not present, or if present with confidential_mailbox_size
4 equal to 0000_h, confidential mailbox is not available.

5 **Private_mailbox_size**: If field not present, or if present with private_mailbox_size equal to
6 0000_h, private mailbox is not available.

7 **Key_slot**: A field in the endpoint configuration that is populated in the friend endpoint
8 configuration during step 1 of the key sharing (see Section [11.8.1](#)) if slot identifiers are retrieved
9 from the vehicle via the owner device. The value of this field is set to one of the slot identifiers
10 from the SlotIdentLst field. The key_slot can be manually set when a fast search among a large
11 number of public keys is required on the vehicle side. The fast search can be facilitated by
12 attributing key slot numbers instead of relying on the automatic hash-based naming during
13 endpoint creation. If key_slot field is not provided, an automatic slot number is generated as
14 described in [Listing 15-6](#), since the key_slot is mandatory in the AUTH1-Response. In case of
15 slot identifiers provided online, the owner device during key sharing, assigns an automatically
16 generated random slot identifier. The slot identifier is used to derive the Kble_oob, see Sections
17 19.5.1 and 19.5.8.

18 *Note*: A device generated key_slot does not safeguard against attestation replay attacks during
19 key sharing.

20 This field does not need to be unique among endpoints created on an instance.

21 This field can be updated after key creation with the SETUP_ENDPOINT command. The value
22 “initial_key_slot” in the endpoint certificate will always remain the one initially assigned during
23 endpoint creation.

24 **Counter_limit**: This field is deprecated and shall not be used.

25 If the applet_implementation_model field is included (i.e., SE-centric applet model) in the
26 INSTALL for INSTALL command, the following additional tags shall appear after the tag
27 7F27_h.

28 *Table 15-16: Endpoint verification information*

Tag	Length (bytes)	Description	Field is
7F28 _h	variable	Vehicle OEM CA Certificate (signed by Device OEM)	conditional
7F4C _h	variable	DER-encoded X.509 Intermediate Certificate	optional
7F4B _h	variable	DER-encoded X.509 Vehicle Public Key Certificate	conditional

29 **Vehicle OEM CA Certificate (signed by Device OEM)**: this field is provided by the
30 framework and is verified by the SE. The SE also uses this data to verify the Vehicle Public Key
31 Certificate.

32 It is assumed that Device OEM CA root certificate is stored in the SE for this operation.

33 **Vehicle Public Key Certificate**: This field is provided by the vehicle during owner pairing. This
34 field is not present during Digital Key sharing. The SE shall verify the Vehicle Public Key
35 Certificate if this field is present.

36 *Listing 15-3: Endpoint Certificate Extension Schema*

1 Schema DEFINITIONS EXPLICIT TAGS

```
2 BEGIN
3 EndpointCertificateExtensionSchema ::= SEQUENCE
4 {
5 extension_version      INTEGER (1..255),
6 vehicle_identifier     OCTET STRING (SIZE (8)),
7 option_group_1          OCTET STRING (SIZE (1)),
8 option_group_2          OCTET STRING (SIZE (1)),
9 protocol_version        OCTET STRING (SIZE (2)),
10 vehicle_PK              PublicKey,
11 initial_key_slot        OCTET STRING (SIZE (1..8)),
12 authorized_PK_list      SEQUENCE(SIZE(1..5)) OF PublicKey OPTIONAL,
13 confidential_mailbox_size [0] INTEGER (1..65535) OPTIONAL,
14 private_mailbox_size    [1] INTEGER (1..65535) OPTIONAL,
15 }
16
17 PublicKey ::= SEQUENCE
18 {
19 algorithm   AlgorithmIdentifier,
20 public_key  BIT STRING
21 }
22
23 AlgorithmIdentifier ::= SEQUENCE
24 {
25 algorithm   OBJECT IDENTIFIER,
26 parameters  ANY DEFINED BY algorithm OPTIONAL
27 }
28 END
```

1 *Listing 15-4: Endpoint Certificate Extension Data*

```
1 endpoint-cert-extension-data EndpointCertificateExtensionSchema ::=
2 {
3 extension_version      1, --value indicating the extension version, this value is meant to be incremented if the extension fields
4 are changed in
5 next applet versions.
6 vehicle_identifier     '...'H, --value as per endpoint configuration
7 option_group_1          '...'H, --value as per endpoint configuration
8 option_group_2          '...'H, --value as per endpoint configuration
9 protocol_version        '...'H, --value as per endpoint configuration
10 vehicle_PK              {
11 algorithm
12 {
13 algorithm {1 2 840 10045 3 1 7} --OID for prime256v1(ANSI X9.62 named elliptic curve)
14 },
15 public_key '04...'H --vehicle_PK pre-pended with 04h
16 },
17 initial_key_slot        '...'H, --value as per endpoint configuration
18 authorized_PK_list      {
19 {
20 {
21 algorithm
22 {
23 algorithm {1 2 840 10045 3 1 7} --OID for prime256v1(ANSI X9.62 named elliptic curve)
24 },
25 public_key '04...'H --authorized_PK
26 },
27 ... --other authorized_PK
28 },
```

```
29 confidential_mailbox_size ..., --value as per endpoint configuration
30 private_mailbox_size     ..., --value as per endpoint configuration
31 }
```

1 *Listing 15-5: Endpoint Certificate Data*

```
1 endpoint-cert-data Certificate ::= 
2 {
3   tbsCertificate
4   {
5     version v3, --shall be v3
6     serialNumber ..., --a random integer chosen by the certificate issuer
7     signature
8     {
9       algorithm {1 2 840 10045 4 3 2} --OID for ecdsaWithSHA256 (ANSI X9.62 ECDSA algorithm with SHA256)--
10      },
11     issuer rdnSequence:
12     {
13       {
14         {
15           type {2 5 4 3}, --OID for CommonName
16           value "..." --the instance_CA_identifier as per endpoint configuration
17             -- (PrintableString or UTF8String recommended)
18         }
19       }
20     },
21     validity
22     {
23       notBefore Time: "...", --value not_before as per endpoint configuration
24       notAfter Time: "..." --value not_after as per endpoint configuration
25     },
26     subject rdnSequence:
27     {
28       {
29         {
30           type {2 5 4 3}, --OID for CommonName
31           value "..." --the endpoint_identifier as per endpoint configuration, shall use PrintableString or UTF8String format
32         }
33       }
34     }
35   },
36   subjectPublicKeyInfo
37   {
38     algorithm
39     {
40       algorithm {1 2 840 10045 2 1}, --OID for ecPublicKey (ANSI X9.62 public key type)
41       parameters {1 2 840 10045 3 1 7} --OID for prime256v1 (ANSI X9.62 named elliptic curve)
42     },
43     subjectPublicKey '04...'H --the public key pre-pended with 04 to indicate uncompressed format
44   },
45   extensions
46   {
47     {
48       extnID   {install_param_oid_endpoint}, --OID for Endpoint certificate
49       critical TRUE,
50       extnValue  '...'H --DER encoding for endpoint-cert-extension-data as per Listing 15-4
51     },
52     {
53       extnID   {2 5 29 15}, --OID for KeyUsage standard extension
```

```

54     critical TRUE,
55     extnValue '03020780'H --DER encoding for KeyUsage, digitalSignature
56 },
57 {
58     extnID {2 5 29 19}, --OID for BasicConstraints standard extension
59     critical TRUE,
60     extnValue '3000'H --DER encoding for cA=FALSE
61 },
62 {
63     extnID {2 5 29 35}, --OID for AuthorityKeyIdentifier standard extension
64     critical FALSE,
65     extnValue '...'"H -- DER encoding of an AuthorityKeyIdentifier sequence as defined in [3], containing only
66             -- a KeyIdentifier element. The KeyIdentifier is an OCTET STRING containing the
67             -- 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey
68             -- from the issuer certificate (excluding the tag, length, and number of unused bits)
69 },
70 ...
71 }
72 },
73 signatureAlgorithm
74 {
75     algorithm {1 2 840 10045 4 3 2}
76 },
77 signatureValue '...'"H --the certificate signature computed as per RFC 5280 [3].
78 }
```

1 Listing 15-6: CREATE ENDPOINT Processing

```

1 input: Table 15-13, Table 15-16
2 output: response_length, certificate
3 begin
4     if no payload present in command
5         if Table 15-13 data fields not present in cod.internal_buffer
6             return 6400h
7         if any mandatory field is not present
8             return 6400h
9         if endpoint_identifier already exists in any of the nvm.endpoint[].identifier
10            return 6400h
11         if protocol_version from input table is not supported
12            return 6400h
13         if instance CA referenced by instance_CA_identifier field does not exist
14            return 6400h
15         if vehicle_identifier is already used by any of the existing of the nvm.endpoint[]
16            return 6400h
17         if SE-centric applet model
18             if Table 15-16 data fields not present in cod.internal_buffer
19                 return 6400h
20             if Table 15-16 data fields are not verified
21                 return 6400h
22         generate key pair endpoint_PK, endpoint_SK according to Listing 15-40
23         if key_slot field not provided
24             generate key_slot as per Listing 15-41 using subjectPublicKey of the endpoint to be created as input (excluding the tag,
length, and number of unused bits)
25             generate certificate using Listing 15-5 as per RFC 5280 [3]
26             atomic start
27                 nvmwrite nvm.endpoint.<...> configuration as per Table 15-13 and certificate parts (including certificate signature) needed to
re-generate the certificate when retrieved with VIEW command
28                 nvmwrite nvm.endpoint.identifier ← endpoint_identifier
29                 nvmwrite nvm.endpoint.key_slot ← key_slot
```

```

30 nvmwrite nvm.endpoint.initial_key_slot ← key_slot
31 nvmwrite nvm.endpoint.PK ← endpoint_PK
32 nvmwrite nvm.endpoint.SK ← endpoint_SK
33 nvmwrite nvm.endpoint.Kpersistent, fill with randoms
34 nvmwrite nvm.endpoint.kpersistent_established ← false
35 nvmwrite nvm.endpoint.terminated ← false
36 nvmwrite nvm.endpoint.transaction_code_list_cless ← []
37 nvmwrite nvm.endpoint.transaction_code_list_wired ← []
38 atomic commit
39
40 store certificate in cod.internal_buffer
41 response_length ← length written in cod.internal_buffer
return response_length (2 bytes big-endian)
end

```

1 15.3.2.5 TERMINATE ENDPOINT command

2 This command sets the endpoint in terminated state and returns a termination attestation. If the
3 endpoint is already in terminated state, the previously computed attestation is returned. The
4 terminated state of an endpoint cannot be reverted; the only possible action is to delete the
5 endpoint.

6 The following command formats are allowed:

- 7 • If no secure channel is to be established between the Digital Key framework and the
8 Digital Key applet, then either command format 1 or command format 2 may be used and
9 implemented by the applet.
- 10 • If a secure channel is to be established between the Digital Key framework and the
11 Digital Key applet (option A), command format 2 shall be used and shall be implemented
12 by the applet.

13 Command format 1:

14 **command: CLA2 72 00 00 Lc [Table 15-17] 00**

15 **response: [Table 15-19] 9000**

16 The CLA2 is as defined in Table 15-3.

17 Command format 2:

18 **command: CLA2 72 00 00 Lc [Table 15-17] 00**

19 **response: [response_length] 9000**

20 The CLA2 is as defined in Table 15-3.

21 *Table 15-17: Endpoint Termination Request*

Tag	Length (bytes)	Description	Field is	Domain Version
7F2E _h	variable	Endpoint Termination Request	mandatory	N/A
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW
91 _h	16	termination_requester_nonce, a random value chosen by the termination requester	mandatory	N/A

Tag	Length (bytes)	Description	Field is	Domain Version
58 _h	1–40	arbitrary_data, arbitrary data field chosen by the termination requester	optional	N/A

1 *Table 15-18: Endpoint Termination Attestation Data Fields*

Tag	Length (bytes)	Description	Field is	Domain Version
41 _h	1	version = 01 _h , version of Endpoint Termination Attestation	mandatory	Informative only. N/A
92 _h	8	random	mandatory	D-VS
5F20 _h	1–30	endpoint_identifier, identifier of the terminated endpoint	mandatory	V-OD-FW
5F49 _h	65	the public key of the terminated endpoint	mandatory	V-OD-FW
91 _h	16	termination_requester_nonce	mandatory	D-VS
58 _h	1–40	arbitrary_data	optional	D-VS
93 _h	4	usage = D6854CB9 _h	mandatory	D-VS

2 *Table 15-19: Endpoint Termination Attestation*

Tag	Length (bytes)	Description	Field is	Domain Version
7F29 _h	variable	Endpoint Termination Attestation	mandatory	D-VS
Content of Table 15-18			mandatory	
9E _h	64	signature with the private key of the terminated endpoint over fields from Table 15-18	mandatory	D-VS

3 *Listing 15-7: TERMINATE ENDPOINT Processing for Command Format 1*

```

1 input: key_identifier
2 output: termination_attestation
3 begin
4   if key_identifier does not match with any of the existing endpoints
5     return 6400h
6   if nvm.endpoint.terminated = true
7     return nvm.endpoint.termination_attestation
8   if UWB present
9     delete URSK associated to the key_identifier
10    nvm.endpoint.terminated ← true
11    generate 8 bytes random as per Listing 15-39
12    generate signature over Table 15-18 using key selected from nvm.endpoint.SK according to Listing 15-42
13    nvmwrite nvm.endpoint.termination_attestation ← data structure as per Table 15-19
14    return nvm.endpoint.termination_attestation
15 end

```

4 *Listing 15-8: TERMINATE ENDPOINT Processing for Command Format 2*

```

1 input: key_identifier
2 output: response length, termination_attestation
3 Begin
4   if key_identifier does not match with any of the existing endpoints
5     return 6400h
6   if nvm.endpoint.terminated = true
7     store nvm.endpoint.termination_attestation in cod.internal_buffer

```

```

8   response_length ← length written in cod.internal_buffer
9   return response_length (2 bytes big-endian)
10  if UWB present
11    delete URSK associated to the key_identifier
12  nvm.endpoint.terminated ← true
13  generate 8 bytes random as per Listing 15-39
14  generate signature over Table 15-18 using key selected from nvm.endpoint.SK according to Listing 15-42
15  nvmwrite nvm.endpoint.termination_attestation ← data structure as per Table 15-19
16  store nvm.endpoint.termination_attestation in cod.internal_buffer
17  response_length ← length written in cod.internal_buffer
18  return response_length (2 bytes big-endian)
19  End

```

1 15.3.2.6 DELETE ENDPOINT command

2 This command will delete an endpoint and release the associated memory.

3 command: CLA2 74 00 00 Lc [Table 15-20]

4 response: 9000

5 The CLA2 is as defined in Table 15-3.

Table 15-20: Deletion Request

Tag	Length (bytes)	Description	Field is	Domain Version
50h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW

Listing 15-9: DELETE ENDPOINT Processing

```
1 input key_identifier
2 output: none
3 begin
4   if key_identifier does not match with any of the existing endpoints
5     return 6400h
6   if UWB present
7     delete URSK associated to the key_identifier
8   atomic start
9     nvmdelete nvm.endpoint
10  atomic commit
11 end
```

8 15.3.2.7 AUTHORIZE ENDPOINT command

9 This command signs an endpoint public key (issued by a receiver device) and an arbitrary
10 additional data field using a local endpoint private key (sender device). Optionally, the command
11 can also extract and encrypt a portion of the confidential mailbox to be inserted in the authorized
12 recipient endpoint.

13 Input and output buffers are transferred using the internal buffer commands READ BUFFER and
14 WRITE BUFFER to avoid APDU length limitations.

15 The supported verification chains are described in Section 16.6. Chain verification starts from
16 one of the authorized public keys stored in the endpoint and continues from top to bottom with
17 the items in Table 15-22.

18 This command requires user authentication to be performed on device.

1 Note: The protocol_version value present in the receiver's endpoint certificate shall not be
2 checked during processing of this command.

3 **command: CLA2 32 00 00 Lc [Table 15-21] 00**

4 **response: [response_length] 90 00**

5 *Table 15-21: AUTHORIZE ENDPOINT Command Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW
4A _h	4	offsetmsb offsetlsb lengthmsb lengthlsb, portion of the confidential mailbox to be exported	optional	V-OD-FW
58 _h	variable	arbitrary_data	optional	N/A

6 *Listing 15-10: External CA Certificate Extension Schema*

```
1 ExternalCACertificateExtensionSchema ::= SEQUENCE
2 {
3     extension_version     INTEGER (1..255)
4 }
```

7 *Listing 15-11: External CA Certificate Extension Data*

```
1 external-ca-cert-extension-data ExternalCACertificateExtensionSchema :=
2 {
3     extension_version 1      --value shall be 1
4 }
```

8 *Listing 15-12: External CA Certificate Basic Constraints Extension Data*

```
1 external-ca-cert-basic-constraints BasicConstraints :=
2 {
3     cA          TRUE,
4     pathLenConstraint ...    --value shall be 1 if an Instance CA is conditionally or always present in the chain
5                           --value shall be 0 if Instance CA is never present in the chain
6 }
```

9 *Listing 15-13: External CA Certificate Data*

```
1 external-ca-cert-data Certificate :=
2 {
3     tbsCertificate
4     {
5         version v3, --shall be v3
6         serialNumber ..., --a random integer chosen by the certificate issuer
7         signature
8         {
9             algorithm {1 2 840 10045 4 3 2} --OID for ecdsaWithSHA256 (ANSI X9.62 ECDSA algorithm with SHA256)
10            },
11         issuer rdnSequence:
12         {
13             {
14                 {
15                     type {2 5 4 3}, --OID for CommonName
16                     value "..." --shall match the subject of the issuer certificate
17                     --(shall use PrintableString or UTF8String format)*
18             }
19         }
20     }
21 }
```

```
18     }
19   }
20 },
21 validity
22 {
23   notBefore Time: "...", -- shall use UTCTime or GeneralizedTime as defined in [3]
24   notAfter Time: "..." – shall use UTCTime or GeneralizedTime as defined in [3]
25 },
26 subject rdnSequence:
27 {
28 {
29 {
30   type {2 5 4 3}, --OID for CommonName
31   value "..." --contains the subject of the certificate
32   --(shall use PrintableString or UTF8String format)*
33 }
34 }
35 },
36 subjectPublicKeyInfo
37 {
38 algorithm
39 {
40   algorithm {1 2 840 10045 2 1}, --OID for ecPublicKey (ANSI X9.62 public key type)
41   parameters {1 2 840 10045 3 1 7} --OID for prime256v1(ANSI X9.62 named elliptic curve)
42 },
43 subjectPublicKey '04...'H --the public key pre-pended with 04 to indicate uncompressed format
44 },
45 extensions
46 {
47 {
48   extnID {1.3.6.1.4.1.41577.5.2}, --OID for external CA Certificate (see Appendix B.2.2)
49   critical TRUE,
50   extnValue '...'H --DER encoding for ExternalCACertificateExtensionSchema extension as per Listing 15-11
51 },
52 {
53   extnID {2 5 29 15}, --KeyUsage standard extension
54   critical TRUE,
55   extnValue '03020204'H --DER encoding for KeyUsage, CertSign only
56 },
57 {
58   extnID {2 5 29 19}, --BasicConstraints standard extension
59   critical TRUE,
60   extnValue '...'H --DER encoding for BasicConstraints extension as per Listing 15-12
61 },
62 {
63   extnID {2 5 29 35}, --OID for AuthorityKeyIdentifier standard extension
64   critical FALSE,
65   extnValue '...'H – DER encoding of an AuthorityKeyIdentifier sequence as defined in [3], containing only
66   – a KeyIdentifier element. The KeyIdentifier is an OCTET STRING containing the
67   – 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey
68   – from the issuer certificate (excluding the tag, length, and number of unused bits)
69 },
70 {
71   extnID {2 5 29 14}, --OID for SubjectKeyIdentifier standard extension
72   critical FALSE,
73   extnValue '...'H --160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey
74   --(excluding the tag, length, and number of unused bits)
75 },
```

```
76  {
77    extnID   {...}, --Optional extensions added by the issuer, content not verified by the applet
78    critical  FALSE,
79    extnValue  '...'H --DER encoding of the extension
80  },
81  ...
82  },
83  },
84  signatureAlgorithm
85  {
86    algorithm {1 2 840 10045 4 3 2} --OID for ecdsaWithSHA256 (ANSI X9.62 ECDSA algorithm with SHA256)
87  },
88  signatureValue '...'H --the certificate signature by the issuer computed as per RFC 5280 [3]
89  --ECDSA signature
90 }
```

* The choice of format between PrintableString and UTF8String in the External CA certificate impacts the format of fields in Vehicle and Device OEM certificates as outlined in Section A.4.7 of Appendix A and Section B.2.5 of Appendix B

1 *Listing 15-14: Instance CA Certificate Extension Schema*

```
1 InstanceCACertificateExtensionSchema ::= SEQUENCE
2 {
3   extension_version      INTEGER (1..255),
4   applet_version          OCTET STRING (SIZE 4),
5   platform_information    OCTET STRING (SIZE (1..20)) OPTIONAL
6 }
```

2 *Listing 15-15: Instance CA Certificate Extension Data*

```
1 instance-ca-cert-extension-data InstanceCACertificateExtensionSchema ::= 
2 {
3   extension_version      1, --value shall be 1
4   applet_version          '...'H --proprietary applet version information
5   platform_information    '...'H --optional proprietary platform information
6   --field typically set by Device OEM Server to
7   --identify the Secure Element platform
8 }
```

3 **Note:** applet_version and platform_information are immutable values assigned at the time of the
4 Instance CA certificate is issued. It might not reflect the current applet_version or
5 platform_information (e.g., in case of applet or Secure Element platform upgrade).

6

7 *Listing 15-16: Instance CA Certificate Data*

```
1 instance-ca-cert-data Certificate :=
2 {
3   tbsCertificate
4   {
5     version v3, --shall be v3
6     serialNumber ..., --a random integer chosen by the certificate issuer
7     signature
8     {
9       algorithm {1 2 840 10045 4 3 2} --OID for ecdsaWithSHA256 (ANSI X9.62 ECDSA algorithm with SHA256)
10    },
11    issuer rdnSequence:
12    {
13      {
14        {
15          type {2 5 4 3}, --OID for CommonName
```

```
16     value "..." --shall match the subject of the issuer certificate
17             --(shall use PrintableString or UTF8String format)*
18 }
19 }
20 },
21 validity
22 {
23     notBefore Time: "...", --shall use UTCTime or GeneralizedTime as defined in [3]
24     notAfter Time: "..." -- shall use UTCTime or GeneralizedTime as defined in [3]
25 },
26 subject rdnSequence:
27 {
28 {
29 {
30     type {2 5 4 3}, --OID for CommonName
31     value "..." --contains the subject of the certificate (instance_CA_identifier)
32             --(shall use PrintableString or UTF8String format)
33 }
34 }
35 },
36 subjectPublicKeyInfo
37 {
38     algorithm
39 {
40         algorithm {1 2 840 10045 2 1}, --OID for ecPublicKey (ANSI X9.62 public key type)
41         parameters {1 2 840 10045 3 1 7} --OID for prime256v1(ANSI X9.62 named elliptic curve)
42 },
43 subjectPublicKey '04...'H
44             --the public key pre-pended with 04 to indicate uncompressed format
45 },
46 extensions
47 {
48 {
49     extnID {install_param_oid_instance_ca}, --OID for Instance CA extension
50     critical TRUE,
51     extnValue '...'H --DER encoding for instance-ca-cert-extension-data as per Listing 15-15
52 },
53 {
54     extnID {2 5 29 15}, --KeyUsage standard extension
55     critical TRUE,
56     extnValue '03020204'H --DER encoding for KeyUsage, CertSign only
57 },
58 {
59     extnID {2 5 29 19}, --BasicConstraints standard extension
60     critical TRUE,
61     extnValue '30060101FF020100'H -- DER encoding for BasicConstraints cA=TRUE, pathLenConstraint=0
62 },
63 {
64     extnID {2 5 29 35}, --OID for AuthorityKeyIdentifier standard extension
65     critical FALSE,
66     extnValue '...'H -- DER encoding of an AuthorityKeyIdentifier sequence as defined in [3], containing only
67             -- a KeyIdentifier element. The KeyIdentifier is an OCTET STRING containing the
68             -- 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey
69             --from the issuer certificate (excluding the tag, length, and number of unused bits)
70 },
71 {
72     extnID {2 5 29 14}, --OID for SubjectKeyIdentifier standard extension
73     critical FALSE,
```

```

74     extnValue ...'H --160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey
75         --(excluding the tag, length, and number of unused bits)
76     },
77     {
78         extnID {...}, --Optional extensions added by the issuer, content not verified by the applet
79         critical FALSE,
80         extnValue ...'H --DER encoding of the extension
81     },
82     ...
83   },
84 },
85     signatureAlgorithm
86   {
87     algorithm {1 2 840 10045 4 3 2}
88   },
89     signatureValue ...'H --the certificate signature by the issuer computed as per RFC 5280 [3]
90 }

```

* The selection of format between PrintableString and UTF8String is outlined in Section B.2.4 of Appendix B

1 *Table 15-22: AUTHORIZE ENDPOINT Internal Buffer Content Before Processing (offset 0)*

Tag	Length (bytes)	Description	Field is	Domain Version
7F20 _h	variable	External CA certificate as per Listing 15-13 from the receiver endpoint	optional	V-OD-FW
7F22 _h	variable	Instance CA Certificate as per Listing 15-16 from the receiver endpoint	optional	V-OD-FW
7F24 _h	variable	Endpoint Creation certificate as per Listing 15-5 from the receiver endpoint	mandatory	V-OD-FW
Encryption Key attestation as per Table 15-47 from the receiver endpoint		optional	D-VS	

2 *Table 15-23: AUTHORIZE ENDPOINT Attestation Data Fields*

Tag	Length (bytes)	Description	Field is	Domain Version
41 _h	1	version = 01 _h , the version of the data structure	mandatory	Informative
92 _h	8	random	mandatory	OD-FD-KS
5A _h	65	public key from the receiver endpoint creation certificate present in internal buffer	mandatory	OD-FD-KS
58 _h	variable	arbitrary_data, field present in command payload	optional	N/A
93 _h	4	usage = 91712C44 _h	mandatory	OD-FD-KS

3 *Table 15-24: AUTHORIZE ENDPOINT Internal Buffer Content After Processing (offset 0)*

Tag	Length (bytes)	Description	Field is	Domain Version
9E _h	64	sender endpoint signature over fields from Table 15-23	mandatory	OD-FD-KS
92 _h	8	random	mandatory	OD-FD-KS
97 _h	65	encsender_ePK	optional	OD-FD-KS
4A _h	variable	encrypted_mailbox mac	optional	OD-FD-KS

1

Listing 15-17: AUTHORIZE ENDPOINT Processing

```
1 input: internal_buffer, key_identifier(), mailbox_offset, mailbox_length, arbitrary_data
2 output: response_length, signature(), encrypted_mailbox, mac, encsender_ePK)
3 begin
4   if key_identifier does not match with any existing endpoint or target endpoint is terminated
5     return 6A88h
6   if AUTHORIZE ENDPOINT is not allowed on selected sender endpoint as per Table 15-13
7     return 6900h
8   if ua_performed != true -- ua_performed true if user authentication performed, false otherwise
9     return 6985h
10  if cod.internal_buffer is not formatted as described in Table 15-22
11    return 6E09h
12  if a certificate from chain does not comply with X.509 ASN.1 as per Listing A-4
13    return 6400h
14  if a certificate from chain cannot be verified as per verifier rules defined in Appendix A.4.7 and data defined in Listing 15-5,
  Listing 15-13, and Note: applet_version and platform_information are immutable values assigned
  at the time of the Instance CA certificate is issued. It might not reflect the current
  applet_version or platform_information (e.g., in case of applet or Secure Element platform
  upgrade).
```

Listing 15-16

```
15  return 6400h
16  if Endpoint certificate configuration has different vehicle_identifier than sender endpoint
17    return 6E10h
18  if Endpoint certificate any bit from option_group_1 is set while the same bit is cleared on sender endpoint
19    return 6E11h
20  if Endpoint certificate option_group_1 bit4 is set while bit5 is cleared on sender endpoint
21    return 6E12h
22  if Endpoint certificate option_group_2 bit0 or bit2 or bit3 or bit5 or bit7 is set while the same bit is
23    cleared on sender endpoint
24    return 6E13h
25  if Endpoint certificate option_group_2 bit1 or bit4 or bit6 is cleared while the same bit is set
26    on sender endpoint
27    return 6E14h
28  if Endpoint certificate configuration has different vehicle_PK than nvm.endpoint.vehicle_PK
29    return 6E15h
30  if Endpoint certificate configuration has authorized_PK keys not present in
31    nvm.endpoint.authorized_PK configuration
32    return 6E16h
33  if External CA certificate is present in cod.internal_buffer and cannot be verified
34    with one of nvm.endpoint.authorized_PK public keys
35    return 6E19h
36  if Instance CA certificate is present in cod.internal_buffer and cannot be verified
37    with one of nvm.endpoint.authorized_PK public keys or with the public key of the External CA certificate
38    return 6E20h
39  if Endpoint certificate stored in cod.internal_buffer cannot be verified
40    with the public key of the Instance CA certificate
41    or with one of nvm.endpoint.authorized_PK public keys
42    or with the public key of the External CA certificate
43    return 6E21h
44  generate 8 bytes random as per Listing 15-39 to be added in data_fields according to Table 15-23
45  generate attestation signature over fields from Table 15-23 according to Listing 15-42 using nvm.endpoint.SK
46  if mailbox_offset and mailbox_length present in command payload
47  if Encryption key certificate not present in cod.internal_buffer
48    return 6E23h
49  if Encryption Key certificate present in cod.internal_buffer cannot be verified with
```

```

50   the public key of the Endpoint certificate present in cod.internal_buffer
51   return 6E24h
52   generate ephemeral key pair encsender_ePK, encsender_eSK according to Listing 15-40
53   generate KEseed according to Listing 15-44 using encsender_eSK, encreceiver_ePK
54   receiver_PK ← public key from receiver endpoint extracted from Endpoint certification present in internal buffer
55   derivation ← 08h
56   KEseed_half ← 16 most significant bytes of KEseed
57   generate KEenc according to Listing 15-48 using KEseed_half, encsender_ePK, encreceiver_ePK,
58     derivation, receiver_PK
59   derivation ← 12h
60   KEseed_half ← 16 least significant bytes of KEseed
61   generate KEmac according to Listing 15-48 using KEseed_half, encsender_ePK, encreceiver_ePK,
62     derivation, receiver_PK
63   encrypt and mac the selected confidential mailbox area according to Listing 15-49
64   if required by endpoint configuration
65     replace the exported portion of confidential mailbox with random bytes
66   clear cod.internal_buffer
67   fill cod.internal_buffer with signature, random[, encrypted_mailbox, mac, encsender_ePK] as described in Table 15-24
68   response_length ← length written in cod.internal_buffer
69   return response_length (2 bytes big-endian)
70   end

```

1 15.3.2.8 VIEW command

2 This command returns information on endpoints and Instance CA. The response is written in the
3 internal buffer and accessible using READ BUFFER command.

4 In case the internal buffer size does not allow storage of all requested data elements, the internal
5 buffer is filled with the maximum number of data elements permitted. No truncated data
6 elements are written in the internal buffer.

7 The applet should respond with a warning SW 6300_h (or any implementation-specific SW in the
8 range 63XX_h or 62XX_h) to indicate that requested data is larger than the internal buffer size, and
9 the length of the data written to the internal buffer which may be read using READ BUFFER
10 command.

11 key_identifier is defined as the SHA-1 hash of the value of the BIT STRING subjectPublicKey
12 of the target endpoint (excluding the tag, length, and number of unused bits).

13 **command: CLA2 76 P1 P2 Lc (key_identifier) 00**

14 **response: [response_length] 9000**

15 The CLA2 is as defined in Table 15-3.

16

Table 15-25: View Parameters

P1	P2	Lc	Payload	Description	Domain Version
00 _h	00–FF _h	0	none	Request a list of LV-formatted key identifiers starting from the n-th element in memory, n represented by P2 (dummy endpoints are not returned).	Internal. N/A
01 _h	00 _h	20	key_identifier	Request endpoint setup information which includes all tags in the Table 15-52	Internal. N/A
02 _h	00 _h	20	key_identifier	Request endpoint creation certificate. Certificate parts (including certificate signature) may be re-generated.	Internal. N/A
03 _h	00 _h	0	none	Request current number of endpoints created in instance.	Internal. N/A

P1	P2	Lc	Payload	Description	Domain Version
04h	00h	20	key_identifier	Request endpoint state active 01h/terminated 00h.	Internal. N/A
80h	00-FFh	0	none	Request a list of LV-formatted instance CA identifiers starting from the n-th element in memory, n represented by P2.	Internal. N/A
82h	00h	0	none	Request instance additional information.	Internal. N/A
83h	00h	0	none	Request current number of Instance CA created in instance.	Internal. N/A
84h	00h	20	key_identifier	Request list of 1-byte transaction codes triggering user authentication on contactless interface; see Table 9-1 and Table 9-2 .	Internal. N/A
85h	00h	20	key_identifier	Request list of 1-byte transaction codes triggering user authentication on wired interface, see Table 9-1 and Table 9-2 .	Internal. N/A

1

Table 15-26: View Endpoint Identifier Response in Internal Buffer

Length	endpoint_identifier
20	key_identifier[n]
20	key_identifier[n+1]
20	...
20	key_identifier[last available Endpoint index (end of list) OR last Endpoint index that can fit in the internal buffer]

2

Table 15-27: View Instance CA Response in Internal Buffer

Length	instance_CA_identifier
1-30	instance_CA_identifier[n]
1-30	instance_CA_identifier[n+1]
1-30	...
1-30	instance_CA_identifier[last available Instance CA index (end of list) OR last Instance CA index that can fit in the internal buffer]

3

Listing 15-18: VIEW Processing

```

1   input (key_identifier, instance_CA_identifier)
2   output response_length
3   begin
4     if key_identifier or instance_CA_identifier provided as input does not exist
5       return 6400h
6     if P1=01h or P1=02h or P1=84h, or P1=85h,
7       if target endpoint with matching key_identifier is terminated
8         return 6A88h
9       if P1 = 01h
10      write in cod.internal_buffer endpoint setup information which includes all tags in Table 15-52, tag 4Eh shall be set to the
11      current value of the key_slot
12    else if P1 = 02h
13      write in cod.internal_buffer endpoint creation certificate as per Listing 15-5
14    else if P1 = 03h
15      write in cod.internal_buffer current number of endpoints created (1 byte)
16    else if P1 = 04h
17      if endpoint designated by key_identifier is active

```

```

19      write in 01h in cod.internal_buffer (1 byte)
20      else
21          write 00h in cod.internal_buffer (1 byte)
22      else if P1 = 82h
23          write in cod.internal_buffer instance additional information formatted as per Table 15-4
24      else if P1 = 83h
25          write in cod.internal_buffer current number of Instance CA created (1 byte)
26      else if P1 = 84h
27          cod.internal_buffer <- nvm.endpoint.transaction_code_list_cless
28      else if P1 = 85h
29          cod.internal_buffer <- nvm.endpoint.transaction_code_list_wired
30      else if P1 = 00h
31          write in cod.internal_buffer list of key_identifier starting from n-th element as per Table 15-26 with n=P2
32      else if P1 = 80h
33          write in cod.internal_buffer list of instance_CA_identifier starting from n-th element as per Table 15-27 with n=P2
34          response_length = number of bytes written in cod.internal_buffer
35      if requested data bigger than internal buffer size
36          return response_length (2 bytes big-endian), 6300h (or implementation specific warning SW in the ranges of 63XXh or
37          62XXh)
38      else
39          return response_length (2 bytes big-endian)
end

```

1 15.3.2.9 AUTH0 command

2 This command allows the vehicle to initiate the authentication procedure. In case a fast
3 transaction is requested by the vehicle, a cryptogram is returned, allowing the vehicle to
4 tentatively proceed with a fast transaction.

5 **command: CLA3 80 P1 P2 Lc [Table 15-29] 00**

6 **response: [Table 15-30] 9000**

7 The CLA3 is as defined in Table 15-3.

8 *Table 15-28: AUTH0 P1, P2 Parameters*

Description
P1 bits 0–7: bit0 Fast(1)/Standard(0) transaction request bit1 RFU (shall be set to 0) bit2 EXCHANGE commands will(1)/might(0) be sent during current transaction bit3–7 RFU (shall be set to 0) P2 bits 0–7: bit0–7 domain specific transaction_code Note: P1, P2 = FFFF _h is reserved (shall not be used in AUTH0)

9 *Table 15-29: AUTH0 Command Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
5C _h	2	protocol_version, the Digital Key applet transaction (V-D-TX) protocol version (ver.high ver.low) selected by vehicle, shall be one of the versions proposed by DK Applet in SELECT response	mandatory	V-D-TX
87 _h	65	vehicle_ePK prepended by 04 _h	mandatory	V-D-TX

Tag	Length (bytes)	Description	Field is	Domain Version
4C _h	16	transaction_identifier (randomly generated)	mandatory	V-D-TX
4D _h	8	vehicle_identifier (see Section 15.2)	mandatory	V-OD-FW

1 *Table 15-30: AUTH0 Response Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
86 _h	65	endpoint_ePK, the applet generated ephemeral public key prepended by 04 _h	mandatory	V-D-TX
9D _h	16	cryptogram, the authentication cryptogram returned by the endpoint	conditional	V-D-TX

- 2 Cryptogram (Tag 9Dh) shall not be included in the response payload if a standard transaction is requested by the vehicle (AUTH0 with P1 bit0 = 0), otherwise it shall be present.

4 *Listing 15-19: AUTH0 Processing*

```

1 input: protocol_version, P1, P2, vehicle_ePK, transaction_identifier, vehicle_identifier
2 output: endpoint_ePK, (cryptogram)
3 begin
4   if cod.transaction_state != select_done
5     return 6400h
6   if protocol_version is not supported (i.e., not part of the nvm.supported_protocol_versions_tlv)
7     return 6400h
8   cod.current_protocol_version ← protocol_version
9   generate an ephemeral key pair cod.endpoint_ePK, cod.endpoint_eSK according to Listing 15-40
10  endpoint_ePK ← 04h || cod.endpoint_ePK
11  cod.flag ← P1 || P2
12  begin constant time or unbiased random time section
13    find endpoint for corresponding vehicle_identifier excluding endpoints terminated or with visibility disabled
14    if no endpoint found
15      endpoint ← dummy_endpoint
16    if fast transaction requested
17      send to digital key framework notify_endpoint_not_found_fast with P1, P2
18    else
19      send to digital key framework notify_endpoint_not_found_standard with P1, P2
20    if fast transaction requested but not allowed on current interface as per Table 15-13
21      endpoint ← dummy_endpoint
22  end constant time or unbiased random time section
23  if interface is contactless
24    if transaction_code from P2 is present in nvm.endpoint.transaction_code_list_cless
25      if user authentication not performed
26        send to digital key framework notify_user_authentication_not_performed with P1, P2
27        endpoint ← dummy endpoint
28    else
29      if transaction_code from P2 is present in nvm.endpoint.transaction_code_list_wired
30        if user authentication is not performed
31          send to digital key framework notify_user_authentication_not_performed with P1, P2
32          endpoint ← dummy_endpoint
33  cod.vehicle_ePK ← vehicle_ePK
34  cod.transaction_identifier ← transaction_identifier
35  cod.mac_chaining ← 0000000000000000000000000000000000000000000000000000000000000000
36  cod.counter ← 00h
37  cod.endpoint ← endpoint
38  if Standard transaction requested

```

```

39 cod.transaction_state ← auth0_std_done
40 send to digital key framework notify_standard with nvm.endpoint.identifier with P1, P2
41 return endpoint_ePK (65 bytes)
42 else
43 cod.transaction_state ← auth0_fast_done
44 if endpoint.kpersistent_established = true
45     send to digital key framework
46         notify_fast_kpersistent_established with nvm.endpoint.identifier with P1, P2
47 else
48     send to digital key framework
49         notify_fast_kpersistent_not_established with nvm.endpoint.identifier with P1, P2
50 // NOTE: The cryptogram computation below applies
51 // even if endpoint.kpersistent_established is set to false,
52 // as nvm.endpoint.Kpersistent was initialized to a random value at endpoint creation
53
54 interface_byte ← C3h for wired interface or 5Eh for contactless interface
55 info ← cod.vehicle_ePK.x || cod.endpoint_ePK.x || cod.transaction_identifier || interface_byte ||
56     cod.flag || "VolatileFast" || | 5Ch || 02h || cod.current_protocol_version
57 keying_material_length ← 64
58 compute derived_keys according to Listing 15-45 using nvm.endpoint.Kpersistent, info, keying_material_length
59 KCmac ← subset of derived_keys at offset 0 with length 16
60 cod.Kenc ← subset of derived_keys at offset 16 with length 16
61 cod.Kmac ← subset of derived_keys at offset 32 with length 16
62 cod.Krmac ← subset of derived_keys at offset 48 with length 16
63
64 compute cryptogram according to Listing 15-50 using KCmac
65 return endpoint_ePK (65 bytes), cryptogram (16 bytes)
66
end

```

1 15.3.2.10 *AUTH1 command*

2 This command allows mutual authentication and establishment of a secure channel between the
3 vehicle and device.

4 This command may be used to optionally pre-compute a ranging key. If this option is
5 implemented, this command shall produce 80 bytes of keying material and the computed ranging
6 key shall subsequently be made available upon reception of the CREATE RANGING KEY
7 command. Otherwise, this command shall produce 48 bytes of keying material only.

8 **command: CLA3 81 00 00 Lc [Table 15-32] 00**

9 **response: [encrypted_payload] [mac] 9000**

10 The CLA3 is as defined in Table 15-3.

11 *Table 15-31: AUTH1 Vehicle Authentication Data Fields*

Tag	Length (bytes)	Description	Field is	Domain Version
4D _h	8	vehicle_identifier	mandatory	V-OD-FW
86 _h	32	endpoint_ePK.x	mandatory	V-D-TX
87 _h	32	vehicle_ePK.x	mandatory	V-D-TX
4C _h	16	transaction_identifier	mandatory	V-D-TX
93 _h	4	usage = 415D9569 _h	mandatory	V-D-TX

1

Table 15-32: AUTH1 Command Payload

Tag	Length (bytes)	Description	Field is	Domain Version
9E _h	64	vehicle_sig computed as per Listing 15-42 using fields from Table 15-31 and vehicle_SK	mandatory	V-D-TX

2

Table 15-33: AUTH1 Response Payload Before Encryption

Tag	Length (bytes)	Description	Field is	Domain Version
4E _h	1–8	key_slot	mandatory	V-OD-FW
9E _h	64	endpoint_sig, computed as per Listing 15-42 using fields from Table 15-34 and endpoint_SK	mandatory	V-D-TX
57 _h	4	counter_value; this field contains the current endpoint counter value in big-endian	deprecated*	N/A
4A _h	variable	confidential_mailbox_data_subset. This tag shall be included if it was included previously in SETUP ENDPOINT command	conditional	V-OD-FW
4B _h	variable	private_mailbox_data_subset. This tag shall be included if it was included previously in SETUP ENDPOINT command	conditional	V-OD-FW

3 * This field has been deprecated and shall no longer be used

4

Table 15-34: Endpoint Authentication Data Fields

Tag	Length (bytes)	Description	Field is	Domain Version
4D _h	8	vehicle_identifier	mandatory	V-OD-FW
86 _h	32	endpoint_ePK.x	mandatory	V-D-TX
87 _h	32	vehicle_ePK.x	mandatory	N/A
4C _h	16	transaction_identifier	mandatory	V-OD-FW
93 _h	4	usage = 4E887B4C _h	mandatory	V-OD-FW

5

Listing 15-20: AUTH1 Processing

```

1 input. vehicle_sig
2 output. encrypted_payload, mac
3 begin
4   if cod.transaction_state != auth0_std_done AND cod.transaction_state != auth0_fast_done
5     return 6400h
6   if standard transaction not allowed on this interface as per Table 15-13
7     return 6900h
8   endpoint ← cod.endpoint
9   fetch nvm.endpoint.vehicle_PK
10  verify vehicle_sig according to Listing 15-43 using nvm.endpoint.vehicle_PK and fields from Table 15-31
11  if vehicle_sig is not verified
12    send to digital key framework notify_vehicle_authentication_failed
13    return 6400h
14  send to digital key framework notify_vehicle_authentication_success
15  compute cod.Kdh according to Listing 15-44 using cod.transaction_identifier, cod.vehicle_ePK, cod.endpoint_eSK
16  interface_byte ← C3h for wired interface or 5Eh for contactless interface
17  info ← cod.vehicle_ePK.x || cod.endpoint_ePK.x || cod.transaction_identifier || interface_byte ||
18    cod.flag || "Volatile" || 5Ch || 02h || cod.current_protocol_version
19  if wired interface

```

```

20   keying_material_length ← 48 or 80
21   else
22     keying_material_length ← 48
23     compute derived_keys according to Listing 15-45 using cod.Kdh, info, keying_material_length
24     cod.Kenc ← subset of derived_keys at offset 0 with length 16
25     cod.Kmac ← subset of derived_keys at offset 16 with length 16
26     cod.Krmac ← subset of derived_keys at offset 32 with length 16
27     if wired interface and keying_material_length = 80
28       cod.URSK ← subset of derived_keys at offset 48 with length 32
29     info ← cod.vehicle_ePK.x || cod.endpoint_ePK.x || cod.transaction_identifier || interface_byte ||
30     cod.flag || "Persistent" || 5Ch || 02h || cod.current_protocol_version
31     keying_material_length ← 32
32     compute derived_keys according to Listing 15-45 using cod.Kdh, info, keying_material_length
33     Kpersistent ← subset of derived_keys at offset 0 with length 32
34     append nvm.endpoint.key_slot to response payload
35   atomic start
36     if fast transaction allowed by endpoint configuration as per Table 15-13 (option_group_1 bit1 and/or bit3 is set)
37       nvmwrite nvm.endpoint.Kpersistent ← Kpersistent
38       nvmwrite nvm.endpoint.kpersistent_established = true
39   atomic commit
40   compute endpoint_sig over fields Table 15-34 according to Listing 15-42 using endpoint.SK
41   append endpoint_sig to response payload
42   append confidential_mailbox_data_subset portion to response payload as per configuration in Section 15.3.2.21
43   append private_mailbox_data_subset to response payload
44   encrypt and mac response payload according to Listing 15-47 using cod.Kenc, cod.Krmac, cod.counter, cod.mac_chaining
45   cod.transaction_state ← auth1_done
46   return mac, encrypted_payload
47 end

```

1 **15.3.2.11 PRESENCE0 command**

2 This command allows the vehicle to initiate the Check Presence transaction.

3 **command: CLA3 3D 00 00 Lc [Table 15-29] 00**

4 **response: [Table 15-35] 9000**

5 The CLA3 is as defined in Table 15-3.

6 *Table 15-35: PRESENCE0 Response Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
86 _h	65	endpoint_ePK, the applet generated ephemeral public key prepended by 04 _h	mandatory	V-D-TX

7 *Listing 15-21: PRESENCE0 Processing*

```

1   input: protocol_version, vehicle_ePK, transaction_identifier, vehicle_identifier
2   output: endpoint_ePK
3   begin
4     if cod.transaction_state != select_done
5       return 6400h
6     If protocol_version is not supported (i.e., not part of nvm.supported_protocol_versions_tlv)
7       return 6400h
8     cod.current_protocol_version← protocol_version
9     generate an ephemeral key pair cod.endpoint_ePK, cod.endpoint_eSK according to Listing 15-40
10    endpoint_ePK ← 04h || cod.endpoint_ePK
11    cod.flag ← 5Bh
12    begin constant time or unbiased random time section
13      find endpoint for corresponding vehicle_identifier excluding endpoints terminated or with visibility disabled

```

```

14   if no endpoint found
15     endpoint ← dummy_endpoint
16     send to digital key framework notify_endpoint_not_found_presence
17   end constant time or unbiased random time section
18   cod.endpoint ← endpoint
19   cod.vehicle_ePK ← vehicle_ePK
20   cod.transaction_identifier ← transaction_identifier
21   cod.transaction_state ← presence0_done
22   cod.mac_chaining ← 00000000000000000000000000000000h
23   cod.counter ← 00h
24   send to digital key framework notify_presence with nvm.endpoint.identifier
25   return endpoint_ePK (65 bytes)
26 end

```

1 15.3.2.12 *PRESENCE1 command*

2 This command allows an authenticated vehicle to retrieve the endpoint key_slot.

3 **command: CLA3 3E 00 00 Lc [Table 15-36] 00**

4 **response: [encrypted_payload] [mac] 9000**

5 The CLA3 is as defined in Table 15-3.

6 *Table 15-36: PRESENCE1 Command Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
9E _h	64	vehicle_sig computed as per Listing 15-42 using fields from Table 15-37 and vehicle_SK	mandatory	V-D-TX

7 *Table 15-37: PRESENCE1 Vehicle Authentication Data Fields*

Tag	Length (bytes)	Description	Field is	Domain Version
4D _h	8	vehicle_identifier	mandatory	V-OD-FW
86 _h	32	endpoint_ePK.x	mandatory	V-D-TX
87 _h	32	vehicle_ePK.x	mandatory	V-D-TX
4C _h	16	transaction_identifier	mandatory	V-D-TX
93 _h	4	usage = 47165979 _h	mandatory	V-D-TX

8 *Table 15-38: PRESENCE1 Response Payload Before Encryption*

Tag	Length (bytes)	Description	Field is	Domain Version
4E _h	1–8	key_slot	mandatory	V-OD-FW

9 *Listing 15-22: PRESENCE1 Processing*

```

1   input: vehicle_sig
2   output: encrypted_payload, mac
3   begin
4     if cod.transaction_state != presence0_done
5       return 6400h
6     cod.transaction_state ← cp_done
7     endpoint ← cod.endpointh
8     verify vehicle_sig according to Listing 15-43 using nvm.endpoint.vehicle_PK and fields from Table 15-37
9     if vehicle_sig is not verified
10      send to digital key framework notify_vehicle_authentication_failed

```

```

11   return 6400h
12   send to digital key framework notify_vehicle_authentication_success
13   compute Kdh according to Listing 15-44 using cod.transaction_identifier, cod.vehicle_ePK, cod.endpoint_eSK
14   interface_byte ← 5Eh for contactless interface
15   info ← cod.vehicle_ePK.x || cod.endpoint_ePK.x || cod.transaction_identifier || interface_byte ||
16     cod.flag || "Volatile" || 5Ch || 02h || cod.current_protocol_version
17   keying_material_length ← 48
18   compute derived_keys according to Listing 15-45 using Kdh, info, keying_material_length
19   cod.Kenc ← subset of derived_keys at offset 0 with length 16
20   cod.Krmac ← subset of derived_keys at offset 32 with length 16
21
22   append nvm.endpoint.key_slot to response payload as per Table 15-38
23   encrypt and mac response payload according to Listing 15-47 using cod.Kenc, cod.Krmac, cod.counter, cod.mac_chaining
24   return mac, encrypted_payload
25   end

```

15.3.2.13 READ BUFFER command

This command is used to read temporarily stored data in an internal buffer. This supplements the APDU buffer for large payloads. The following command formats are allowed:

- If no secure channel is to be established between the Digital Key framework and the Digital Key applet, then either command format 1 or command format 2 may be used and implemented by the applet.
- If a secure channel is to be established between the Digital Key framework and the Digital Key applet (option A), command format 2 shall be used and shall be implemented by the applet.

Command format 1:

command: CLA2 B0 [offsetmsb] [offsetlsb] Le
response: [data] 90 00

The CLA2 is as defined in Table 15-3.

Command format 2:

command: CLA2 B0 [offsetmsb] [offsetlsb] Lc [Table 15-39] 00
response: [data] 90 00

The CLA2 is as defined in Table 15-3.

Table 15-39: READ BUFFER Command Payload

Tag	Length (bytes)	Description	Field is	Domain Version
80 _h	1	1-byte unsigned size of the data that shall be read from the internal buffer	mandatory	Internal. N/A

Listing 15-23: READ BUFFER Processing for Command Format 1

```

1   input: offsetmsb, offsetlsb, Le
2   output: data
3   begin
4     if data to be read overflows the cod.internal_buffer size
5     return 6400h
6     read cod.internal_buffer at offsetmsb || offsetlsb
7     return data

```

8 | **end**

1 Listing 15-24: READ BUFFER Processing for Command Format 2

```
1 input: offsetmsb, offsetlsb, sizeofdata
2 output: data
3 Begin
4   if data to be read overflows the cod.internal_buffer size
5     return 6400h
6   read cod.internal_buffer at offsetmsb || offsetlsb
7   return data
8 end
```

2 15.3.2.14 WRITE BUFFER command

3 This command is used to temporarily store data in an internal RAM buffer. This supplements the
4 APDU buffer for large payloads.

5 **command: CLA2 D0 [offsetmsb] [offsetlsb] Lc [data]**

6 **response: 90 00**

7 The CLA2 is as defined in Table 15-3.

8 Listing 15-25: WRITE BUFFER Processing

```
1 input: offsetmsb, offsetlsb, data
2 output: n/a
3 begin
4   if data to be written overflows the cod.internal_buffer size
5     return 6400h
6   write cod.internal_buffer at offsetmsb || offsetlsb with data from command payload
7   return
8 end
```

9 15.3.2.15 EXCHANGE command

10 This command reads, writes or sets data from confidential/private mailboxes. This command is
11 available after AUTH0 and/or AUTH1, depending on endpoint creation parameters. The
12 commands and responses are always wrapped in the currently established secure channel.

13 The command also is used to send notification messages to the Digital Key framework. In such
14 cases, the messages are not stored in mailbox.

15 In case command MAC verification fails, the applet returns 6982_h, and the atomic session, if any,
16 is aborted.

17 The set request fills the indicated memory area with the provided single byte value.

18 Multiple read, write, set, and notify operations can be present in a single EXCHANGE
19 command. Device shall support up to one notify operation to be transferred per EXCHANGE
20 command, if more than one notify operation is sent in an EXCHANGE command the behavior is
21 undefined

22 All read operations contained in an EXCHANGE command are returned in the order they appear
23 in the request buffer. All write/set operations contained in an EXCHANGE command are written
24 atomically and in the order they appear in the request buffer if the Atomic Session indicator flag
25 is set to 0. Write/set operations are atomic per group of EXCHANGE commands processed as
26 part of an atomic session.

1 An atomic session spanned over multiple commands is executed by setting the Atomic Session
2 indicator flag to 1 on a series of consecutive EXCHANGE commands. The session is closed, and
3 all data written in NVM by setting the Atomic Session indicator flag to 0 on the last
4 EXCHANGE command of the series.

5 When multiple EXCHANGE commands are part of an atomic session, all written values are
6 effectively readable only after successful execution of a command with the Atomic Session
7 indicator flag set to 0. All read operations occurring before that point return the old value.

8 If the transaction has been started over the wired interface, as conditionally permitted depending
9 on endpoint configuration, the Digital Key framework shall avoid prematurely interrupting the
10 transaction, e.g., by reselecting the applet. However, if it hasn't been notified of transaction end
11 after some unusually long time period (i.e., timeout), the Digital Key framework may decide to
12 forcefully end the transaction, e.g., by reselecting the applet.

13 **command: CLA4 C9 00 00 Lc [encrypted_command_payload] [command_mac] 00**
14 **response: [encrypted_response_payload] [response_mac] 90 00**

15 The CLA4 is as defined in Table 15-3.

16 *Table 15-40: Exchange Command Decrypted Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
option byte: bit0 Atomic Session indicator start(1)/stop(0), other bits RFU (left to 0)				mandatory
88 _h	3	offsetmsb offsetlsb length, read request in private mailbox	optional	V-D-TX
89 _h	3	offsetmsb offsetlsb length, read request in confidential mailbox	optional	V-D-TX
8A _h	variable	offsetmsb offsetlsb data, write request in private mailbox	optional	V-D-TX
8B _h	variable	offsetmsb offsetlsb data, write request in confidential mailbox	optional	V-D-TX
8C _h	4	offsetmsb offsetlsb length value, set request in private mailbox	optional	V-D-TX
8D _h	4	offsetmsb offsetlsb length value, set request in confidential mailbox	optional	V-D-TX
95 _h	5	offsetmsb offsetlsb lengthmsb lengthlsb value, set request in private mailbox	optional	V-D-TX
96 _h	5	offsetmsb offsetlsb lengthmsb lengthlsb value, set request in confidential mailbox	optional	V-D-TX
8E _h	variable	notify Digital Key framework with data present in this field. Data present in this field is stored in the Digital Key framework to be transmitted to the Vehicle OEM app. The maximum size supported for the value field of this TLV object is 211 bytes (size available for the <i>notify_message_in_exchange</i> notification to the Digital Key framework). This is reduced by 40 bytes if the EXCHANGE command also includes one or more write or set requests (to manage the <i>notify_mailbox_written</i> notification). In any case,	optional	V-D-TX

Tag	Length (bytes)	Description	Field is	Domain Version
		the size of this value field is limited by the available payload in the EXCHANGE command.		
		...other read, write or set requests	optional	

1 *Table 15-41: Exchange Response Decrypted Payload*

Length (bytes)	Description	Field is	Domain Version
variable	data read from request 1	conditional	V-D-TX
variable	data read from request 2	conditional	V-D-TX
...data read from request n		conditional	

2 *Listing 15-26: EXCHANGE Processing*

```

1 input: encrypted_command_payload, command_mac
2 output: encrypted_response_payload, response_mac
3 begin
4   if cod.counter = FFh
5     cod.transaction_state ← select_done
6     cod.atomic_session ← stopped
7     return 6900h
8
9   if cod.transaction_state != exchange_done and cod.transaction_state != auth1_done
10    and cod.transaction_state != auth0_fast_done
11    return 6400h
12
13   if cod.transaction_state == auth0_fast_done and EXCHANGE not allowed after AUTH0
14    return 6400h
15
16   cod.counter ← cod.counter + 1
17   endpoint ← cod.endpointh
18   verify and decrypt command according to Listing 15-46
19   inputs: cod.Kenc, cod.Kmac, cod.counter, cod.mac_chaining
20   output: decrypted_command_payload, mac_chaining_out
21   if command mac verification fails
22     cod.transaction_state ← select_done
23     cod.atomic_session ← stopped
24     return 6982h
25     cod.mac_chaining ← mac_chaining_out
26     cod.transaction_state ← exchange_done
27     if total requested output data in decrypted_command_payload is more than 239 bytes
28       return 6400h
29     if write/read/set requests present in decrypted_command_payload are out of mailbox boundaries
30       return 6400h
31     if notification requests are present in command payload
32       send to digital key framework notify_message_in_exchange
33     execute all notification requests from Table 15-40
34     execute all read requests from Table 15-40 on endpoint mailboxes
35     append read data to response payload for all read requests as per Table 15-41 from endpoint mailboxes
36     if Atomic Session indicator bit = 1
37       cod.atomic_session ← started
38     if cod.atomic_session = started
39       if all write/set requests exceeds cod.commit_buffer size
40         cod.atomic_session ← stopped
41       return 6A84h

```

```

41 store all write/set requests from Table 15-40 in cod.commit_buffer
42 if Atomic Session indicator bit = 0
43   atomic start
44     if cod.commit_buffer is not empty
45       execute all write/set requests pending in cod.commit_buffer on endpoint mailboxes
46       send to digital key framework notify_mailbox_written
47   atomic commit
48   cod.atomic_session ← stopped
49 else
50   atomic start
51     if decrypted_command_payload contains write or set requests
52       execute all write/set requests from Table 15-40 on endpoint mailboxes
53       send to digital key framework notify_mailbox_written
54   atomic commit
55   compute and return encrypted_response_payload and response_mac according to
56   Listing 15-47 using cod.Kenc, cod.Krmac, cod.mac_chaining, cod.counter
57 end

```

1 15.3.2.16 *CONTROL FLOW command*

2 This command allows the vehicle to indicate the final success or failure of the transaction or to
3 signal application-specific codes.

4 The P2 field is used for vehicle error codes, domain specific codes, or timing information.

5 The vehicle error codes are hints provided by the vehicle to the device in case of a transaction
6 failure. The decision to send such error codes is vehicle implementation-specific.

7 The variables P1 and P2 are version controlled under the V-D-TX domain version.

8 **command: CLA3 3C [Table 15-42] [Table 15-43]**

9 **response: 90 00**

10 The CLA3 is as defined in Table 15-3.

11 *Table 15-42: CONTROL FLOW P1 Parameters*

P1 value	Description
00 _h	transaction finished with failure
01 _h	transaction finished with success
40 _h	application specific
other values	RFU

12 *Table 15-43: CONTROL FLOW P2 Parameters*

P2 value	Description
00 _h	with P1 = 00 _h , 01 _h , or 40 _h , no information provided

P2 value	Description
00 _h -7F _h	Only with P1 = 00 _h , this range is used for the following vehicle error codes: 00 _h no information provided 01 _h public key not found 02 _h public key expired 03 _h public key not trusted 04 _h invalid signature 05 _h invalid channel 06 _h invalid data format 07 _h invalid data content 08 _h -7F _h RFU
80 _h -FF _h	With P1 = 00 _h , 01 _h , or 40 _h , this valid range is shown in Table 15-44

1

Table 15-44: CONTROL FLOW P1/P2 Values for Applet

P1	P2	Description	Reference
00 _h	A0 _h	Key deleted in/not known to vehicle. Vehicle was offline during key deletion in vehicle or unknown key without attestation package detected	See Note 4
00 _h	B0 _h	Cannot lock the vehicle because not all doors/trunk are closed	See Note 4, 5
00 _h	B1 _h	Cannot lock the vehicle because it is not in parking state (e.g. engine is still running/vehicle is still in driving state)	See Note 4, 5
01 _h	B0 _h	First approach successful	Section 19.5.8.2
01 _h	81 _h	Successful end, vehicle has completed owner pairing phase 3	Section 6.3.4.3
01 _h	90 _h	End, key is tracked, and all data have been written successfully into the mailboxes	Section 6.3.5.5
01 _h	91 _h	End, key is not tracked, key sharing is not possible, and the owner needs to go online to track the key before using it	Section 6.3.5.5
40 _h	81 _h	Optional token refill start	Section 6.3.5.2
40 _h	82 _h	Continue, Attestation package delete start	Section 6.3.5.4
40 _h	83 _h	Optional service data write start	See Note 1
40 _h	88 _h	Continue, key tracking response received in device, next step is to read the receipt from the mailbox	Section 6.3.5.2
40 _h	89 _h	Continue, key tracking response received in car, go directly to verification of key tracking receipt	Section 6.3.5.2
40 _h	90 _h	Extend reader processing time (do nothing on device)	See Note 2
40 _h	A0 _h	Long processing time (UI indication)	See Note 3
<i>Note 1:</i> Vehicle OEM may use this option for providing application-specific service data after engine start sequence.			
<i>Note 2:</i> Extended reader processing time is a command that can be sent by the reader to prevent a reader-side timeout. It has no effect on the device.			

P1	P2	Description	Reference
<i>Note 3:</i> When the vehicle executes operations that take longer than the usual UI-tolerated times, it should send this indication to the device. The device can show an appropriate UI, which is out of scope of this specification.			
<i>Note 4:</i> Used to indicate unsuccessful completion of transaction or inability to perform intended action due to a conflicting state on the vehicle.			
<i>Note 5:</i> Typically, only relevant for NFC transactions.			

1

Listing 15-27: CONTROL FLOW Processing

```

1 input: P1, P2
2 output: none
3 begin
4   if cod.transaction_state = auth0_std_done or cod.transaction_state = auth1_done or cod.transaction_state =
      auth0_fast_done or cod.transaction_state = exchange_done or cod.transaction_state = create_rk_done
5     if P1 == 00h
6       cod.transaction_state ← select_done
7       cod.atomic_session ← stopped
8       send to digital key framework notify_end_of_transaction with P1, P2 of CONTROL FLOW, P1, P2 of AUTH0 command
9     else if P1 == 01h
10    cod.transaction_state ← select_done
11    cod.atomic_session ← stopped
12    send to digital key framework notify_end_of_transaction with P1, P2 of CONTROL FLOW, P1, P2 of AUTH0 command
13  else
14    send to digital key framework notify_application_specific with P1, P2 of CONTROL FLOW, P1, P2 of AUTH0 command
15  else
16    if P1 == 00h or P1 == 01h
17      send to digital key framework notify_end_of_transaction with P1, P2 of CONTROL FLOW, outside of a fast or standard
          transaction
18    else
19      send to digital key framework notify_application_specific with P1, P2 of CONTROL FLOW, outside of a fast or standard
          transaction
20  End

```

- 2 15.3.2.17 **CREATE ENCRYPTION KEY command**
- 3 Create a confidential mailbox encryption key attestation in order to allow data insertion in the
- 4 endpoint. This encryption key is stored in NVM and valid for only one usage with the SET
- 5 CONFIDENTIAL DATA command. After single usage with the SET CONFIDENTIAL DATA
- 6 command, the key is erased from internal memory.
- 7 The attestation containing the encryption public key is stored in the internal buffer, accessible
- 8 with the READ BUFFER command.
- 9 **command: CLA2 8A 00 00 Lc [Table 15-45] 00**
- 10 **response: [response_length] 90 00**

11

Table 15-45: CREATE ENCRYPTION KEY Command Payload

Tag	Length (bytes)	Description	Field is	Domain Version
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW

1

Table 15-46: Encryption Key Attestation Data Fields

Tag	Length (bytes)	Description	Field is	Domain Version
41 _h	1	version = 01 _h , version of Encryption Key Attestation	mandatory	N/A. Informative Only
92 _h	8	random	mandatory	N/A. Informative Only
5F49 _h	65	encreceiver_ePK, the encryption ephemeral public key prepended by 04 _h	mandatory	N/A. Informative Only
5D _h	20	authority_key_identifier, 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey from the issuer (endpoint) certificate (as defined in Listing 15-5 excluding the tag, length, and number of unused bits)	mandatory	N/A. Informative Only
93 _h	4	usage = 49A8A741 _h	mandatory	N/A. Informative Only

2

Table 15-47: Encryption Key Attestation

Tag	Length (bytes)	Description	Field is	Domain Version
7F26 _h	variable	Encryption Key Attestation	mandatory	N/A. Informative Only
		content of Table 15-46	mandatory	N/A. Informative Only
9E _h	64	Receiver endpoint signature over fields from Table 15-46 with endpoint.SK	mandatory	N/A. Informative Only

3

Listing 15-28: CREATE ENCRYPTION KEY Processing

```

1 input.key_identifier
2 output.response_length, encryption_key_attestation
3 begin
4   if key_identifier does not match with any of the existing endpoints or target endpoint is terminated
5     return 6A88h
6   generate a key pair encreceiver_ePK, encreceiver_eSK according to Listing 15-40
7   atomic start
8     store nvm.endpoint.encreceiver.eSK using encreceiver_eSK
9     nvm.endpoint.encreceiver.isActive ← true
10   atomic commit
11   generate signature using nvm.endpoint.SK and fields from Table 15-46 according to Listing 15-42
12   generate data field encryption_key_attestation as per Table 15-47
13   store encryption_key_attestation in cod.internal_buffer
14   response_length ← length written in cod.internal_buffer
15   return response_length (2 bytes big-endian)
16 end

```

4 15.3.2.18 GET PRIVATE DATA command

5 Retrieve data in the private mailbox. The following command formats are allowed:

- 6 • If no secure channel is to be established between the Digital Key framework and the
- 7 Digital Key applet, then either Command Format 1 or Command Format 2 may be used
- 8 and implemented by the applet.
- 9 • If a secure channel is to be established between the Digital Key framework and the
- 10 Digital Key applet (option A), command format 2 shall be used and shall be implemented
- 11 by the applet.

- 1 Command Format 1:
- 2 **command: CLA2 78 [offsetmsb] [offsetlsb] Lc [Table 15-48] Le**
- 3 **response: [response] 90 00**
- 4 The CLA2 is as defined in Table 15-3.

- 5 Command Format 2:
- 6 **command: CLA2 78 [offsetmsb] [offsetlsb] Lc [Table 15-48] 00**
- 7 **response: [response] 90 00**
- 8 The CLA2 is as defined in Table 15-3.

9 *Table 15-48: GET PRIVATE DATA Command Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW
80 _h	1	1-byte unsigned size of the data that shall be read from the internal buffer	mandatory for command format 2 not required for command format 1	V-OD-FW

10 *Listing 15-29: GET PRIVATE DATA Processing for Command Format 1*

```
1 input: key_identifier, offset_msb, offset_lsb, Le
2 output: response
3 begin
4   if key_identifier does not match with any existing endpoints or target endpoint is terminated
5     return 6A88h
6   return response of Le bytes at offset offsetmsb || offsetlsb from the private mailbox
7 end
```

11 *Listing 15-30: GET PRIVATE DATA Processing for Command Format 2*

```
1 input: key_identifier, offset_msb, offset_ls, size_of_data
2 output: response
3 begin
4   if key_identifier does not match with any existing endpoints or target endpoint is terminated
5     return 6A88h
6   return response of size_of_data bytes at offset offset_msb || offset_ls from the private mailbox
7 end
```

- 12 15.3.2.19 *SET PRIVATE DATA command*
- 13 Store data in the private mailbox.
- 14 **command: CLA2 7A [offsetmsb] [offsetlsb] Lc [Table 15-49]**
- 15 **response: 90 00**
- 16 The CLA2 is as defined in Table 15-3.

1

Table 15-49: SET PRIVATE DATA Command Payload

Tag	Length (bytes)	Description	Field is	Domain Version
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW
4B _h	variable	data, field to be written in private mailbox	mandatory	V-OD-FW

2

Listing 15-31: SET PRIVATE DATA Processing

```

1 input: key_identifier, offsetmsb, offsetlsb, data
2 output: none
3 begin
4   if key_identifier does not match with any existing endpoints or target endpoint is terminated
5     return 6A88h
6   atomic start
7     write command data in NVM private mailbox at offsetmsb || offsetlsb
8   atomic commit
9   return
10  end

```

3 15.3.2.20 *SET CONFIDENTIAL DATA command*

4 Store data in the confidential mailbox. The appropriate data fields as described Table 15-51, shall
5 have been previously loaded in the internal buffer at offset 0 using the WRITE BUFFER
6 command. The CREATE ENCRYPTION KEY shall have been executed for this endpoint, and
7 the encryption key shall still be active.

8 **command: CLA2 7C [offsetmsb] [offsetlsb] Lc [Table 15-50]**

9 **response: 90 00**

10

Table 15-50: SET CONFIDENTIAL DATA Command Payload

Tag	Length (bytes)	Description	Field is	Domain Version
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW

11

Table 15-51: SET CONFIDENTIAL DATA Internal Buffer Content Before Processing

Tag	Length (bytes)	Description	Field is	Domain Version
97 _h	65	encsender_ePK	mandatory	N/A. Informative Only
4A _h	variable	encrypted_mailbox mac	mandatory	N/A. Informative Only

12

Listing 15-32: SET CONFIDENTIAL DATA Processing

```

1 input: key_identifier, offsetmsb, offsetlsb
2 output:
3 begin
4   if key_identifier does not match with any existing endpoints or target endpoint is terminated
5     return 6A88h
6   if command is not allowed as per Table 15-13
7     return 6900h

```

```

8   if nvm.endpoint.encreceiver.isActive ← false
9     return 6400h
10  if cod.internal_buffer does not contain data described in Table 15-51
11    return 6400h
12  nvm.endpoint.encreceiver.isActive ← false
13  generate KEseed according to Listing 15-44 using nvm.endpoint.encreceiver.eSK, encsender_ePK
14  derivation ← 08h
15  KEseed_half ← 16 most significant bytes of KEseed
16  generate KEenc according to Listing 15-48 using KEseed_half, encsender_ePK, nvm.endpoint.encreceiver_ePK,
17    derivation, nvm.endpoint.PK
18  derivation ← 12h
19  KEseed_half ← 16 least significant bytes of KEseed
20  generate KEmac according to Listing 15-48 using KEseed_half, encsender_ePK, nvm.endpoint.encreceiver_ePK,
21    derivation, nvm.endpoint.PK
22  verify and decrypt the selected confidential mailbox area according to Listing 15-49
23  if verification fails
24    randomize nvm.endpoint.encreceiver.eSK
25    return 6982h
26  atomic start
27  randomize nvm.endpoint.encreceiver.eSK
28  write command data in NVM confidential mailbox at offsetmsb || offsetlsb
29  atomic commit
30  clear cod.internal_buffer
31  return
32  end

```

1 15.3.2.21 *SETUP ENDPOINT command*

- 2 This command is used to change the default private/confidential mailbox content returned in
3 AUTH1 response; by default, no mailbox content is returned in the AUTH1 response.
4 This command is used to enable/disable individual endpoint visibility. When visibility is
5 disabled, the endpoint cannot be found during the AUTH0 command. Consequently, no
6 transaction can be executed with the endpoint. Other management commands remain available.
7 A newly created endpoint is visible by default.
8 Persistent configurations are applied immediately upon request (overriding current volatile
9 configuration, if any) and are reapplied on each SE power cycle.
10 Volatile configurations are applied immediately upon request.
11 When persistent and volatile configurations are present in the same command, the configurations
12 are applied in the order mentioned in [Listing 15-33](#).
13 **command: CLA2 7E 00 00 Lc [Table 15-52]**
14 **response: 90 00**
15 The CLA2 is as defined in Table 15-3.

16 *Table 15-52: SETUP ENDPOINT Command Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW

Tag	Length (bytes)	Description	Field is	Domain Version
4Ah	3	2 bytes offset (unsigned big-endian) 1 byte length to be returned from confidential mailbox in AUTH1 response (persistent)	optional	N/A. Informative Only
4Bh	3	2 bytes offset (unsigned big-endian) 1 byte length to be returned from private mailbox in AUTH1 response (persistent)	optional	N/A. Informative Only
82h	1	00h (disable) / 01h (enable) endpoint visibility contactless interface (persistent)	optional	V-D-TX
83h	1	00h (disable) / 01h (enable) endpoint visibility on contactless interface (volatile)	optional	V-D-TX
84h	0–16	List of 1-byte transaction codes triggering a user authentication if present in AUTH0 P2 over contactless interface (persistent), see Table 9-1, Table 9-2	optional	V-D-TX
85h	0–16	List of 1-byte transaction codes triggering a user authentication if present in AUTH0 P2 over wired interface (persistent), see Table 9-1, Table 9-2	optional	V-D-TX
4Eh	0–8	key_slot to be used by friend endpoint (1–8 byte) If tag present but empty, key_slot will be restored to initial_key_slot assigned during key creation.	optional	V-OD-FW
9Bh	1	00h (disable) / 01h (enable) endpoint visibility on wired interface (persistent)	optional	V-D-TX
9Ch	1	00h (disable) / 01h (enable) endpoint visibility on wired interface (volatile)	optional	V-D-TX

1

Listing 15-33: SETUP ENDPOINT Processing

```

1 input: key_identifier, Table 15-52
2 output:
3 begin
4   if key_identifier does not match any of the existing endpoints or target endpoint is terminated
5     return 6A88h
6   if offset/length out of mailbox boundaries
7     return 6400h
8   if the selected mailbox areas sizes do not fit in AUTH1 response
9     return 6400h
10  if one of the transaction_code present in nvm.endpoint.transaction_code_list_cless is not present in tag 84h
11    if user authentication has not been performed
12      return 6982h
13    if one of the transaction_code present in nvm.endpoint.transaction_code_list_wired is not present in tag 85h
14      if user authentication has not been performed
15      return 6982h
16  atomic start
17    store the persistent endpoint configuration from command payload
18    if tag 4Eh is present
19      if tag 4Eh is empty
20        nvmwrite nvm.endpoint.key_slot ← nvm.endpoint.initial_key_slot (restore initial value from key creation)
21      else
22        nvmwrite nvm.endpoint.key_slot ← key_slot (assign new value provided in tag 4Eh)

```

```
23 | atomic commit
24 | apply the persistent endpoint configuration from command payload
25 | apply the volatile endpoint configuration from command payload
26 | return
27 | end
```

1 15.3.2.22 *SETUP INSTANCE command*

2 This command is used to enable/disable endpoint visibility of all endpoints on the contactless
3 interface. If visibility is disabled, the endpoints appear as non-existent on the contactless
4 interface.

5 A newly created endpoint is enabled by default. Persistent configurations are applied
6 immediately upon request (overriding current volatile configurations, if any) and are reapplied
7 on each SE power cycle.

8 Volatile configurations are applied immediately upon request. When persistent and volatile
9 configurations are present in the same command, the configurations are applied in the order
10 mentioned in Listing 15-34 (SETUP INSTANCE processing).

11 **command:** CLA2 34 00 00 Lc [Table 15-53]

12 **response:** 90 00

13 The CLA2 is as defined in Table 15-3.

14 *Table 15-53: SETUP INSTANCE Command Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
82 _h	1	00 _h (disable) / 01 _h (enable) all endpoints visibility on contactless interface (persistent)	optional	V-D-TX
83 _h	1	00 _h (disable) / 01 _h (enable) all endpoints visibility on contactless interface (volatile)	optional	V-D-TX
9B _h	1	00 _h (disable) / 01 _h (enable) all endpoints visibility on wired interface (persistent)	optional	V-D-TX
9C _h	1	00 _h (disable) / 01 _h (enable) all endpoints visibility on wired interface (volatile)	optional	V-D-TX
90 _h	0	If this tag is present, all volatile configurations are cleared, and the current persistent configurations are applied.	optional	V-D-TX

15 *Listing 15-34: SETUP INSTANCE Processing*

```
1    input: Table 15-53
2    output:
3    begin
4       atomic start
5          store the persistent endpoint configuration for all endpoints in the instance
6       atomic commit
7          apply the persistent instance configuration for all endpoints in the instance
8          apply the volatile instance configuration for all endpoints in the instance
9       return
10   end
```

1 15.3.2.23 *SIGN command*

2 This command signs an arbitrary data field using the private key of the selected endpoint. The
3 command can be used with or without performing a user authentication; a different “usage”
4 value is included in the signature by the SE to differentiate these two contexts.

5 **command: CLA2 30 [Table 15-54] 00 Lc [Table 15-55] 00**

6 **response: [Table 15-57] 90 00**

7 *Table 15-54: SIGN Command Payload*

P1 value	Description	Domain Version
00 _h	User authentication is required	V-D-TX
01 _h	User authentication is not required	V-D-TX
02 _h -FF _h	RFU	N/A

8 The CLA2 is as defined in Table 15-3.

9 If P1 value is set to 0 (i.e., user authentication is required), the device will perform user
10 authentication. The Digital Key applet obtains information on whether or not the user
11 authentication was performed in a proprietary manner and uses the appropriate usage value for
12 SIGN response (see Listing 15-35).

13 *Table 15-55: SIGN Command Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the endpoint certificate that issues the attestation (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW
58 _h	32	Arbitrary data to be signed	mandatory	N/A. Informative Only

14 *Table 15-56: SIGN Data Fields*

Tag	Length (bytes)	Description	Field is	Domain Version
41 _h	1	Version = 01 _h , version of Signature Data Fields	mandatory	V-D-TX
92 _h	8	Random	mandatory	N/A. Informative Only
5D _h	20	key_identifier, 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey from the endpoint certificate that issued the attestation (excluding the tag, length, and number of unused bits)	mandatory	V-OD-FW
58 _h	32	arbitrary_data (provided in command)	mandatory	N/A. Informative Only
57 _h	4	counter_value of the endpoint	deprecated*	N/A
93 _h	4	usage = D074DA4F _h , if user authentication was performed usage = FC6F4C17 _h , if user authentication was not performed	mandatory	N/A. Informative Only

15 * This field has been deprecated and shall no longer be used.

16

1 The following table describes the content of the SIGN response.

2 *Table 15-57: SIGN Response*

Tag	Length (bytes)	Description	Field is	Domain Version
41 _h	1	version = 01 _h , version of Signature Data Fields	mandatory	V-D-TX
92 _h	8	Random	mandatory	N/A. Informative Only
5D _h	20	key_identifier	mandatory	V-OD-FW
57 _h	4	counter_value, this field contains the current endpoint counter value in big-endian	deprecated*	N/A
9E _h	64	Signature over fields from Table 15-56 with the endpoint private key	mandatory	N/A. Informative Only

3 * This field has been deprecated and shall no longer be used.

4 *Listing 15-35: SIGN Processing*

```

1 input: P1, key_identifier, arbitrary_data
2 output: signature, random, key_identifier, version, (counter)
3 begin
4   if key_identifier does not match any of the existing endpoints or target endpoint is terminated
5     return 6A88h
6   if SIGN not allowed on this endpoint as per Table 15-13
7     return 6900h
8   if P1 == 00h
9     if user authentication is not performed
10    return 6985h
11    usage D074DA4Fh
12  else if P1 == 01h
13    usage FC6F4C17h
14  else
15    return 6A86h
16    version ← 01h
17    generate 8 bytes random as per Listing 15-39 to be added in data_fields according to Table 15-56
18    generate data_fields to be signed according to Table 15-56
19    version, usage, key_identifier, random, arbitrary_data
20    generate signature according to Listing 15-42 using data_fields and nvm.endpoint.SK of the current endpoint
21    return signature, random, key_identifier, version
22 end

```

5 **15.3.2.24 MANAGE UA command**

6 This command provides the user authentication status. Implementing this command is optional
7 and the user authentication status may be provided using a proprietary method.

8 **command: CLA2 A0 00 00 Lc [Table 15-58] 00**
9 **response: 90 00**

10 The CLA2 is as defined in Table 15-3.

11 *Table 15-58: MANAGE UA Command Payload*

Tag	Length (bytes)	Description	Field is	Domain Version
80 _h	1	User authentication status 00 _h (user not authenticated) / 01 _h (user authenticated)	mandatory	V-OD-FW

Tag	Length (bytes)	Description	Field is	Domain Version
81 _h	variable	Discretionary data	optional	V-OD-FW
A1 _h	variable	TLV-structured discretionary data	optional	V-OD-FW

1 The applet shall ignore tags 81_h and A1_h if it has no interpretation for them

2 *15.3.2.25 onDeselect() Event*

3 This event is called on explicit or implicit deselection of the applet.

4 Event processing pseudo-code:

5 *Listing 15-36: onDeselect Event Processing*

```

1 input: reason
2 output: none
3 begin
4   if contactless interface
5     (optional) send to digital key framework notify_deselect with reason
6 end

```

6 *15.3.2.26 CREATE RANGING KEY command*

7 This command generates a ranging key (pre-derived URSK) and makes it available to a UWB along with arbitrary data. This command shall only be executed after successfully receiving AUTH1 Response. If the creation of pre-derived URSK is not possible (e.g. two pre-derived URSKs already exist), the status word 6484_h is returned.

11 The arbitrary_data field (up to 127 bytes) is optional and reserved for future usage.

12 **command: CLA2 71 00 00 Lc [arbitrary_data]**

13 **response: 90 00**

14 *Listing 15-37: CREATE RANGING KEY processing.*

```

1 input: arbitrary_data
2 output: none
3 begin
4   if UWB not supported by platform
5     return 6D00h
6   if cod.transaction_state != auth1_done and cod.transaction_state != exchange_done
7     return 6400h
8   cod.transaction_state ← create_rk_done
9
10  if cod.URSK not initialized
11    interface_byte ← C3h
12    info ← cod.vehicle_ePK.x || cod.endpoint_ePK.x || cod.transaction_identifier || interface_byte ||
13      cod.flag || "Volatile" || 5Ch || 02h || cod.current_protocol_version
14
15    keying_material_length ← 80
16    compute derived_keys according to Listing 15-45 using cod.Kdh, info, keying_material_length
17    URSK ← subset of derived_keys at offset 48 with length 32
18  else
19    URSK ← cod.URSK
20
21  key_identifier ← 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey from the currently

```

```

21     selected endpoint certificate (excluding the tag, length, and number of unused bits)
22 Make URSK, cod.transaction_identifier, key_identifier, arbitrary_data available
23 if storing URSK fails because queue is full
24     return 6484h
25 if URSK is not available for other reason
26     return 6400h
27 send to digital key framework notify_URSK_created
28     return 9000h
29 end

```

1 15.3.2.27 *DELETE RANGING KEYS command*

2 This command is optionally supported to delete either a given pre-derived URSK or all pre-
3 derived URSKs associated to an endpoint. This command shall not affect the URSK used in the
4 active ranging session.

5 **command: CLA2 41 00 00 Lc** [Table 15-59]

6 **response: 9000**

7 The CLA2 is as defined in [Table 15-3](#).

8 *Table 15-59: Delete Ranging Keys Request.*

Tag	Length (bytes)	Description	Field is	Domain Version
CF _h	4	UWB session ID	conditional	V-D-BT
50 _h	20	key_identifier, SHA-1 hash of the value of the BIT STRING subjectPublicKey of the target endpoint (excluding the tag, length, and number of unused bits)	conditional	V-OD-FW

9 *Listing 15-38: DELETE RANGING KEYS Processing.*

```

1    input: uwb_session_ID, key_identifier
2    output: none
3    begin
4     if uwb_session_ID is present
5       if uwb_session_ID does not match with any existing secure ranging session
6         return 6A88h
7       if uwb_session_ID matches the session id of a currently active secure ranging session
8         return 6400h
9       atomic start
10         delete pre-derived URSK (and related data) associated with uwb_session_ID
11         atomic commit
12         return 9000h
13       if key_identifier is present
14         if key_identifier does not match with any of the existing endpoints
15             return 6A88h
16         atomic start
17         delete all pre-derived URSKs (and related data) associated with key_identifier except if URSK used
18         for a currently active secure ranging session
19         atomic commit
20         return 9000h
21     end

```

10 15.3.2.28 *GET NOTIFICATION command*

11 Implementing this command is optional and the notification(s) may be provided using a
12 proprietary method. If Option D is supported, this command shall be implemented as defined in

1 Section 15.3.1.9, in order to retrieve the notification(s) that may have been generated during the
2 processing of the previous APDU command over the wired interface and the same logical
3 channel. **command: CLA2 A1 00 00 00**

4 **response: [list of notification TLVs (7F60h)] 90 00**

5 The CLA2 is as defined in [Table 15-3](#).

6 The command response contains a list of notifications. The coding of each notification is as
7 defined in Table 15-9. These notifications are appended to the list in the sequential order they are
8 generated. If no notification has been generated during the execution of the previous APDU
9 command, the response shall be empty.

10 15.3.3 Security

11 15.3.3.1 Cryptography Algorithms

12 *Listing 15-39: Generate Random*

```
1 input: length
2 output: random
3 begin
4   select random generator compliant with 'AIS31' standard
5   random = buffer filled with random bytes for the size indicated in input parameters
6   return random
7 end
```

13 *Listing 15-40: Generate Key Pair*

```
1 input: none
2 output: ecc.PK.x, ecc.PK.y, ecc.PK, ecc.SK
3 begin
4   set curve parameters ← 'ECC NIST P-256' as per \[8\]
5   generate ecc.PK.x, ecc.PK.y, ecc.SK
6   ecc.PK = (ecc.PK.x, ecc.PK.y)
7   return ecc.PK.x (32 bytes), ecc.PK.y (32 bytes), ecc.SK (32 bytes)
8 end
```

14 *Listing 15-41: Generate Identifier*

```
1 input: buffer
2 output: identifier
3 begin
4   hash ← SHA-1(buffer)
5   identifier = 6 first bytes of hash
6   return identifier
7 end
```

15 *Listing 15-42: Generate Attestation Signature*

```
1 input: data_fields, private_key
2 output: signature
3 begin
4   set curve_parameters ← 'ECC NIST P-256' as per \[8\]
5   signature ← ECDSA(curve_parameters, private_key, data_fields) using SHA-256 as per \[8\]
6   return signature (64 bytes)
7 end
```

16 *Listing 15-43: Verify Attestation Signature*

```
1 input: data_fields, public_key, signature
```

```
2 output: boolean
3 begin
4   hash ← SHA-256(data_fields)
5   set curve parameters ← 'ECC NIST P-256' as per [8]
6   boolean ← Verify(public_key, hash, signature)
7   return boolean (true/false)
8 end
```

1 Listing 15-44: Compute Shared Key with Diffie-Hellman

```
1 input: ePK, eSK, (transaction_identifier)
2 output: Kdh
3 begin
4   Compute the steps indicated by Section 4.3 of BSI TR-03111 [6] with the following mapping:
5   KeyAgreementProtocol : ECKA-DH
6   ( $p, a, b, G, n, h$ ) : ECC NIST P-256 Curve parameters as per [8]
7    $\hat{a}$  : eSK
8    $\hat{b}$  : ePK
9    $S_{AB}$  : key agreement output (shared secret point)
10   $Z_{AB}$  : computed from  $S_{AB}$  as per Section 3.1.3 of BSI TR-03111 [6]
11  key derivation input :  $Z_{AB}$ 
12  Key Derivation Function: KDFX9.63 as per Section 4.3.3 of BSI TR-03111 [6]
13  K : 256
14  H : SHA-256
15  SharedInfo : (transaction_identifier)
16  KeyData : Kdh (32 bytes)
17 end
```

2 Listing 15-45: Key Derivation

```
1 input: input_keying_material, info, keying_material_length
2 output: derived_keys
3 begin
4   Compute the steps indicated by Section 2 of RFC 5869 [13] with the following mapping:
5   Hash : SHA-256
6   IKM : input_keying_material
7   salt : NULL
8   info : info
9   L : keying_material_length
10  OKM : derived_keys
11 end
```

3 Listing 15-46: Secure Channel Command Decryption and Authentication

```
1 input: encrypted_command_payload, Kmac, Kenc, counter, mac_chaining_in
2 output: decrypted_command_payload, mac_chaining_out
3 begin
4   Compute the steps indicated by Section 6.2.6 of GPC_SPE_014 [7] with the following mapping:
5   CommandDataField - Plain Text : payload
6   PaddedCounterBlock : 00000000000000000000000000000000h || counter (1 byte, up to max FFh)
7   S-ENC : Kenc
8   CipheredCommandDataField : encrypted_command_payload
9   CommandDataField - PlainText : decrypted_command_payload
10
11  Compute the steps indicated by Section 6.2.4 of GPC_SPE_014 [7] with the following mapping:
12  Variationfromspec : MAC plaintext payload only includes command data field (CLA, INS, P1, P2, Lc, Le are omitted)
13  CommandDataField : encrypted_command_payload
14  MACChainingValue : mac_chaining_in (16 bytes)
15  NewMACChainingValue : mac_chaining_out (16 bytes)
16  S-MAC : Kmac
```

17 | **end**

1 Listing 15-47: Secure Channel Response Encryption and Authentication

```
1 input: payload, Krmac, Kenc, counter, mac_chaining_in
2 output: encrypted_payload, mac
3 begin
4   Compute the steps indicated by Section 6.2.7 of GPC_SPE_014 [7] with the following mapping:
5     ResponseDataField – PlainText : payload
6     PaddedCounterBlock : 80000000000000000000000000000000h || counter (1 byte, up to max FFh)
7     S-ENC : Kenc
8     CipheredResponseDataField : encrypted_payload
9
10  Compute the steps indicated by Section 6.2.5 of GPC_SPE_014 [7] with the following mapping:
11    Variationfromspec : MAC plaintext payload only includes MAC Chaining Value || encrypted payload, (SW is omitted)
12    ResponseDataField : encrypted_payload
13    MACChainingValue : mac_chaining_in (16 bytes)
14    S-RMAC : Krmac
15    R-MAC : mac (8 bytes)
16 end
```

2 Listing 15-48: Derive KEenc, KEmac

```
1 input: KEseed_half, sender_ePK, receiver_ePK, derivation, receiver_PK
2 output: KEenc, KEmac
3 begin
4   Compute the steps indicated by Section 4.1 of NIST SP800-108r1 [4] with the following mapping:
5     PRF: CMAC as defined by NIST SP800-38B using AES-128 block cipher [5]
6     h : 128
7     r : 8
8     KIN : KEseed_half
9     Label : 000000000000000000000000h concatenated with 08h/12h for KEenc/KEmac generation
10    Context : receiver_ePK.x || sender_ePK.x || receiver_PK.x
11    L : 0080h (AES-128)
12    PRF (KIN, Label || 00h || L || i || Context) : concatenation order
13    KOUT : KEenc(16 bytes) or KEmac(16 bytes)
14 end
```

3 Listing 15-49: Confidential Mailbox Encryption and Authentication

```
1 input: payload, KEenc, KEmac
2 output: encrypted_payload, mac
3 begin
4   Compute the steps indicated by Section 6.2.6 of GPC_SPE_014 [7] with the following mapping:
5     CommandDataField - PlainText : payload
6     PaddedCounterBlock : 00000000000000000000000000000001h
7     S-ENC : KEenc
8     CipheredCommandDataField : encrypted_payload
9
10  Compute the steps indicated by Section 6.2.4 of GPC_SPE_014 [7] with the following mapping:
11    Exception: The header [84, Ins P1 P2 Lc] shall not be included in the CMAC input,
12    only MAC Chaining Value and payload are used as input to the CMAC block.
13    CommandDataField : encrypted_payload
14    MACChainingValue : 00000000000000000000000000000000h
15    S-MAC : KEmac
16    C-MAC : mac (8 bytes)
17 end
```

4 Listing 15-50: Compute DeviceCryptogram

```
1 input: KCmac, endpoint_PK.x, vehicle_PK.x, transaction_identifier, vehicle_identifier
```

```
2   output: cryptogram
3   begin
4     Compute the steps indicated by Section 4.1 of NIST SP800-108r1 [4] with the following mapping:
5     PRF : CMAC as defined by NIST SP800-38B using AES-128 block cipher [5]
6     h : 128
7     r : 8
8     KIN : KCmac
9     Label : 00000000000000000000000000000000h concatenated with 32h
10    Context : vehicle_PK.x || endpoint_PK.x || transaction_identifier || vehicle_identifier
11    L : 0080h (AES-128)
12    PRF(KIN, Label || 00\hex || L || i || Context) : concatenation order
13    KOUT : cryptogram(16 bytes)
14  end
```

1 15.3.4 Optimization

2 This section provides a non-exhaustive list of optimizations that may improve the applet
3 performance. These optimizations depend on platform capabilities. The section is provided for
4 informational purposes only.

- 5 1. The ephemeral public key generated during the AUTH0 command could be pre-
6 generated at the end of each transaction (onDeselect). In order to avoid transaction delays
7 in case of communication errors, several pre-generated ephemeral keys could be used
8 until fallback to runtime generation. For transactions over wired interface, ephemeral key
9 pairs should be generated at runtime instead of from the pool of pre-generated keys
10 because of more relaxed timing constraints.
- 11 2. An ephemeral public key generated during the AUTH0 command could be re-used for a
12 limited amount of time (e.g., a few seconds) in order to avoid transaction-time key pair
13 generation on early failed transactions. However, an ephemeral key pair should be
14 deleted after being used in an ECDH operation.
- 15 3. The generation of the endpoint signature could be done in parallel with reader processing
16 in the vehicle after sending the AUTH0 response.
- 17 4. The next Kpersistent could be generated in parallel with reader processing in the vehicle
18 after sending the AUTH0 response.
- 19 5. The Kmac, Kenc, and Krmac could be generated in parallel with reader processing in the
20 vehicle after sending the AUTH0 response.
- 21 6. The AUTH1 response wrapped in the secure channel could be prepared in parallel with
22 reader processing in the vehicle after sending the AUTH0 response.

23 15.4 Helper Method

24 This section describes groups of APDU commands to be referred as a single functional block in
25 other specifications. The section is provided for informational purposes only.

1 15.4.1 *Hierarchy*

```
framework
  └── instance1
    ├── instanceCA1
    ├── instanceCA2
    ├── endpoint1
    ├── endpoint2
    ├── [...]
    └── endpointN
  └── instance2
    ├── instanceCA1
    ├── instanceCA2
    ├── endpoint1
    ├── endpoint2
    ├── [...]
    └── endpointN
  └── instanceN
    ├── instanceCA1
    └── endpoint1
  [...]
```

2

3 15.4.1.1 *Types*

4 For the purpose of description, variable names are prefixed with the following types:

- 5 • **bool_**: A boolean
- 6 • **bytes_**: An array of 8-bit bytes
- 7 • **uX_**: An unsigned value of X bits
- 8 • **obj**: An object
- 9 • **type[]**: A list of objects of the defined type

10 15.4.1.2 *framework.createInstance*

11 Create a new instance of the Digital Key applet. The instance can contain several endpoints.

12 **Inputs:**

- 13 **bytes_AID**: The instance AID.
- 14 **u32_endpointCountMax**: The maximum number of endpoints.
- 15 **u32_instanceCACountMax**: The maximum number of Instance CAs.
- 16 **u32_internalBufferSize**: The size of internal buffer.
- 17 **u32_maxAllocatablePrivateMailboxSize**: The maximum private mailbox size of an endpoint.
- 18 **u32_maxAllocatableConfidentialMailboxSize**: The maximum confidential mailbox size of an endpoint.

20 **Outputs:**

- 21 **obj_instance**: An object representing the Digital Key applet instance created

22 *Listing 15-51: framework.createInstance Processing*

```
1    input: bytes_AID, u32_endpointCountMax, u32_instanceCACountMax, u32_internalBufferSize
2    output: obj_instance
3    begin
```

```
4 request instance provisioning to the Device OEM Server with input parameters
5 create a handle representing the Digital Key applet instance, obj_instance.
6 return obj_instance
7 end
```

- 1 *15.4.1.3 framework.getInstance*
- 2 Retrieve an object representing the Digital Key applet instance.

3 **Inputs:**

4 **bytes_AID**: The instance AID

5 **Outputs:**

6 **obj_instance**: An object representing the selected instance

7 *Listing 15-52: framework.getInstance Processing*

```
1 input: bytes_AID
2 output: obj_instance
3 begin
4 create a handle representing the Digital Key applet instance, obj_instance
5 return obj_instance
6 end
```

- 8 *15.4.1.4 framework.view*
- 9 Get AID list of installed instance.

10 **Inputs:**

11 n/a

12 **Outputs:**

13 **bytes_instanceAID[]**: AID list of the instances installed

14 *Listing 15-53: framework.view Processing*

```
1 input: n/a
2 output: bytes_instanceAID[]
3 begin
4     return installed instance AID list
5 end
```

- 15 *15.4.1.5 framework.deleteInstance*
- 16 Delete a Digital Key applet instance and all related resources. The method described is currently
- 17 not called in this specification. It is included for completeness and potential usage by other
- 18 specifications.

19 **Inputs:**

20 **obj.instance**: Object representing the instance

21 *Listing 15-54: framework.deleteInstance Processing*

```
1 input: obj.instance
2 output: n/a
3 begin
```

4 | request instance deletion to the Device OEM Server with input parameters
5 | **end**

1 *15.4.1.6 instance.view*
2 Retrieve the instance related information. This method is called over an instance object
3 (*obj_instance*).

4 **Outputs:**

5 **u8_endpointCount**: The number of endpoints created.
6 **u8_endpointCountMax**: The maximum number of endpoints
7 **u16_internalBufferSize**: The size of the internal buffer
8 **bytes_keyIdentifier[]**: A list of key identifiers present on the instance
9 **bytes_instanceCAIdentifier[]**: A list of Instance CA identifiers present on the instance

10 *Listing 15-55: instance.view Processing*

```
1     input: obj_instance
2     output: u8_endpointCount,
3         u8_endpointCountMax,
4         u16_internalBufferSize,
5         bytes_keyIdentifier[],
6         bytes_instanceCAIdentifier[]
7     begin
8         retrieve instance informations using command in Section 15.3.2.8
9         return u8_endpointCount,
10         u8_endpointCountMax,
11         u16_internalBufferSize,
12         bytes_keyIdentifier[],
13         bytes_instanceCAIdentifier[]
14     end
```

11 *15.4.1.7 instance.createEndpoint*
12 Create an endpoint on the selected instance. This method is called over an instance object
13 (*obj_instance*).
14 If *bool_onlineCertificate* is set, the endpoint creation certificate is signed by the Device OEM
15 CA instead of the Instance CA. The device needs to be able to reach the Device OEM Server for
16 this feature to be available.

17 **Inputs:**

18 **bytes_vehicleIdentifier**: The vehicle identifier
19 **bytes_endpointIdentifier**: The endpoint_identifier
20 **bytes_instanceCAIdentifier**: The identifier of the Instance CA to be used for key
21 attestation
22 **u8_optionGroup1**: The option group 1
23 **u8_optionGroup2**: The option group 2
24 **u16_protocolVersion**: The Digital Key applet protocol version
25 **bytes_vehiclePK**: The vehicle's public key

```
1   bytes_notBefore: DER encoded GeneralizedTime.  
2   bytes_notAfter: DER encoded GeneralizedTime.  
3   bytes_authorizedPK[]: List of public keys of the authorized CAs (n × 65 bytes, n < 6).  
4   u16_confidentialMailboxSize: The confidential_mailbox_size (optional)  
5   u16_privateMailboxSize: The private_mailbox_size (optional)  
6   bytes_keySlot: The key slot (optional)  
7   u32_counterLimit: The signature counter limit (deprecated)
```

8 **Outputs:**

9 **obj_endpoint:** An object representing the created endpoint

10 *Listing 15-56: instance.createEndpoint Processing*

```
1   input: input list  
2   output: obj_endpoint  
3   begin  
4     generate input_data as per Table 15-7  
5     if input_data > 255  
6       send the WRITE BUFFER command as described in Section 15.3.2.14 with input input_data  
7       send the CREATE ENDPOINT command as described Section 15.3.2.4 without payload  
8     else  
9       send the CREATE ENDPOINT command as described Section 15.3.2.4 with input_data as payload  
10    create a handle representing the created endpoint, obj_endpoint  
11    return obj_endpoint  
12  end
```

11 *15.4.1.8 instance.deleteEndpoint*

12 Delete an endpoint on a specific instance. This method is called over an instance object
13 (obj_instance).

14 This method is currently not called in this specification. It is included for completeness and
15 potential usage by other specifications.

16 **Inputs:**

17 **bytes_keyIdentifier:** The key identifier

18 *Listing 15-57: instance.deleteEndpoint Processing*

```
1   input: bytes_keyIdentifier  
2   output: n/a  
3   begin  
4     send the DELETE ENDPOINT command as described in Section 15.3.2.6 with input_data as payload  
5     delete endpoint corresponding data on digital key framework  
6   end
```

19 *15.4.1.9 instance.getEndpoint*

20 Obtain a handle on an endpoint. This method is called over an instance object (obj_instance).

21 **Inputs:**

22 **bytes_keyIdentifier:** The key identifier

Outputs:

obj_endpoint: An object representing the endpoint

Listing 15-58: instance.getEndpoint Processing

```
1 input: bytes_keyIdentifier  
2 output: obj_endpoint  
3 begin  
4   create a handle representing the endpoint, obj_endpoint.  
5   return obj_endpoint  
6 end
```

4 15.4.1.10 *instance.setParameters*

5 Set instance parameters. This method is called over an instance object (`obj_instance`).

This method is currently not called in this specification. It is included for completeness and potential usage by other specifications.

8 Inputs:

bool_cless_visibility_persistent: If set to false, all endpoints are invisible from the contactless interface. This configuration is persistent across SE reboots.

bool_cless_visibility_volatile: If set to false, all endpoints are invisible from the contactless interface. This configuration is valid for the current power on session.

bool_wired_visibility_persistent: If set to false, all endpoints are invisible from the wired interface on AUTH0 command. This configuration is persistent across SE reboots.

bool_wired_visibility_volatile: If set to false, all endpoints are invisible from the wired interface on AUTH0 command. This configuration is valid for the current power on session.

Listing 15-59: instance.setParameters Processing

```
1 input: obj_instance, parameters
2 output: n/a
3 begin
4     set instance parameters using command as described in Section 15.3.2.22
5 end
```

19 15.4.1.11 endpoint.setParameters

20 Set endpoint parameters. This method is called over an endpoint object (`obj_endpoint`). Note
21 that all inputs listed below in this subsection are optional.

22 Inputs:

u16_offset_confidential: The offset from which data contained in confidential mailbox is returned in AUTH1.

u8_length_confidential: The length of confidential data returned in AUTH1.

u16_offset_private: The offset from which data contained in private mailbox are returned in AUTH1.

u8_length_private The length of private data returned in AUTH1.

1 **bool_cless_visibility_persistent:** If set to false, the endpoint is not visible from the
2 contactless interface. This configuration is persistent across SE reboots.
3 **bool_cless_visibility_volatile:** If set to false, the endpoint is not visible from the
4 contactless interface. This configuration is valid for the current power-on session.
5 **bytes_cless_transaction_codes:** List of 1-byte transaction codes triggering a user
6 authentication if present in AUTH0 P2 over contactless interface; see [Table 9-1](#) and
7 [Table 9-2](#).
8 **bytes_wired_transaction_codes:** List of 1-byte transaction codes triggering a user
9 authentication if present in AUTH0 P2 over contactless interface; see [Table 9-1](#) and
10 [Table 9-2](#).
11 **bool_wired_visibility_persistent:** If set to false, the endpoint is not visible from the
12 wired interface on AUTH0 command. This configuration is persistent across SE reboots.
13 **bool_wired_visibility_volatile:** If set to false, the endpoint is not visible from the wired
14 interface on AUTH0 command. This configuration is valid for the current power-on
15 session.

16 *Listing 15-60: endpoint.setParameters Processing*

```
1     input: obj_endpoint, parameters
2     output: n/a
3     begin
4       if bytes_cless_transaction_codes is missing any transaction code present in current configuration
5        perform user authentication
6       if bytes_wired_transaction_codes is missing any transaction code present in current configuration
7        perform user authentication
8       set endpoint parameters using command in Section 15.3.2.21
9     end
```

17 *15.4.1.12 endpoint.getCertificate*

18 Retrieve an endpoint certificate. This method is called over an endpoint object (obj_endpoint).

19 **Input:**

20 **bool_onlineCertificate request:** Attestation created by Device OEM CA (true) or by
21 Instance CA (false).

22 **Output:**

23 **bytes_endpointAttestation:** An endpoint certificate as per Listing 15-5.

24 *Listing 15-61: endpoint.getCertificate Processing*

```
1     input: bool_onlineCertificate
2     output: bytes_endpointCertificate
3     begin
4       if bool_onlineCertificate = true
5       if endpoint certificate issued by device OEM exists
6        return endpoint certificate issued by device OEM stored in digital key framework
7       else
8        send VIEW command to retrieve endpoint certificate
9        return endpoint certificate issued by instance CA
10      end
```

1 15.4.1.13 *endpoint.terminate*

2 Terminate an endpoint and obtain a termination attestation. This method is called over an
3 endpoint object (obj_endpoint).

4 **Inputs:**

5 **bytes_nonce:** Termination nonce (16 bytes)
6 **bytes_arbitrary_data:** Arbitrary data (optional, 1–40 bytes).

7 **Output:**

8 **obj_attestation:** A key termination attestation

9 *Listing 15-62: endpoint.terminate Processing*

```
1    input: bytes_nonce, (bytes_arbitrary_data)
2    output: bytes_terminationAttestation
3    begin
4        send TERMINATE ENDPOINT command to retrieve attestation as described
5        In Table 15-19 using bytes_nonce (and bytes_arbitrary_data)
6        return bytes_terminationAttestation
7    end
```

10 15.4.1.14 *endpoint.authorize*

11 Sign an externally received endpoint attestation along with arbitrary data and optionally export
12 confidential mailbox data. This method is called over an endpoint object (obj_endpoint).

13 *Table 15-60: Receiver Endpoint Key Attestation Signed by Sender*

Tag	Length (bytes)	Description	Field is
7F25 _h	variable	Receiving Endpoint Key Attestation	mandatory
content of Table 15-23			mandatory
9E _h	64	Signature with sender endpoint private key over Table 15-23	mandatory

14 **Inputs:**

15 **u16_offset:** Offset of confidential mailbox to be retrieved (optional)
16 **u16_length:** Length of confidential mailbox to be retrieved (optional)
17 **bytes_arbitraryData:** Arbitrary data to be signed (optional)
18 **bytes_externalCertificate:** External CA certificate of the receiver as per [Listing 15-13](#)
19 (optional)

20 **bytes_instanceCertificate Instance:** CA Certificate of the receiver as per **Note:** applet_version and
21 platform_information are immutable values assigned at the time of the Instance CA certificate is
22 issued. It might not reflect the current applet_version or platform_information (e.g., in case of
23 applet or Secure Element platform upgrade).

24 Listing 15-16 (optional)
25 **bytes_encryptionKeyAttestation:** Encryption key attestation of the receiver as per [Table](#)
26 [15-47](#) (optional)

1 **bytes_endpointCreationCertificate:** Endpoint creation certificate of the receiver as per
2 Listing 15-5

3 **Outputs:**

4 **bytes_attestation:** Attestation signed by the sender endpoint containing arbitrary data
5 and public key of the receiving endpoint

6 **bytes_encryptedData:** Data encrypted for the receiver confidential mailbox

7 **bytes_encSenderePK:** Encryption public key of the sender

8 *Listing 15-63: endpoint.authorize Processing*

```
1       input: (u16_offset, u16_length, bytes_arbitraryData, bytes_instanceCertificate,  
2            bytes_encryptionKeyAttestation, bytes_externalCertificate,) bytes_endpointCreationCertificate  
3       output: bytes_attestation, bytes_encryptedData, bytes_encSenderePK  
4       begin  
5           perform user authentication  
6           send WRITE BUFFER command in order to send bytes_instanceCertificate,  
7            bytes_encryptionKeyAttestation, bytes_endpointCreationCertificate  
8           send AUTHORIZE ENDPOINT command with u16_offset, u16_length, bytes_arbitraryData  
9           bytes_attestation = prepare table as per Table 15-60  
10          bytes_encryptedData = content of tag 4Ah from Table 15-24  
11          bytes_encSenderePK = content of tag 97h from Table 15-24  
12  
13          return bytes_attestation, bytes_encryptedData, bytes_encSenderePK  
14       end
```

9 *15.4.1.15 endpoint.createEncryptionKey*

10 Retrieve a single usage encryption key for confidential mailbox writing by other entities. This
11 method is called over an endpoint object (obj_endpoint).

12 **Output:**

13 **bytes_attestation:** Attestation containing encryption key as per Table 15-47

14 *Listing 15-64: endpoint.createEncryptionKey Processing*

```
1       input: none  
2       output: bytes_attestation  
3       begin  
4           send CREATE ENCRYPTION KEY command to retrieve attestation as described in Section 15.3.2.17  
5           return attestation  
6       end
```

15 *15.4.1.16 endpoint.setConfidentialData*

16 Write data in the confidential mailbox. This method is called over an endpoint object
17 (obj_endpoint).

18 **Inputs:**

19 **bytes_encSenderePK:** Public ephemeral key of the sender

20 **bytes_encData:** Encrypted data to be written in private mailbox

21 **u16_offset:** Offset of data to be written

1 *Listing 15-65: endpoint.setConfidentialData Processing*

```
1 input: bytes_encsenderpk, bytes_encredata, u16_offset, u16_offset
2 output: n/a
3 begin
4   send WRITE BUFFER command to write bytes_encsenderpk and bytes_encredata in internal buffer as described in Section 15.3.2.14
5   send SET CONFIDENTIAL DATA command to write in confidential mailbox as described in Section 15.3.2.20
6 end
```

2 *15.4.1.17 endpoint.getPrivateData*

3 Retrieve data from the private mailbox. This method is called over an endpoint object
4 (obj_endpoint).

5 **Inputs:**

6 **u16_offset:** Offset of private mailbox to be retrieved
7 **u16_length:** Length of private mailbox to be retrieved

8 *Listing 15-66: endpoint.getPrivateData Processing*

```
1 input: u16_offset
2 output: u16_length
3 begin
4   execute one or multiple GET PRIVATE DATA command as described in Section 15.3.2.18 to retrieve the
      requested data
5 end
```

9 *15.4.1.18 endpoint.setPrivateData*

10 Write data in the private mailbox. This method is called over an endpoint object (obj_endpoint).

11 **Inputs:**

12 **u16_offset:** Offset to start writing in private mailbox
13 **bytes_data:** Data to be written in private mailbox

14 *Listing 15-67: endpoint.setPrivateData Processing*

```
1 input: u16_offset, bytes_data
2 output: n/a
3 begin
4   Send SET PRIVATE DATA command to write in private mailbox as described in Section 15.3.2.19
5 end
```

15 *15.4.1.19 endpoint.sign*

16 Obtain a signature on arbitrary data. This method is called over an endpoint object
17 (obj_endpoint).

18 *Table 15-61: Arbitrary Data Attestation*

Tag	Length (bytes)	Description	Field is
7F2Dh	variable	Arbitrary Data Attestation	mandatory

Tag	Length (bytes)	Description	Field is
Content of Table 15-56			mandatory
9E _h	64	signature with endpoint.SK over fields from Table 15-56	mandatory

1 Inputs:

2 **bytes_arbitraryData**: Arbitrary data to be signed

3 Outputs:

bytes_attestation: An attestation as described in Table 15-61

5 Listing 15-68: endpoint.sign Processing

```
1 input: bytes_arbitraryData
2 output: bytes_attestation
3 begin
4     perform user authentication if required
5     send SIGN command to retrieve attestation as described in Table 15-61
6     return attestation
7 end
```

6 15.4.1.20 endpoint.auth0

7 Start the mutual authentication procedure. This method is called over an endpoint object
8 (`obj_endpoint`).

9 Inputs:

10 **bytes_epkinput**: Ephemeral public key of the vehicle
11 **bytes_transactionIdentifier**: Transaction identifier
12 **bytes_vehicleIdentifier**: Vehicle identifier

13 Outputs:

14 **bytes_epkoutput**: Ephemeral public key of the endpoint
15 **u16_protocolconversion**: Digital Key applet protocol version supported by endpoint

Listing 15-69: endpoint.auth0 Processing

```
1 input: bytes_epkinput, bytes_transactionIdentifier, bytes_vehicleIdentifier
2 output: bytes_epkoutput, u16_protocolversion
3 begin
4   send SELECT command using the AID of the obj_endpoint to retrieve u16_protocolversion
5   send AUTH0 command as described in Section 15.3.2.9 using bytes_epkinput, bytes_transactionIdentifier,
6   bytes_vehicleIdentifier
7   P1 parameters are Standard Transaction requested, User Authentication not requested and
8   retrieve bytes_epkoutput
9   return bytes_epkoutput, u16_protocolversion
end
```

17 15.4.1.21 endpoint.auth1

18 The second step of the mutual authentication procedure, endpoint.auth0 shall have been executed
19 previously. This method is called over an endpoint object (obj_endpoint).

1 Input:

bytes_signature: Signature of the third party entity wishing to authenticate

3 Output:

bytes_encryptedPayload: Encrypted payload

5 Listing 15-70: endpoint.auth1 Processing

```
1 input: bytes_signature
2 output: bytes_encryptedPayload
3 begin
4   send AUTH1 command using bytes_signature as described in Section 15.3.2.10
5   return bytes_encryptedPayload
6 end
```

6 15.4.1.22 endpoint.exchange

7 Read and write data in mailboxes upon establishment of a secure channel, provided that
8 endpoint.auth0 (and endpoint.auth1 depending on endpoint configuration) have been
9 successfully executed. This method is called over an endpoint object (obj_endpoint).

10 Input:

11 **bytes_encryptedInput**: Encrypted payload

12 Output:

bytes_encryptedOutput: Encrypted payload

Listing 15-71: endpoint.exchange Processing

```
1 input: bytes_encryptedInput
2 output: bytes_encryptedOutput
3 begin
4   send EXCHANGE command using bytes_encryptedInput as described in Section 15.3.2.15
5   return bytes_encryptedOutput
6 end
```

15 15.4.1.23 endpoint.createRangingKey

16 Generate a ranging key and make it available to UWB. Shall only be called after successfully
17 receiving AUTH1 Response.

18 Inputs:

19 **bytes_arbitraryData**: arbitrary data.

Listing 15-72: `endpoint.createRangingKey` processing.

```
1 input: bytes_arbitraryData
2 output: none
3 begin
4   send CREATE RANGING KEY command using bytes_arbitraryData as described in Section 15.3.2.26
5 end
```

1 **15.5 Vehicle Implementation**

2 **15.5.1 Security Guidelines**

3 The following items describe important implementation guidelines for the vehicle.

- 4 1. During the standard transaction, the vehicle shall always verify the endpoint signature
5 (endpoint_sig) before any other operation. A successful verification of this signature is
6 the only way for the vehicle to authenticate the endpoint.
- 7 2. During the standard transaction, the vehicle shall never modify its persistent memory
8 before having successfully verified the endpoint signature (endpoint_sig).
- 9 3. During the standard transaction, the vehicle shall never write data in the endpoint
10 confidential or private mailbox before having successfully verified the endpoint signature
11 (endpoint_sig).
- 12 4. During the standard and fast transactions, the vehicle shall verify that the use of
13 contactless interface is indicated in the response to the AUTH0 and AUTH1 command
14 when the transaction is performed over the NFC interface.[WCC1]

15 **15.5.2 Optimizations**

16 The following items describe recommended implementation guidelines for the vehicle.

- 17 1. One or several vehicle ephemeral keys can be pre-generated such that a fresh ephemeral
18 key is immediately ready when the next transaction starts.
- 19 2. If a vehicle only supports the fast transaction, a random number can be generated instead
20 of an ephemeral key pair.

16 CERTIFICATE

16.1 General

3 The PKI model described in this section is based on PKI mechanisms to build a chain of trust
4 down to a single Digital Key. The following certificate chains are defined:

- 5 1. Device certificate chain
- 6 2. Vehicle certificate chain
- 7 3. Owner pairing key attestation chain
- 8 4. Key sharing key attestation chain

9 The objective is to provide full offline capability through the verification chain.

10 Certificate expiry shall be checked by the KTS. If technical capabilities allow, the Digital Key
11 applet, vehicle, and owner device should also verify certificate expiry.

12 In this specification, two variants of the SE root of trust certificate chain model are provided.

- 13 • **Variant 1:** SE root of trust based on CASD: In this model (see Figure 16-1), the SE root
14 of trust is managed by the SE Root CA implemented by the CASD.
- 15 • **Variant 2:** SE root of trust based on DK applet associated security domain: In this model
16 (see Figure 16-2), the SE root of trust is managed using the secure channel of the security
17 domain associated with the DK applet.

18 The provisioning of the root of trust in the SE is out of scope of this specification.

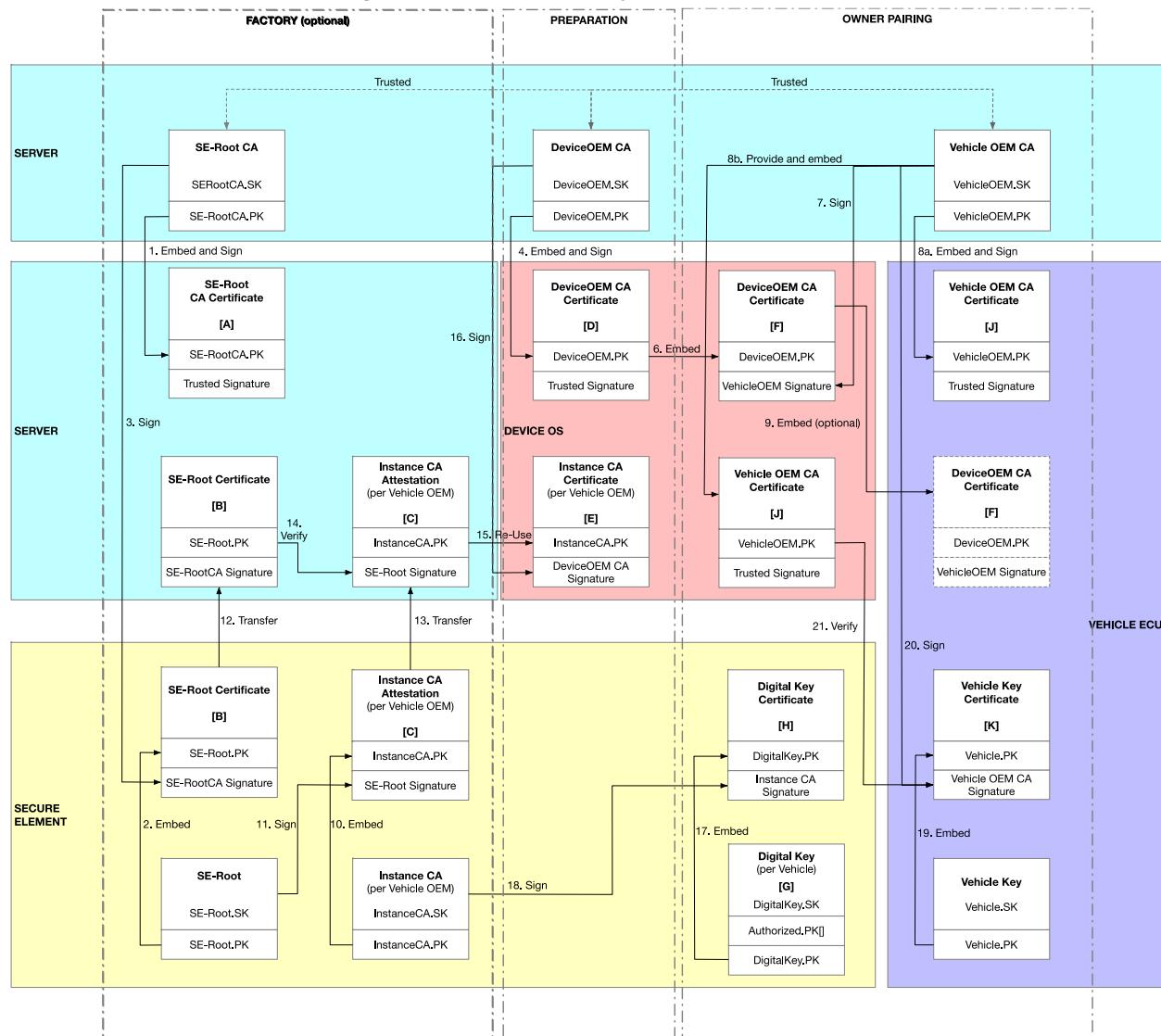
16.2 Certificates and Relationships

20 Figure 16-1 provides an overview of the overall certification chains when Variant 1 is
21 implemented.

22 The Device OEM may choose to directly embed the Vehicle OEM CA Certificate [J] in the
23 device OS as depicted in Figure 16-1. Alternatively, the Device OEM may choose to embed the
24 Vehicle OEM CA Certificate signed by the Device OEM CA [M] in the device OS.

1

Figure 16-1: Variant 1 Certification Chain Model

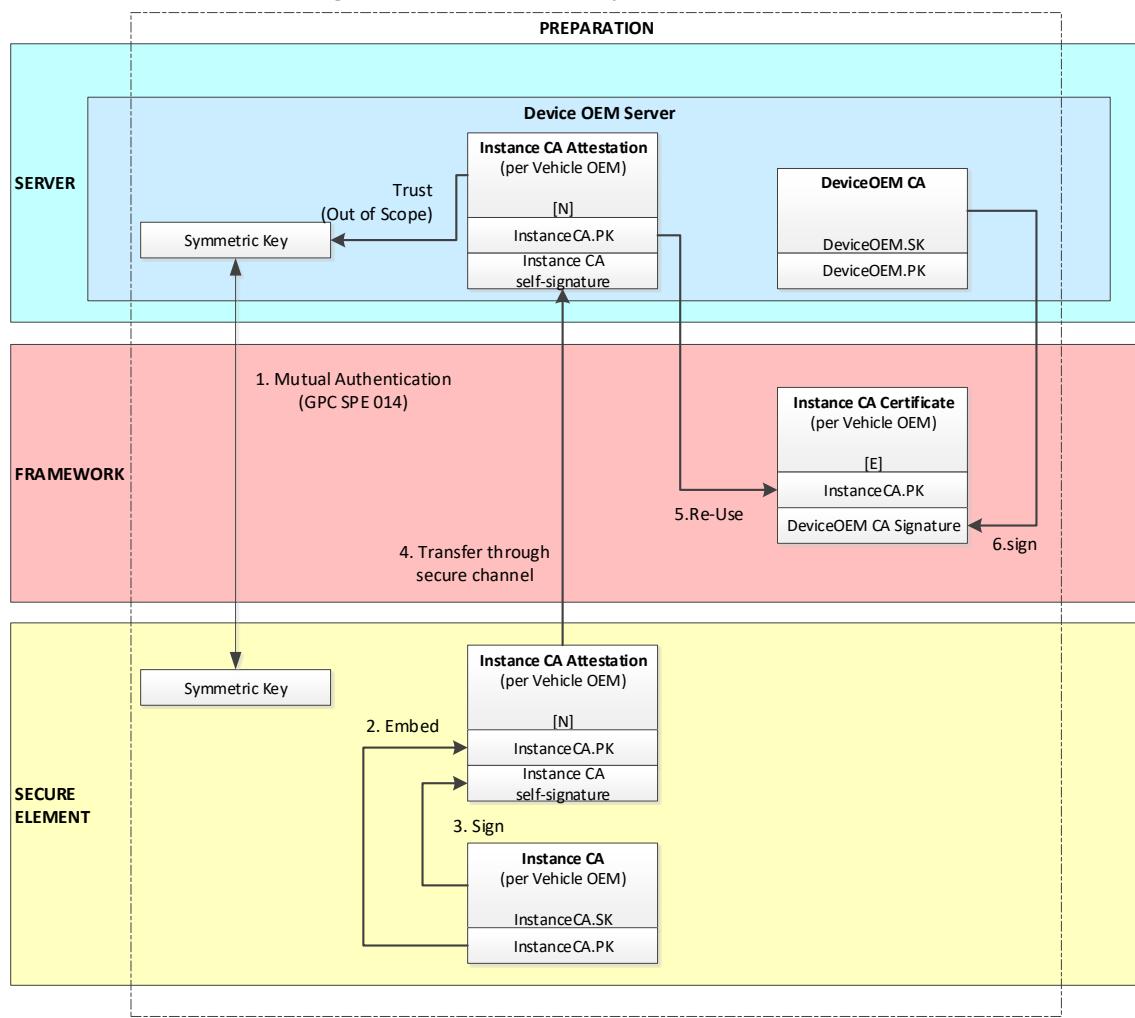


2

3 Figure 16-2 provides an adaptation when Variant 2 is implemented.

1

Figure 16-2: Variant 2 Certification Chain Model



2

- 3 [A] - SE Root CA Certificate – Variant 1
- 4 [B] - SE Root Certificate – Variant 1
- 5 [C] - Instance CA Attestation (signed by SE Root) per Vehicle OEM – Variant 1
- 6 [D] - Device OEM CA Certificate
- 7 [E] - Instance CA Certificate (signed by Device OEM CA) per Vehicle OEM
- 8 [F] - Device OEM CA Certificate (signed by Vehicle OEM CA)
- 9 [G] - Digital Key per Vehicle
- 10 [H] - Digital Key Certificate
- 11 [J] - Vehicle OEM CA Certificate
- 12 [K] - Vehicle Public Key Certificate
- 13 [L] - Digital Key Creation Data
- 14 [M] - Vehicle OEM CA Certificate (signed by Device OEM CA)
- 15 [N] - Instance CA Attestation (self-signed) per Vehicle OEM – Variant 2

1 16.2.1 *[A] – SE Root CA Certificate*

2 This certificate applies to Variant 1.
3 This certificate is provided by the entity that controls the root key pair in the SE, e.g., the SE
4 manufacturer. It is trusted by the Device OEM CA.
5 The SE Root CA private key has signed the SE Root Certificates [B] embedded in the SE. Those
6 certificates include immutable identifiers that could allow device tracking.
7 It is mandatory for every Digital Key-eligible Device OEM to be able to validate SE root
8 signatures from the SEs on its devices.

9 16.2.2 *[B] – SE Root Certificate*

10 This certificate applies to Variant 1.
11 The SE Root Certificate [B] is generated securely and attests to the identity of the SE. [B] is
12 signed by the SE Root CA. [B] is used to verify the Instance CA Attestation [C] created by the
13 Digital Key applet instance. If [C] is successfully verified by [B], the Device OEM signed
14 Instance CA Certificate [E] is issued by the Device OEM Server and stored in the device OS of
15 the corresponding device.
16 Within the SE, [B] is stored in the CASD. The provisioning of the [B] in the SE is out of scope
17 of this specification.

18 16.2.3 *[C] – Instance CA Attestation (signed by SE Root) per Vehicle OEM*

19 This certificate applies to Variant 1.
20 The Instance CA Attestation [C] is generated after an Instance CA is created by the Digital Key
21 applet instance. [C] includes the public key of the Instance CA and is signed by the private key
22 of the SE root. One Instance CA is created per Vehicle OEM. The Device OEM CA verifies and
23 signs [C] with its private key to issue a new Instance CA Certificate [E] (See Section [16.2.5](#)). [E]
24 is verifiable using the Device OEM CA Certificate [D] and Vehicle OEM signed Device OEM
25 CA Certificate [F].
26 Within the SE, the private key of the SE root is stored in the CASD. The provisioning of this
27 private key is out of scope of this specification.
28 The signature of [C] by the private key of the SE root may rely on a proprietary API
29 implemented by the CASD or may rely on the CASD signature service available through
30 AuthoritySignature interface as defined in [\[15\]](#).

31 16.2.4 *[D] – Device OEM CA Certificate*

32 This certificate is the Device OEM root of trust for the Digital Key system. This certificate is
33 trusted by the Device OEM and all Vehicle OEMs. The Vehicle OEM receives a CSR from the
34 Device OEM in order to countersign the Device OEM CA Certificate [D]. This then becomes a
35 Vehicle OEM signed Device OEM CA Certificate [F] with the same Device OEM public key.
36 Depending on the Device OEM's implementation or the Digital Key service deployment model,
37 the Device OEM CA Certificate [D] may be stored in the device OS or in the SE.

1 16.2.5 *[E] – Instance CA Certificate (Signed by Device OEM) per Vehicle OEM*

2 The Device OEM-signed Instance CA Certificate [E] contains the public key of the Instance CA
3 Attestation [C] (Variant 1) or [N] (Variant 2) and is signed by the private key of the Device
4 OEM CA. [E] allows the verification of the Instance CA Attestation using a Device OEM CA
5 Certificate [F]. Using the Device OEM CA signature removes the requirements of otherwise
6 having to trust the SE Root CA. It also anonymizes the static device information to external
7 parties. During the owner paring, [E] is transferred to the vehicle and is used to verify the Digital
8 Key Certificate [H].

9 The certificate should have a short lifetime and should be renewed often. Revocation may be
10 achieved by stopping the renewal process or through revocation methods (CRL, OCSP), which
11 are out of scope of this specification.

12 16.2.6 *[F] – Device OEM CA Certificate (Signed by Vehicle OEM)*

13 The Vehicle OEM CA public key is embedded in the Vehicle OEM CA Certificate [J] in the
14 vehicle (for owner pairing) and in authorized PK in the owner SE (for key sharing). This Vehicle
15 OEM CA public key is used to verify the Vehicle OEM signed Device OEM CA Certificate [F]
16 (note that [F] is also referred to as an “external CA certificate” in Section 15). [F] is then used to
17 verify the Device OEM signed Instance CA Certificate [E]. [F] shall be stored in the device OS
18 of all devices eligible for Digital Keys.

19 16.2.7 *[G] – Digital Key per vehicle*

20 [G] is the Digital Key generated by the Digital Key applet. There is one Digital Key per vehicle.
21 During owner pairing, all Digital Key elements are provided by the vehicle and transferred to the
22 device.

23 16.2.8 *[H] – Digital Key Certificate*

24 When a Digital Key is created, the applet signs the public key of the Digital Key structure using
25 the corresponding Instance CA’s private key (see Section 4.1) to create the Digital Key
26 Certificate [H]. The certificate is required for owner pairing and key sharing.

27 16.2.9 *[J] – Vehicle OEM CA Certificate*

28 This certificate is the Vehicle OEM root of trust for the Digital Key system. [J] is trusted by all
29 Device OEMs and the corresponding Vehicle OEM. The private key corresponding to the
30 Vehicle OEM CA Certificate is used to sign the Device OEM CA Certificate [D] to become [F].
31 [F] is transferred to the vehicle in order to verify the certificate chain at owner pairing, in which
32 [F] is first verified by [J]. This leads to the Digital Key attestation as shown in Figure 6-7.

33 The public key of this certificate is also embedded in the owner’s Digital Key (as authorized
34 PK). It is used during key sharing to verify the certificate chain, i.e., to validate the friend’s
35 public key origin.

36 The Device OEM may embed [J] on the device and use it to validate the authenticity of the
37 vehicle public key (i.e., Vehicle.PK) contained in [K].

1 **16.2.10 [K] – Vehicle Public Key Certificate**

2 Before accepting the Digital Key Creation Data [L] from the vehicle during owner pairing, the
3 device verifies the origin of the vehicle public key (i.e., Vehicle.PK) using the Vehicle Public
4 Key Certificate [K]. The Vehicle.PK is attested to by the Vehicle OEM CA Certificate [J].

5 *Note:* The device uses the Vehicle OEM CA Certificate [J] or Device OEM signed Vehicle OEM
6 CA Certificate [M] to verify [K] as described in Sections [16.2.9](#) and [16.2.12](#), respectively, before
7 using [K] to verify Digital Key Creation Data [L].

8 **16.2.11 [L] – Digital Key Creation Data**

9 The Digital Key Creation Data [L] (not shown in Figure 16-1) is described in Figure 6-5.

10 The authorized public keys are the root of trust for key sharing.

11 Authorized Public Key #1: Vehicle OEM CA PK (mandatory)

12 This is used as the root for the key sharing validation chain. It cannot be updated once the Digital
13 Key is created.

14 **16.2.12 [M] – Vehicle OEM CA Certificate (signed by Device OEM)**

15 The device may use the Device OEM signed Vehicle OEM CA Certificate [M] to validate the
16 authenticity of the vehicle public key (i.e., Vehicle.PK) contained in the Vehicle Public Key
17 Certificate [K]. The root of the validation chain is the Device OEM CA Certificate [D], which is
18 trusted by the device. The public key of the Device OEM is used to validate the signature of [M].

19 **16.2.13 [N] – Instance CA Attestation (self-signed) per Vehicle OEM**

20 This attestation applies to Variant 2.

21 [N] is generated when a new Instance CA is created in the Digital Key applet. [N] contains the
22 public key of the Instance CA in the Digital Key applet and is self-signed by the Instance CA
23 using its private key, instanceCA.SK.

24 [N] is retrieved and verified by the Device OEM Server, and then the Device OEM Server issues
25 a new Instance CA Certificate [E] by signing the InstanceCA.PK with the DeviceOEM.SK.

26 When implementing Variant 2, the SE root of trust is ensured by the following mechanisms:

- 27 • The Digital Key applet generates the Instance CA Attestation [N].
- 28 • The Device OEM Server opens a secure channel with the Digital Key applet to retrieve
29 [N].
- 30 • One of the SCPs defined by GlobalPlatform (e.g., SCP03) shall be used. This secure
31 channel shall be set to provide, at a minimum, integrity and data origin authentication on
32 the APDU responses (i.e., requesting R-MAC with or without R-ENCRYPTION in
33 parameter “i” when SCP03 is used).
- 34 • The Digital Key applet relies on its security domain for the implementation of the secure
35 channel.
- 36 • The secure channel keys used by the security domain (i.e., AES keys when SCP03 is
37 used) are specific to a given SE and are trusted by the Device OEM Server.

- 1 • When the Device OEM Server analyzes the response of the Digital Key applet, a
2 successful verification of the R-MAC ensures that the data received over this secure
3 channel has been generated by the expected Digital Key applet in the SE.
4 Figure 16-2 depicts the adaptation of the certificate chain model for Variant 2.
5 After successful verification of the Instance CA Attestation [N], the Instance CA Certificate [E]
6 is issued by the Device OEM Server.

7 **16.3 Certificate Size Restrictions**

8 This section defines X.509 certificate size restrictions to ensure compatibility with embedded
9 system (memory) constraints. Maximum certificate sizes are chosen to allow the addition of
10 proprietary, non-critical extensions in future certificate versions.

11 **External CA Certificate, issued by Vehicle OEM**

12 The maximum size of the DER-encoded certificate as per RFC 5280 [3] is 650 bytes.

13 The length of commonName (CN) fields for subject and issuer is limited to 30 bytes,
14 independently of the chosen value type.

15 **Instance CA Certificate**

16 The maximum size of the DER-encoded certificate as per RFC 5280 [3] is 650 bytes.

17 The length of commonName (CN) fields for issuer is limited to 30 bytes independently of the
18 chosen value type.

19 *Note:* The subject CN field size is limited by the TLV definition in section “endpoint
20 configuration” of the applet description (see Table 15-13).

21 **Endpoint Certificate**

22 The certificate shall include all authorized public keys that the vehicle provides.

23 In this version of the specification, the owner endpoint certificate shall be created with exactly
24 one authorized public key (enforced by car); the friend endpoint certificate shall be created with
25 exactly zero authorized public keys (enforced by owner phone DK framework).

26 The applet allows for up to five authorized public keys per endpoint; usage of more than one
27 authorized public key at the secure element level is kept open for future use cases.

28 The maximum size, of the DER-encoded certificate as per RFC5280 [3], shall be 650 bytes when
29 holding one authorized public key. The maximum size shall be 970 bytes when holding five
30 authorized public keys.

31 *Note:* The subject CN field size is limited by the TLV definition in section “endpoint
32 configuration” of the applet description (see Table 15-13).

34 **Encryption Key Attestation**

35 The size of this attestation (not an X.509 certificate) is invariable.

16.4 Device Certificate Chain

- 2 The root of trust is moved from the SE Root to the Device OEM. The Device OEM then trusts
3 and attests to signatures done by the on-device Instance CA.
4 → attest/verify
5 ⇒ trust
6 Instance CA chains:
7 SE Root CA Certificate → Instance CA Attestation
8 Device OEM CA ⇒ SE Root CA
9 Device OEM CA ⇒ Instance CA
10 Device OEM CA Certificate → Instance CA Certificate

16.5 Vehicle Certificate Chain

- 12 The vehicle public key is attested to by a Vehicle OEM CA Certificate in the device OS. This
13 certificate is trusted by the Device OEM.
14 → attest /verify
15 ⇒ trust
16 Device OEM CA ⇒ Vehicle OEM CA
17 Device OEM embeds Vehicle OEM CA Certificate in device
18 Vehicle OEM CA Certificate → Vehicle Public Key Certificate

16.6 Supported Verification Chain

- 20 The following chain shall be supported by Digital Key applet for key sharing:
21 Authorized PK → External CA certificate → Instance CA Certificate→ endpoint

16.7 Owner Pairing Certificate Chain

- 23 At owner pairing, the vehicle verifies the authenticity and the correctness of the key created in
24 the SE. The key properties, including the device public key, are attested to by the Digital Key
25 certification chain, which is rooted in the vehicle.

16.7.1 Device and Vehicle

- 27 → attest /verify
28 ⇒ trust
29 Vehicle OEM CA ⇒ Device OEM CA
30 Device OEM CA ⇒ Instance CA Certificate
31 Vehicle OEM CA Certificate → Device OEM CA Certificate → Instance CA Certificate→
32 Digital Key Certificate

16.8 Key Sharing Certificate Chain

- 2 Before issuing the Digital Key to the friend, the owner device verifies the authenticity of the
- 3 public key created in the friend device. The friend device may be a different device brand than
- 4 the owner device.
- 5 It is assumed that the vehicle has provided to the owner SE the Vehicle OEM CA PK, stored
- 6 safely in the authorized_PK structure within the Digital Key Creation Data [L]. For verification,
- 7 the friend device provides the Instance CA Certificate [E] signed by the Device OEM and the
- 8 friend Device OEM CA Certificate [F] signed by the Vehicle OEM.
9 → attest /verify
- 10 ⇒ trust
- 11 The owner device verifies:
12 Vehicle OEM CA PK → friend Device OEM CA Certificate → friend Instance CA Certificate
13 → friend Digital Key Certificate
- 14 The owner device signs the friend Digital Key Certificate using the owner's private key. The
- 15 vehicle verifies the owner signature before accepting the friend's public key when presented by
- 16 the friend device or online through the Vehicle OEM Server:
- 17 Owner PK→ friend Digital Key Certificate

17 SERVER-TO-SERVER COMMUNICATIONS AND API

17.1 Introduction

3 This section describes the API specification for communications between Device OEM Server
4 and Vehicle OEM Server.
5 Request and response bodies shall be formatted as JSON. The communication protocol used for
6 all interfaces shall be HTTPS. All strings shall be UTF-8 encoded (Unicode Normalization Form
7 C (NFC)).

17.2 API Design

- 9 • All APIs shall be implemented as HTTPS POST requests unless otherwise specified.
- 10 • Request and response payloads shall consist of simple JSON based data structures that
allow for future expansion. The main data types used are array, dictionary, boolean, and
11 string.
- 12 • Error codes are specified at three levels:
 - 13 ○ Transport level error codes
 - 14 ■ HTTP error code as described in Section 17.9
 - 15 ○ Standardized application layer error codes
 - 16 ■ Status code field in common response header; values defined in Section
17 17.9.
 - 18 ○ Detailed error code
 - 19 ■ Optional Sub Status code field in common response header as described in
20 Section [17.10](#)

17.3 Security

23 Client application shall be hosted on user's devices. Device OEM Server and Vehicle OEM
24 Server shall be hosted in secure data centers.
25 Server APIs shall be supported only over https. Sensitive data elements, where applicable, shall
26 be protected with additional encryption protocols.
27 Server APIs shall be supported only over https with mutual authentication, i.e., 2-way TLS.

17.4 URL Scheme

29 All server services URIs shall conform to the following scheme:

30 scheme://host:port/{ service }/{ version }/apiName

31 where:

- 32 • **scheme** = https
- 33 • **host:port** = Host name and Port number for the endpoint.
- 34 • **service** = Target application that hosts the API endpoint.

- 1 • **version** = Version number of the schema. Current major version is v1.
- 2 • **apiName** = Name of the API being called
- 3 An API version shall be included in the URI for all interfaces. The first supported version is v1.
- 4 The version shall be incremented by 1 for major API changes or backward incompatible
- 5 iterations on existing APIs.

6 **17.5 Healthcheck**

7 The healthcheck is a way to determine server availability of Device OEM and Vehicle OEM
8 Servers.

9 **17.6 User Interface**

10 **17.6.1 Description**

11 User interfaces are created and controlled by Device OEM entities. Each Device OEM can
12 customize the user experience as per its platform.

13 The Vehicle OEM provides all necessary visual elements for a specific vehicle/model to the
14 Device OEM in an offline process, prior to going live. Details of the visual elements are out of
15 scope of the specification and should be negotiated between the Vehicle OEMs and Device
16 OEMs. An identifier, which can be chosen by the Vehicle OEM, is assigned for UI elements.
17 This identifier is called uiIdentifier.

18 To ensure uniqueness across Vehicle OEMs, a prefix may be used. The uiIdentifier should be
19 generic and the system should not be able to identify owner or friend using uiIdentifier.

20 An example of an uiIdentifier for a specific vehicle: “VehicleOEMNameModelYearColor.” The
21 UI elements are stored on the Device OEM Server to ensure the best availability at runtime.

22 During different types of provisioning (owner pairing, key sharing), the uiIdentifier is provided
23 to the Device OEM Server by the Vehicle OEM Server. The Device OEM may then select the
24 predefined UI elements. It is the responsibility of the Device OEM to display the UI elements
25 appropriate to device-specific parameters.

26 **17.7 Server APIs**

27 **17.7.1 API - Track Key**

28 This is a generic API offered by the Vehicle OEM Server to track a Digital Key. This API should
29 have idempotent behavior. If trackKey is called for a keyID that is already registered, the API
30 call shall be successful and return with a usual response. This API is used in owner pairing and
31 key sharing as described in Section [6.3.4.4](#) and Section [11.8.5](#), respectively.

32 **17.7.1.1 End Point**

33 HTTPS POST /{service}/{version}/trackKey

1 17.7.1.2 *trackKey()*

Parameter	Type	Max Length (bytes)	Description	Required	Domain Version
encryptionCertChain	List of String	4096	Base 64 encoded array of X509 certificates in DER format representing a certificate chain for encrypting uiBundle from Vehicle OEM Server to Device OEM Server. Encryption of trackKeyResponse() is performed using the first certificate in the list. This is a chain of trust established between vehicle and Device OEMs and is out of scope for this specification.	Mandatory	DS-VS
encryptionVersion	String	32	Defines the version of encryption to be used for encryption from Vehicle OEM Server to Device OEM Server. Value is ECIES_v1	Mandatory	DS-VS
keyID	String	128	The unique identifier for a key. Computed as the 160-bit SHA-1 hash of the value of the bit string subjectPublicKey from the keyData (excluding the tag, length, and number of unused bits).	Mandatory	V-OD-FW
keyType	Enum (see Section 17.8.3)	32	OWNER or SHARED. See Section 17.8.3	Mandatory	V-OD-FW
deviceType	Enum	32	Type of device on which key is being provisioned.	Mandatory	D-VS
accountIdHash	String	128	Hash of a Device OEM account id. For the same Device OEM account, accountIdHash should be the same throughout the life of the account.	Mandatory	DS-VS
keyData	Encrypted-DataContainer (see Section 17.8.4)	4096	Encrypted from Device to Vehicle OEM Server. Encrypted using VehicleOEM.Enc.PK as defined in Section 14 . Refer to Section 6.3.4.4 (see Table 6-2) and Section 11.10 (see Table 11-19)	Mandatory	Encryption: DS-VS Data: V-OD-FW, D-VS
vehicleOwnerDeviceFrameworkVersionList (V-OD-FW-versionList)	String	512	Contains vehicle owner device framework version V-OD-FW list with V-OD-FW-agreedVersion first	Conditional: Present if request sent by owner device	D-VS
deviceVehicleServerVersionList (D-VS-deviceList)	String	512	Contains the sorted list of all versions the device supports for communication with the vehicle OEM server (D-VS)	Mandatory	D-VS

1 17.7.1.3 *trackKeyResponse ()*

Parameter	Type	Max Length (bytes)	Description	Required	Domain Version
responseHeader	Response-Header Object	1024	The common response header	Mandatory	DS-VS
uiBundle	EncryptedData-Container (see Section 17.8.4)	4096	Specified in Section 17.8.5 . The encrypted uiBundle, encrypted from Vehicle OEM Server to Device OEM Server	Mandatory	Encryption: DS-VS Data: V-OD-FW
Vehicle-Mobilization-Data	EncryptedData-Container (see Section 17.8.4)	4096	Encrypted from Vehicle OEM Server to device. Encrypted using Device.Enc.PK as defined in Section 14 . Refer to Unencrypted vehicleMobilizationData (see Section 17.8.15)	Conditional. Required if immobilizer token or slot identifier or both are retrieved online.	Encryption: DS-VS Data: V-OD-FW
eventType	enum	64	SHARED_KEY_ADDED from eventType	Conditional. Required only if a tracked owner key has shared keys associated with it, and if owner migrates to new device and tracks new owner key for the vehicle.	Encryption: DS-VS
eventData	Map	2048	eventData used for SHARED_KEY_ADDED event.	Conditional. Required only if a tracked owner key has shared keys associated with it and if owner migrates to new device and tracks new owner key for the vehicle.	V-OD-FW
brand	String	32	The brand of the vehicle.	Mandatory	D-VS
model	String	32	The model of the vehicle.	Mandatory	D-VS
serverSupportedVersions (D-VS-serverList)	String	512	Contains the sorted list of all versions the vehicle OEM server supports for communication with the device (D-VS). First version is the agreed	Mandatory	D-VS

Parameter	Type	Max Length (bytes)	Description	Required	Domain Version
			version selected by the server, then all other versions are listed in descending order		

- 1 All version lists shall be coded as hex strings, identical to tags 5Ah, 5Bh and 5Ch in sections 5.1.1 and 5.1.2.
- 2 All version lists shall be sorted in descending order. Where indicated, the first version is the agreed version, then the remaining list shall be sorted in descending order.

5 17.7.1.4 Sample trackKey

```
6 HTTP POST /{service}/{version}/trackKey
7 x-requestId: 80BCCEF8F59127A6D1BB5CFEB534D95
8 x-device-oemId: device OEM A
9 {
10     "encryptionCertChain"      : [ "MIIBx...pglH", "nMIIBx...pglH" ],
11     "encryptionVersion"        : "ECIES_v1",
12     "keyID"                   : "394858302023",
13     "keyType"                 : "OWNER | SHARED",
14     "deviceType"               : "PHONE",
15     "accountIDHash"           :
16     "C8645830202EFEB53427A6D75F15C85E78A5195307E2351858349AB9",
17     "keyData"                 :
18         "version"              : "ECIES_v1",
19         "ephemeralPublicKey"   :
20         "04613197827d91806d630bc4adff44686b012316eb03825f2d6587ffd58d32f4522ada80cc9
21         3679e1a316dc0729ebf8172fd41f0c0c1bdda01126f1a6186b2a008",
22         "publicKeyHash"       :
23         "d6e3bbc95bf11d533677a1aa052d7caef29fd76a02e43e3609579966168d7d19",
24         "data"                  :
25         "5GRH7/ecb85HFstAlxn3Idet7ARtfFZn2AuMft1p...IkccchjFBLJTGm9qvJtxK/3EdWob+iFS9
26         FwHeKSjq8MxdNgQ1rj4fq6CzQfY8O9",
27         "vehicleOwnerDeviceFrameworkVersionList" : "02000100",
28         "deviceVehicleServerVersionList"       : "02000201"
29     }
30
31 }
```

1 17.7.1.5 *Sample trackKeyResponse*

```
2    HTTP 200
3    x-responseId : 80BCCEF8F59127A6D1BB5CFEB534D95
4
5    {
6     "responseHeader" : {
7       "statusCode" : "200"
8     },
9     "uiBundle" : {
10       "version" : "ECIES_v1",
11       "ephemeralPublicKey" :
12         "806d6306b04613197827d91012316eb03bc4adff4468825f2d6587fada80cc93679e1af58d
13         32f4522172fd41f0c0c1b1126f1316dc0729ebf8a6186b2a008dda0",
14         "publicKeyHash" :
15         "1d533677a1aa052dd6e3bbc95bf17cae02e4f29fd76a8d7d193e360957996616",
16         "data" :
17         "1xn3IdeT5b85HFstA7ARtfFZGRH7/ecn2AuMft1p...K/3EdWob+iFS9m9qvJtxCzQfY8O9FwHe
18         KSjq8MxdNgQ1rj4fq6IkccbjFBLJTG"
19     }
20     "eventType" : "SHARED_KEY_ADDED",
21     "eventData" : {
22       "sharedKeys" : "3",
23       "shareableKeys" : "7",
24       "keyValidTo" : "2021-02-19T23:05:17.462Z",
25       "keyValidFrom" : "2019-02-19T23:05:17.462Z",
26       "reason" : "shared key is added",
27       "encryptedData" : {
28         "version" : "ECIES_v1",
29         "ephemeralPublicKey" :
30         "098234098234750893476372832742373468923482357508932832742373468923482347637
31         283274237346892348235",
32         "publicKeyHash" :
33         "50982347502373468923482385098234750237346892394346892348234763728327",
34         "data" :
35         "ydST1ZG5Q3cfzfM1qjjwpmOmWQHta...SBBiIAGK3PUbdE2IF/MmPefY9bsH"
36       },
37       },
38       "brand" : "xyz",
39       "model" : "abc"
40       "vehicleMobilizationData" : {
```

```
1     "version"          : "ECIES_v1",
2     "ephemeralPublicKey":
3     "098234098234750893476372832742373468923482357508932832742373468923482347637
4     283274237346892348235",
5     "publicKeyHash"    :
6     "50982347502373468923482385098234750237346892394346892348234763728327",
7     "data"             :
8     "ydST1ZG5Q3cflzfM1qjjwpmOmWQHta...SBBiIAGK3PUbdE2IF/MmPefY9bsH"
9     },
10    "serverSupportedVersions" : "02000100"
11 }
```

12 Sample of Unencrypted uiBundle data

```
13 {
14     "ktsSignature"      :
15     "3045022017B5FBBCF2F475F15C85E78A519530792211CC4DA50EC0BBDC42C7C28FA780E022
16     100D20D47E8624C851473602A02BFCD4F4C0DB02C2FA1A4A1E56A9530019CE289D4",
17     "keyValidFrom"     : "2019-03-06T18:00:49.239Z",
18     "keyValidTo"       : "2022-03-05T18:00:49.239Z",
19     "sharedKeys"       : "3",
20     "shareableKeys"   : "7",
21     "uiIdentifier"    : "35E3406EFDA511E88EB2F2801F1B9FD1",
22     "sharingPasswordRequired" : false,
23     "entitlement"     : {
24         "value"           : 0,
25         "description"    : "full",
26         "longDescription" : "full access and drive capabilities"
27     },
28     "supportedEntitlements" : {
29         "entitlements" : [
30             {
31                 "value"           : 0,
32                 "description"    : "full",
33                 "longDescription" : "full access and drive capabilities"
34             },
35             {
36                 "value"           : 1,
37                 "description"    : "accessOnly",
38                 "longDescription" : "only vehicle access , no drive
39                 capability"
```

```
1      },
2      {
3          "value"          : 2,
4          "description"    : "accessAndDriveRestricted",
5          "longDescription" : "access and drive with restrictions"
6      },
7      {
8          "value"          : 3,
9          "description"    : "vehicleDelivery",
10         "longDescription" : "vehicle delivery profile"
11     },
12     {
13         "value"          : 4,
14         "description"    : "valet",
15         "longDescription" : "valet parking key"
16     },
17     {
18         "value"          : 5,
19         "description"    : "vehicleService",
20         "longDescription" : "service key"
21     },
22     {
23         "value"          : 6,
24         "description"    : "custom entitlement description",
25         "longDescription" : "custom entitlement long description"
26     }
27 ]
28 }
29 }
30 }
```

31 Sample of Unencrypted vehicleMobilizationData (See section [17.8.15](#))

```
32 {
33     "slotIdentifier"      : "11223344",
34     "confidentialMailboxData" : "3ZCVBjxM2 ... Cwk265HB+sB9",
35     "ephemeralPublicKey"   : "
36 04F98CCA31651AD2E63266144B2450FD6081D8FEA8CEB826E1FB10E8034E932446CAD19D2010
37 62DD1C7CB0BB293BF16A4BEFB2ED500977E7197E01F26906E39B5F"
38 }
```

1 Note: Please refer to Section [17.11](#) for encryption scheme and other details.

2 **17.7.2 Manage Key**

3 This is a generic API offered by the Device OEM Server and Vehicle OEM Server to manage the
4 lifecycle of a Digital Key. This API is used in key sharing and key termination as described in
5 Section [11](#) and Section [13](#), respectively.

6 **17.7.2.1 End Point**

7 HTTP POST /{service}/{version}/manageKey

8 **17.7.2.2 manageKey()**

Parameter	Type	Max Length (bytes)	Description	Required	Domain Version
keyID	String	128	The unique identifier for a key.	Mandatory	V-OD-FW
action	enum	128	Action of manage key (See Section 17.8.7).	Mandatory	DS-VS
termination-Attestation	String	4096	Termination attestation of a key when owner Digital Key or shared Digital Key is terminated on own device. Signed by DigitalKeyx.SK, as defined in Section 14 .	Conditional: provided by owner/friend device whenever possible	D-VS
deviceRemote-Termination-Request	EncryptedData-Container (see Section 17.8.4)	4096	Remote termination request for termination of a remote key when shared key is terminated from owner device. Signed by DigitalKeyx.SK as defined in Section 14 .	Conditional: provided by owner device	D-VS
serverRemote-Termination-Request	String (see Section 14.2)	4096	Remote termination request for termination of a remote key when owner key or shared key is terminated from Vehicle OEM Server. Signed by	Conditional: provided by KTS	D-VS

Parameter	Type	Max Length (bytes)	Description	Required	Domain Version
			VehicleOEM.Sig.SK as defined in Section 14 .		
vehicleOEM-Proprietary-Data	String	4096	Subsection of the private mailbox of a Digital Key containing the Vehicle OEM proprietary data when owner or shared key is terminated on the device (local or remote termination).	Conditional: Provided by the device whenever possible	n/a

1 *17.7.2.3 manageKeyResponse()*

Parameter	Type	Max Length (bytes)	Description	Required	Domain Version
Response-Header	Response-Header Object	1024	The common response header	Mandatory	DS-VS
Encrypted-Data	Encrypted-Data-Container (See Section 17.8.4)	4096	Decrypted contents format is specified in Section 17.8.11. Used to send the latest status of all shared keys to owner device. When this response is a result of owner deleting shared key, the decrypted value shall contain the respective shared key with a status for InTermination	Conditional. Required if manageKey called with deviceRemote-TerminationRequest or if requested action is SUSPEND or RESUME.	D-VS

2 *17.7.2.4 Sample manageKeyRequest for owner/friend deleting own key*

```

3   HTTP POST /{service}/{version}/manageKey
4   x-requestId : 80BCCEF8F59127A6D1BB5CFEB534D95
5   x-device-oemId: Device OEM A
6   {
7     "keyID"          : "4938205039594",
8     "action"         : "TERMINATE",
9     "terminationAttestation" :
10    "5CFEB534D8F5CEFA6D1B780BCB5CFEB534D95912CEFA6D1B780BCB5CFE"

```

```
1   "vehicleOEMProprietaryData" : "5CEFA6D1B780BCB5CFEB534D"  
2 }
```

3 *Note:* For terminationAttestation data structure, see Section [15.3.2.5](#), Table 15-18, and Table
4 15-19.

5 *17.7.2.5 Sample manageKeyRequest for owner deleting Shared Key*

```
6 HTTP POST /{service}/{version}/manageKey  
7 x-requestId: 80CCEF8F59127A6D1BB5CFEB534D95  
8 x-device-oemId: device OEM A  
9 {  
10    "keyID" : "4938205039594",  
11    "action" : "TERMINATE",  
12    "deviceRemoteTerminationRequest" : {  
13      "version" : "ECIES_v1",  
14      "ephemeralPublicKey" :  
15      "16eb03825f2d97827d91806d630bc4adff44686587ffd58d32f4522ada80cc01126f1a6186b  
16      2a00893679e1abf8172fd41f0c0c1bdda3b0046131123616dc0729e",  
17      "publicKeyHash" :  
18      "677a1aa052d7caef29d6e3bbc95bf11d43e3609579966168d7d533fd76a02e19",  
19      "data" :  
20      "kE8/Rs9sNo4olc6s3Qej7mXhAtDPcp ...1Bm/r7bKNX6m365vnIDEDc"  
21    }  
22 }
```

23 *Note:* For deviceRemoteTerminationRequest data structure, see Section [14.2](#).

24 *17.7.2.6 Sample manageKeyRequest for vehicle deleting Shared Key*

```
25 HTTP POST /{service}/{version}/manageKey  
26 x-requestId: 80CCEF8F59127A6D1BB5CFEB534D95  
27 x-device-oemId: device OEM A  
28 {  
29    "keyID" : "4938205039594",  
30    "action" : "TERMINATE",  
31    "serverRemoteTerminationRequest":  
32    "3AEBF536D2F5CEFA6D1C780BCA9CEBF534E90734CEFA2D1B780ACB8EBC"  
33 }
```

34 *Note:* For serverRemoteTerminationRequest data structure, see Section [14.2](#).

1 17.7.2.7 *Sample manageKeyResponse for Owner/Friend deleting own key*

```
2    HTTP 200
3    x-responseId : 80BCCEF8F59127A6D1BB5CFEB534D95
4
5    {
6       "responseHeader" : {
7           "statusCode" : "200"
8       }
9    }
```

10 17.7.2.8 *Sample manageKeyResponse for Owner deleting shared key*

```
11    HTTP 200
12    x-responseId : 80BCCEF8F59127A6D1BB5CFEB534D95
13
14    {
15       "responseHeader" : {
16           "statusCode" : "200"
17       }
18       "encryptedData" : {
19           "version" : "ECIES_v1",
20           "ephemeralPublicKey" :
21              "098234098234750893476372832742373468923482357508932832742373468923482347637
22              283274237346892348235",
23              "publicKeyHash" :
24              "50982347502373468923482385098234750237346892394346892348234763728327",
25              "data" :
26              "ydST1ZG5Q3cflzfM1qjjwpmOmWQHta...SBBiIAGK3PUbdE2IF/MmPefY9bsH"
27       }
28    }
```

29 Sample unencrypted encryptedData value, specified at Unencrypted Event Notification Data
30 (See Section [17.8.11](#))

```
31    {
32       "keyID" : "483729239475C864E4F",
33       "version" : "1.0",
34       "sharedKeysData" : [
35           {
36              "keyID" : "292348379464E4F75C8",
37              "keyConfiguration" : {
```

```
1      "friendlyName" : "Jon's Phone",
2      "rights"       : 0,
3      "keyValidFrom" : "2021-02-19T23:05:17.462+0000",
4      "keyValidTo"   : "2021-02-19T23:05:17.462+0000"
5    },
6      "status"        : 1,
7      "groupIdentifier" : "E24A5978-66BB-4810-A29B-E4B193D5076F",
8      "deviceType"    : "PHONE"
9  },
10  {
11    "keyID"          : "498277430125B8F85F1",
12    "keyConfiguration" : {
13      "friendlyName" : "Jon's Watch",
14      "rights"       : 0,
15      "keyValidFrom" : "2021-02-19T23:05:17.462+0000",
16      "keyValidTo"   : "2021-02-19T23:05:17.462+0000"
17    },
18    "status"        : 1,
19    "groupIdentifier" : "E24A5978-66BB-4810-A29B-E4B193D5076F",
20    "deviceType"    : "WATCH"
21  },
22  {
23    "keyID"          : "276459836197C6A34D2",
24    "keyConfiguration" : {
25      "friendlyName" : "Jane's Phone",
26      "rights"       : 0,
27      "keyValidFrom" : "2021-02-19T23:05:17.462+0000",
28      "keyValidTo"   : "2021-02-19T23:05:17.462+0000"
29    },
30    "status"        : 15,
31    "groupIdentifier" : "D308745E-35E3-4FCD-B121-D159457B79C6",
32    "deviceType"    : "PHONE"
33  }
34 ]
35 }
```

36 17.7.3 API – eventNotification

37 This is a generic API offered by the Device OEM Server and the Vehicle OEM Server to
38 communicate different events on a Digital Key.

- 1 *17.7.3.1 End Point*
- 2 HTTP POST /{service}/{version}/eventNotification
- 3 *17.7.3.2 EventNotification ()*

Parameter	Type	Max Length (bytes)	Description	Required
keyID	String	128	The receiver's unique key identifier.	Conditional
eventType	enum	64	The type of the event. Specified in Section 17.8.9 .	Mandatory
EventData	Map	2048	Contextual based on event type. Specified in Section 17.8.10 .	Mandatory

- 4 *17.7.3.3 EventNotificationResponse ()*

Parameter	Type	Max Length (bytes)	Description	Required
response-Header	Response-Header	1024	The common response header	Mandatory

- 5 *17.7.3.4 Sample eventNotificationRequest to send owner key information to owner Device OEM Server*

```
7   HTTP POST /partner/v1/eventNotification
8   x-requestId : 80BCCEF8F59127A6D1BB5CFEB534D95
9   x-vehicle-oemId: Vehicle OEM A
10  {
11    "keyID"          : "483729239475C864E4F",
12    "eventType"      : "SUBSCRIPTION_CHANGED",
13    "EventData"      : {
14      "sharedKeys"     : "3",
15      "shareableKeys" : "7",
16      "keyValidTo"    : "2021-02-19T23:05:17.462Z",
17      "keyValidFrom"  : "2019-02-19T23:05:17.462Z",
18      "reason"        : "key subscription is changed"
19    }
20  }
```

1 17.7.3.5 *Sample eventNotificationRequest to send shared key information to owner Device OEM*
2 *Server*

```
3    HTTP POST /partner/v1/eventNotification
4    x-requestId    : 80BCCEF8F59127A6D1BB5CFEB534D95
5    x-vehicle-oemId : Vehicle OEM A
6    {
7       "keyID"          : "483729239475C864E4F",
8       "eventType"      : "SHARED_KEY_ADDED",
9       "eventData"       :
10          "sharedKeys"     : "3",
11          "shareableKeys"   : "7",
12          "keyValidTo"      : "2021-02-19T23:05:17.462Z",
13          "keyValidFrom"    : "2019-02-19T23:05:17.462Z",
14          "reason"         : "shared key is added",
15          "encryptedData"   :
16              "version"        : "ECIES_v1",
17              "ephemeralPublicKey" :
18              "098234098234750893476372832742373468923482347637
19              283274237346892348235",
20              "publicKeyHash"    :
21              "50982347502373468923482385098234750237346892394346892348234763728327",
22              "data"           :
23              "ydST1ZG5Q3cflzfM1qjjwpmOmWQHta...SBBiIAGK3PUbdE2IF/MmPefY9bsH"
24         }
25     }
26 }
```

27 Sample of unencrypted encryptedData value, specified at Unencrypted Event Notification Data
28 (See Section [17.8.11](#)):

```
29 {
30     "keyID": "483729239475C864E4F",
31     "version": "1",
32     "sharedKeysData": [
33       {
34         "keyID"          : "292348379464E4F75C8",
35         "keyConfiguration" : {
36              "friendlyName" : "Jon's Phone",
37              "rights"       : 0,
38              "keyValidFrom" : "2021-02-19T23:05:17.462Z",
39              "keyValidTo"   : "2021-02-19T23:05:17.462Z"
```

```
1             },
2         "status" : 1
3         "groupIdentifier" : "E24A5978-66BB-4810-A29B-
4 E4B193D5076F",
5         "deviceType" : "PHONE"
6     },
7
8     {
9         "keyID" : "379292464E4F75C8348379",
10        "keyConfiguration" : {
11            "friendlyName" : "Jane's Phone",
12            "rights" : 0,
13            "keyValidFrom" : "2021-02-19T23:05:17.462Z",
14            "keyValidTo" : "2021-02-19T23:05:17.462Z"
15        },
16        "status" : 6
17        "groupIdentifier" : "D308745E-35E3-4FCD-B121-
18 D159457B79C6",
19        "deviceType" : "PHONE"
20    }
21 ]
22 }
```

23 17.7.3.6 Sample eventNotificationRequest to friend Device OEM Server

```
24 HTTP POST /partner/v1/eventNotification
25 x-requestId : 80BCCEF8F59127A6D1BB5CFEB534D95
26 x-vehicle-oemId: Vehicle OEM A
27 {
28     "keyID" : "483729239475C864E4F",
29     "eventType" : "IN_TERMINATION",
30     "eventData" : {
31         "keyValidTo" : "2021-02-19T23:05:17.462Z",
32         "keyValidFrom" : "2019-02-19T23:05:17.462Z",
33         "reason" : "shared key being terminated"
34     }
35 }
```

36 17.7.3.7 Sample eventNotificationRequest to send model field to Owner or Friend Device Server

```
37 HTTP POST /partner/v1/eventNotification
```

```
1 x-requestId : 80BCCEF8F59127A6D1BB5CFEB534D95
2 x-vehicle-oemId: Vehicle OEM Name
3 {
4     "keyID"      : "483729239475C864E4F",
5     "eventType"  : "UI_ELEMENTS_UPDATED",
6     "eventData"  : {
7         "keyValidTo"   : "2021-02-19T23:05:17.462+0000",
8         "keyValidFrom" : "2019-02-19T23:05:17.462+0000",
9         "reason"       : "Model is updated",
10        "uiElements"   : {
11            "model" : "Model Name"
12        }
13    }
14 }
```

15 *17.7.3.8 Sample eventNotificationResponse*

```
16 HTTP 200
17 x-responseId : 80BCCEF8F59127A6D1BB5CFEB534D95
18 {
19     "responseHeader" : {
20         "statusCode" : "200"
21     }
22 }
```

23 **17.7.4 API – Healthcheck**

24 This is a generic API offered by the Device OEM Server and the Vehicle OEM Server to
25 determine the availability of the corresponding server. This API does not require any headers.

26 *17.7.4.1 End Point*

27 HTTP GET /{service}/healthcheck

28 *17.7.4.2 Input*

29 N/A

30 *17.7.4.3 Output*

31 N/A

32 *17.7.4.4 Sample healthcheckRequest*

33 HTTP GET /partner/healthcheck

1 17.7.4.5 *Sample healthcheckResponse*

2 HTTP 200

3 17.7.5 *API – versionUpdate*

4 This API is used for the exchange and update of vehicle and device versions.

5 Upon an update (either device or vehicle), Device, Vehicle and Servers determine the highest
6 common supported version (D-VS, V-OD-FW) for subsequent communication.

7 If after the update of a vehicle SW, the list of supported versions affecting the device has been
8 modified, this API should be called by the vehicle OEM server. In this case, the vehicle OEM
9 determines the timing of when this API is called.

10 If after the update of a device SW, the list of supported versions affecting the vehicle or vehicle
11 server has been modified, this API should be called by the device OEM server. In this case, the
12 device OEM determines the timing of when this API is called.

13 Note that due to the asynchronous call chain (vehicle or device might be offline) the version
14 agreement occurs on both sides without providing the agreed version back to the caller.

15 All version lists within JSON shall coded as hex strings, identical to tags 5Ah and 5Bh as outlined
16 in Sections 5.1.1 and 5.1.2.

17 The following sections lists all scenarios in which the API is used and provides the list of
18 parameters to be included.

19 17.7.5.1 *Call Scenario: Vehicle Software Update impacting V-OD-FW*

20 - Called by VS for every owner key for the vehicle at the called Device OEM server. Refer
21 to Figure 2-7.

22 - Call parameters

- 23 o *keyID*
- 24 o *deviceVehicleServerVersionList* (D-VS-serverList, should not change)
- 25 o *vehicleOwnerDeviceFrameworkVehicleVersionList* (V-OD-FW-vehicleList)

26 - Response parameters

- 27 o success/failure

28 Listing 17-1: Content of *versionUpdate* API on Vehicle Software Update

```
1    {  
2     "updateScenario": "VEHICLE_SW_UPDATE",  
3     "keyID": "3ED46B38D85BCC0A1F31C9BAC509116BC09CC3D2",  
4     "vehicleOwnerDeviceFrameworkVehicleVersionList": "02000201",  
5     "deviceVehicleServerVersionList": "02000201"  
6   }
```

29 17.7.5.2 *Call Scenario: Device SW update*

30 - Called by DS (device OEM server) for every key on the updated device at the called
31 Vehicle OEM server

- 1 - Parameters
 - 2 o *keyID*
 - 3 o *deviceVehicleServerVersionList* (D-VS-deviceList)
 - 4 o *vehicleOwnerDeviceFrameworkDeviceVersionList* (V-OD-FW-deviceList)
- 5 - Response parameters
 - 6 o success/failure

7 Refer to Figure 2-5

8

9 *Listing 17-2: Content of versionUpdate API after Device Software Update*

```
1 {  
2   "updateScenario": "DEVICE_SW_UPDATE",  
3   "keyID": "3ED46B38D85BCC0A1F31C9BAC509116BC09CC3D2",  
4   "deviceVehicleServerVersionList": "02000201",  
5   "vehicleOwnerDeviceFrameworkDeviceVersionList": "02000201"  
6 }
```

10

11 17.7.5.3 End Point

12 HTTP POST /{service}/{version}/versionUpdate

13 17.7.5.4 *versionUpdate()*

14 The caller of this API can provide a keyID and associated version lists as described in individual
15 call scenarios above. An example of a full HTTP request is provided below

```
16    HTTP POST /{service}/{version}/versionUpdate  
17    x-requestId: 80BCCEF8F59127A6D1BB5CFEB534D95  
18    x-device-oemId: device OEM A  
19    {  
20        "updateScenario": "DEVICE_SW_UPDATE",  
21        "keyID": "3ED46B38D85BCC0A1F31C9BAC509116BC09CC3D2",  
22        "deviceVehicleServerVersionList": "2021",  
23        "vehicleOwnerDeviceFrameworkDeviceVersionList": "202224"  
24    }
```

25

26

Parameter	Type	Max Length (bytes)	Description	Required
updateScenario	String	64	Identifies the scenario from the above list. Possible identifiers are: VEHICLE_SW_UPDATE (17.7.5.1), DEVICE_SW_UPDATE (17.7.5.2)	Mandatory
keyID	String	128	keyID for which the version information apply.	Conditional. To be used in scenarios 17.7.5.1, 17.7.5.2
deviceVehicleServer VersionList	String	512	Contains the sorted list of all versions the device/vehicle supports for communication with the vehicle/device server (D-VS, D-VS).	Conditional. Present for DEVICE_SW_UPDATE (17.7.5.2) and VEHICLE_SW_UPDATE(17.7.5.1)
vehicleOwnerDevice FrameworkVehicleVersionList	String	512	Contains the sorted list of all versions the vehicle supports for communication with the device (V-OD-FW).	Conditional. Present for VEHICLE_SW_UPDATE (17.7.5.1), for owner device only
vehicleOwnerDevice FrameworkDeviceVersionList	String	512	Contains the sorted list of all versions the device supports for communication with the vehicle (V-OD-FW).	Conditional. Present for DEVICE_SW_UPDATE (17.7.5.2), for owner device only

1

2 *17.7.5.5 versionUpdateResponse()*

3 versionUpdateResponse does not contain version information.

4 Response body example:

```

5   HTTP 200
6   x-responseId      : 80BCCEF8F59127A6D1BB5CFEB534D95
7   {
8     "responseHeader" : {
9       "statusCode"    : "200"
10      }
11    }
```

17.8 Structure Definitions

17.8.1 Request Headers

All requests by the Vehicle OEM Server and Device OEM Server server shall contain HTTP request headers as defined below

value	Description
x-requestId	All requests by Device OEM Servers and Vehicle OEM Servers shall have an HTTP header “x-requestId”. The value shall be a UUID of length 36 containing hyphens.
x-timestamp	All requests by Device OEM Servers and Vehicle OEM Servers shall have an HTTP header “x-timestamp”. Timestamp should be specified in Unix timestamp. The value shall be given in milliseconds. The value shall not be modified in case of request retry. This is used to identify outdated requests in case of retry.
x-device-oemId	All requests by Device OEM Servers shall have an HTTP header “x-device-oemId”. Value is the name of the Device OEM.
x-vehicle-oemId	All requests by Vehicle OEM Servers shall have an HTTP header “x-vehicle-oemId”. Value is the Vehicle OEM identifier (See Table 2-1 of [35]).
x-responseId	The corresponding response to the API shall have HTTP header “x-responseId”, which should echo the value of “requestId” in the request header. This is used to identify the request associated to the response for a particular API request and response pair. The x-responseId shall be sent by Device OEM Server and Vehicle OEM Server.
x-device-oem-host	The FQDN of the Device OEM’s endpoint. Vehicle OEM uses this information to initiate communication to Device OEM for a specific keyID. The “x-device-oem-host” shall be sent by the Device OEM Server
x-user-identifier	Device OEM server may include a HTTP header “x-user-identifier”. Value is <ul style="list-style-type: none">• generated by the device OEM server and stored by Vehicle OEM servers• a AES-GCM encrypted base64 encoded string representation of a hash value generated using a pseudonymized device side userId.• up to 128 bytes long If included by Device OEM server, then all server API messages transmitted to device OEM server by Vehicle OEM server shall include the value.

17.8.2 Response Headers

All responses by the Vehicle OEM Server and Device OEM Server shall contain header data of type “ResponseHeader” within the payload. Contents keys are defined below

Parameter	Type	Max Length (bytes)	Description	Required
statusCode	String	64	Status code used to indicate an error, or “200” for success.	Mandatory
subStatus-Code	String	64	Conveys failure codes from downstream entities or for more granular conveyance of specific error conditions.	Optional
Status-Message	String	256	Not parsed programmatically. Example: “Downstream system offline”	Optional

1 17.8.3 Key Type

Key	Type	Max Length (bytes)	Description	Required
OWNER	String	32	owner key type	N/A
SHARED	String	32	shared key type	N/A

2 17.8.4 EncryptedDataContainer

Parameter	Type	Max Length (bytes)	Description	Required
version	String	10	Identifier for the algorithm specified in this document. Example: “ECIES_v1”.	Mandatory
Ephemeral-PublicKey	String	Variable	Hex-encoded sender's ephemeral public key. This should be from an ephemeral key pair generated for a single message. Format for encoding the sender public key: concat(0x04, x, y) (This will be 65 bytes total for a key generated based on named curve secp256r1 - 1.2.840.10045.3.1.7)	Mandatory
Public-KeyHash	String	Variable	Hex-encoded recipients' key agreement public key fingerprint. Fingerprint generation algorithm: sha256Hash(toUncompressedRawECPublicKey-Format(recipientKAPublicKey)). SHA-256 hash of the uncompressed key agreement public key (concat(0x04, x, y)) of the receiving system used in ECIES_v1 Key Agreement	Mandatory

Parameter	Type	Max Length (bytes)	Description	Required
data	String	Variable	Base64-encoded encrypted data. data=base64Encode(encrypt(unencryptedData, symmetricEncryptionParams))	Mandatory

1 17.8.5 Unencrypted uiBundle

Parameter	Type	Max Length (bytes)	Description	Required
kts-Signature	String	256	Signature created by key tracking server (KTS); see Section 6.3.4.4 .	Mandatory
uiIdentifier	String	32	Identifier referencing visual elements that are required to create user interface for Digital Key. The definition and encoding of the string value is agreed between vehicle and device OEMs individually and is out of scope of this specification.	Mandatory
sharing-Password-Required	boolean	16	Indicates if sharing password is required	Conditional. Required if activationRequired is not present.
activationRequired	boolean	16	Indicates that a 2 nd factor is required to activate a shared key	Optional. Transmitted in trackKeyResponse
activationOptions	See Section 17.8.20	N/A	Describes vehicle OEM-defined activation options for an inactive shared key	Conditional: Required if vehicle supports at least an activation option.
approvedSharingMethods	See Section 17.8.21	N/A	List of SharingMethodGroups accepted by vehicle OEM as secure	Optional, only for owner keys

Parameter	Type	Max Length (bytes)	Description	Required
			sharing methods that do not require 2 nd factor activation	
entitlement	See Section 17.8.14	N/A	Describes entitlement supported for the key being tracked, maps to the “profiles” field as described in Section 11.6 , Table 11-5.	Mandatory
supported-Entitlements	See Section 17.8.16	N/A	Describes list of all the vehicle supported entitlements. Entitlement in the supportedEntitlements list maps to the “profiles” field as described in Section 11.6 , Table 11-5	Conditional. Required for owner keys
keyValid-From	String	32	Key validity start date. Time should be specified in UTC ISO-8601 Format yyyy-MM-dd'T'HH:mm:ss.SSSZ.	Required for shared key Optional for owner key
keyValidTo	String	32	Key validity end date. Time should be specified in UTC ISO-8601 Format yyyy-MM-dd'T'HH:mm:ss.SSSZ.	Required for shared key Optional for owner key
sharedKeys	String	4	Number of keys shared by the owner.	Conditional. Required for owner keys
Shareable-Keys	String	4	Number of keys available for sharing.	Conditional. Required for owner keys

1 17.8.6 Digital Key Status

Status ID	Status	Type	Max Length (bytes)	Description	Required
1	Active	Integer	32	Key is active and is usable.	N/A
2	Personalizing	Integer	32	Key is in personalizing state and is unusable.	N/A

Status ID	Status	Type	Max Length (bytes)	Description	Required
6	Suspend: By Issuer	Integer	32	Key is suspended by issuer (Vehicle OEM Server) and is unusable.	N/A
7	Suspend: Lost Mode	Integer	32	Key is suspended due to lost mode and is unusable.	N/A
10	Terminated: By Issuer	Integer	32	Key is terminated by issuer (Vehicle OEM Server) and is unusable.	N/A
14	InActive	Integer	32	Key is inactive	N/A
15	InTermination	Integer	32	Key is being terminated.	N/A

1 17.8.7 Action for manageKey

Action	Type	Max Length (bytes)	Description	Required
TERMINATE	enum	32	Terminate the key	N/A
SUSPEND ¹	enum	32	Suspend the key	N/A
RESUME ¹	enum	32	Resume the key	N/A

¹: SUSPEND and RESUME actions are only applicable in manageKey API calls from Device OEM server to Vehicle OEM Server.

2 17.8.8 Key Actions

- 3 All key actions can be used as Event Types of Notification (see Section [17.8.9](#)).

Name	Description	Reason for key action to be triggered
CREATE_SHARED_KEY	Create shared key for cross-platform key sharing	When owner sends keyCreationRequest for friend device, this notification is sent from Vehicle OEM Server to friend Device OEM Server.
SIGN_SHARED_KEY	Sign shared key for cross-platform key sharing	When friend sends keySigningRequest for owner device, this notification is sent from Vehicle OEM Server to owner Device OEM Server.

IMPORT_SHARED_KEY	Import shared key for cross-platform key sharing	When owner sends keySigningRequest for friend device, this notification is sent from Vehicle OEM Server to friend Device OEM Server.
--------------------------	--	--

1 17.8.9 Event Type of Notification

Name	Description	Reason for the event to be triggered
IN_TERMINATION	Key is being terminated	When owner or shared key is being terminated from local/remote source, this notification is sent from the Vehicle OEM Server to owner or friend Device OEM Server. Digital Key is not terminated yet.
SUSPENDED	Key is suspended	When the vehicle suspends owner or shared key, this notification is sent from the Vehicle OEM Server to owner or friend Device OEM Server. Digital Key is suspended.
RESUMED	Key is resumed	When the vehicle resumes the owner or shared key, this notification is sent from the Vehicle OEM Server to owner or friend Device OEM Server. Digital Key is resumed.
SHARED_KEY_-IN_TERMINATION	Shared key is being terminated	When shared key is being terminated from local/remote source, this notification is sent from the Vehicle OEM Server to the owner Device OEM Server. Digital Key is not terminated yet.
SHARED_KEY_-TERMINATED	Shared key is terminated	When shared key is terminated (after fade-out period), this notification is sent from the Vehicle OEM Server to the owner Device OEM Server. Digital Key is terminated.
SHARED_KEY_-SUSPENDED	Shared key is suspended	When the vehicle suspends a shared key, this notification is sent from the Vehicle OEM Server to the owner Device OEM Server. Shared key is suspended.
SHARED_KEY_-RESUMED	Shared key is resumed	When the vehicle resumes a shared key, this notification is sent from the Vehicle OEM Server to the owner Device OEM Server. Shared key is resumed.
RESUMING	Digital Key is being resumed	When the vehicle is resuming all access, this notification is sent to all devices with this Digital Key for the vehicle. Digital Key is not yet resumed on the vehicle.

Name	Description	Reason for the event to be triggered
SHARED_KEY_-ADDED	Shared key is tracked and activated in device	When a shared key has been successfully tracked in the vehicle OEM server and activated on the device (key is in status “active”), this notification is sent from the Vehicle OEM Server to the owner Device OEM Server.
SUBSCRIPTION_-CHANGED	Key subscription changed	When Digital Key service subscription changes in Vehicle OEM Server, this notification is sent from the Vehicle OEM Server to owner or friend Device OEM Server. This may occur, e.g., when the Digital Key service is renewed.
SHARING_-PASSWORD_-REQUIRED	Password is required for key sharing	When the policy for the sharing password changes in Vehicle OEM Server, this notification is sent from the Vehicle OEM Server to owner or friend Device OEM Server.
SHARING_POLICY_-CHANGED	Sharing policy for shared key required	When the policy for the approved sharing methods changes in Vehicle OEM Server, this notification is sent from the Vehicle OEM Server to owner Device OEM Server (if no proprietary method is used).
ACTIVATION_OPTIONS_CHANGED	Activation options for vehicle have changed	When the available activation options change in the vehicle (e.g. SW update), this notification is sent from the Vehicle OEM Server to owner Device OEM Server.
ENTITLEMENTS_-UPDATED	Entitlements are updated	When entitlements supported by the vehicle are updated in the Vehicle OEM Server and vehicle. These entitlements are then available to the owner for future key sharing operations; existing shared keys are not affected.
UI_ELEMENTS_-UPDATED	UI elements are updated	When UI elements such as model is updated in Vehicle OEM Server and propagated to device for existing Digital Key.
VEHICLE_-ATTESTATION	Vehicle Attestation Notification	When vehicle transmits an attestation to the owner device in case a friend enters a correct sharing password in the vehicle. Implementation of this event type is optional for device OEM.

1 17.8.10 Event Data of Event Notification

Name	Type	Description	Required
status	Integer	State of the Digital Key (see Section 17.8.6)	Conditional. Only if the digital key already exists (e.g., not during CREATE_SHARED_KEY)
keyValidFrom	String	Key validity start date. Time should be specified in UTC ISO-8601 Format yyyy-MM-dd'T'HH:mm:ss.SSSZ.	Conditional. Only if the digital key already exists (e.g., not during CREATE_SHARED_KEY)
keyValidTo	String	Key validity end date. Time should be specified in UTC ISO-8601 Format yyyy-MM-dd'T'HH:mm:ss.SSSZ.	Conditional. Only if the digital key already exists (e.g., not during CREATE_SHARED_KEY)
reason	String	Reason that triggered event. For informational purposes only; not displayed in UI.	Mandatory
uiElements	UIElements (see Section 17.8.18)	Updated UI elements	Conditional. Required if eventType is UI_ELEMENTS_UPDATED
sharedKeys	String	Number of keys shared by the owner	Conditional. Required if eventNotification is sent to owner Device OEM Server and if it contains shared key information
shareableKeys	String	Number of keys available for sharing	Conditional. Required if eventNotification is sent to owner Device OEM Server and if it contains shared key information
resume-Attestation	String	Resume attestation for resuming a Digital Key as defined in Section 13.4.3	Conditional. Required only for eventNotification, eventType RESUMING
encryptedData	Encrypted-Data-Container (see Section 17.8.4)	Used to send shared key information to owner device. Encrypted from Vehicle OEM Server to Device. Encrypted using Devx.Enc.PK" as defined in Section 14.1 .	Conditional. Required if eventNotification is sent to owner Device OEM Server and if it contains shared key information

Name	Type	Description	Required
supported-Entitlements	See Section 17.8.16	Describes list of all the vehicle supported entitlements	Conditional. Required if entitlements are updated in Vehicle OEM Server and vehicle
activationOptions	See Section 17.8.19	Describes activation options for an inactive shared key	Conditional: Required for ACTIVATION_OPTIONS_CHANGED event
approvedSharingMethods	See Section 17.8.21	List of SharingMethods accepted by car OEM as approved sharing method	Conditional: Required for SHARING_POLICY_CHANGE D event

1 17.8.11 Unencrypted Event Notification Data

Parameter	Type	Max Length (bytes)	Description	Required
keyID	String	128	The unique identifier for an owner key	Mandatory
Shared KeysData	List of Shared Key (see Section 17.8.12)	N/A	List of all associated shared keys	Mandatory
vehicle-Attestation	String	N/A	For a description of the TLV encoded attestation, refer to Table 11-22	Conditional. Required if vehicle OEM server supports this optional feature as described in Section 11.11 and is sending VEHICLE_ATTESTATION notification when a friend has entered the correct sharing password in the vehicle.

1 17.8.12 Shared Key

Parameter	Type	Max Length (bytes)	Description	Required
keyID	String	128	The unique identifier for a shared key	Mandatory
Key-Configuration	See Section 17.8.13	4096	Shared key configuration	Mandatory
status	Integer	16	Status of a shared key (see Section 17.8.6)	Mandatory
groupIdentifier	String	4096	UUID per friend per given vehicle, generated by Vehicle OEM. This field is a unique identifier over (vehicle, accountIdHash).	Mandatory
deviceType	enum	32	Type of device on which key is being provisioned.	Mandatory

2 17.8.13 Key Configuration

Key	Type	Max Length	Description	Required
friendlyName	String	30	The friendly friend's name to be displayed to the user	Mandatory
rights	Integer	16	Integer indicating which access profile the shared key has; see Table 11-5.	Mandatory
keyValidFrom	String	32	Key validity start date. Time should be specified in UTC ISO-8601 Format yyyy-MM-dd'T'HH:mm:ss.SSSZ.	Mandatory
keyValidTo	String	32	Key validity end date. Time should be specified in UTC ISO-8601 Format yyyy-MM-dd'T'HH:mm:ss.SSSZ.	Mandatory

3 17.8.14 Entitlement

4 Entitlement is also referred to as access profile as described in Section [11.6](#), Table 11-5.

Name	Type	Description	Required
value	Integer	Value of the access profile of the shared key; see Table 11-5.	Mandatory
description	String	A short English description used to look up localized description strings for this Digital Key on the device, e.g., accessOnly.	Mandatory
long-Description	String	A long form description of the entitlement. This should be a one-sentence description of the entitlement (aka access profile), e.g., only vehicle access, no drive capability.	Mandatory

1 17.8.15 Unencrypted vehicleMobilizationData Fields

Parameter	Type	Max Length (bytes)	Description	Required
slotIdentifier	String	8	The slot identifier to assign for the key Table 6-3 and Table 11-20	Conditional. Required if slot identifier is retrieved online
confidential-MailboxData	String	Variable	Encrypted confidential mailbox data (see Table 6-3 and Table 11-20)	Conditional. Required if immobilizer token is retrieved online
ephemeral-PublicKey	String	Variable	Ephemeral encryption public key of the Vehicle OEM server (see Table 6-3 and Table 11-20)	Conditional. Required if immobilizer token is retrieved online

2 17.8.16 Supported Entitlements

Name	Type	Description	Required
entitlements	List of entitlement	Describes the list of all vehicle supported entitlements	Mandatory

1 17.8.17 *Device Type*

Key	Type	Max Length (bytes)	Description	Required
PHONE	String	32	phone	N/A
WATCH	String	32	watch	N/A
OTHER	String	32	other	N/A

2 17.8.18 *UI Elements*

Name	Type	Description	Required
model	String	The vehicle model description to be updated.	Yes

3 17.8.19 *Activation Options*

4 Activation options should be indicated on owner and friend device UI during the key sharing
5 process. The friend can then choose any supported activation option.

6 Activation options are an extension and should be used instead of the “sharing password
7 required” indication in the uiBundle (see section 17.8.6).

8

Name	Type	Description	Required
activationOptionsList	List of strings	Provides the list of all supported 2nd factor activation options for activation of a shared key (17.8.21)	Conditional: Required only if more options than the if sharing password and/or other activation options are supported by the vehicle
sharingPasswordLength	String	String representation of Vehicle OEM specific SHARING_PASSWORD_LENGTH as per Appendix B.1	Optional: To be included if SHARING_PASSWORD_LENGTH needs to be updated

9

10 The following activation options may be supported by the vehicle. Device UIs shall support all
11 listed options.

Activation options	Description	Required
sharingPasswordActivation	Friend key can be activated by entering the sharing password in the vehicle UI	Optional
ownerKeyActivation	Friend key can be activated by presenting the owner key to the vehicle when the friend key is used for the first time	Optional
friendKeyActivation	Friend key can be activated by presenting another active friend key to the vehicle when the friend key is used for the first time	Optional
keyFobActivation	Friend key can be activated by presenting the key fob to the vehicle when the friend key is used for the first time	Optional

- 1
- 2 The text to describe each activation option in the UI is defined by the device OEM.
- 3 *17.8.20 Supported approved sharing methods*

Name	Type	Description	Required
approvedSharingMethods	List of strings	Describes the list of all approved sharing method group identifiers as described in Table 11-11 (list of string encoded hex values of tags 61h, without tag and length of the subtags, e.g., ["4002000041020001","4002000041020001"])	Optional

4 17.9 HTTP Status Codes

Status Code	Description	Caller should retry
200	Request was received and processed	No
301	A resource is moved permanently, retry on new location	Yes
302	A resource is moved temporarily, retry on new location	Yes
400	Bad Request	No
404	Malformed or unknown URI	No

Status Code	Description	Caller should retry
500	Internal server error	Yes
503	Service unavailable	Yes

- 1 The default “Caller should retry” policy of a given status code is shown in this section, which
2 shall be followed unless otherwise specified with a sub-status code described in Section [17.10](#).

3 **17.10 Sub Status Codes**

Status Code	Description	HTTP Status Code	Caller should retry	Note
50000	Unexpected error	500	Yes	Shall not be used unless a more appropriate sub-status code has not been defined
50001	Downstream systems unavailable	500	Yes	Shall be used when systems on callee side are temporarily unavailable for unknown reasons and a later retry will likely succeed. There will be no immediate retries
50002	OEM scheduled downtime	503	No	Shall be used when systems on callee side are temporarily unavailable for unknown reasons and a later retry will likely succeed. There will be no immediate retries
50009	No associated user	404	Yes	Shall be used when there is no user account registered.
50110	Unknown key id supplied	404	No	Shall be used when the referred key id is not known (or has already been deleted) on callee side. If possible a more detailed sub-status code shall be used to refer to either owner or friend key id.
50111	Unknown owner Digital Key identifier supplied	404	No	Shall be used when the owner key id is not known (or has already been deleted) on callee side
50112	Unknown Digital Key identifier of a friend supplied	404	No	Shall be used when the friend's key id is not known (or has already been deleted) on callee side

Status Code	Description	HTTP Status Code	Caller should retry	Note
50113	Maximum sharing key limit reached	500	No	Shall be used when the received key tracking request refers to a key that would exceed the vehicle's capacity of storing keys (or hit a business limit). This error should be avoided by restricting the number of shared keys to the allowed limit on device side.
50114	Cryptography error occurred	400/ 500	No	Shall be used when any cryptographic element (signature) in the supplied data is wrong.
50115	Feature not supported	500	No	Shall be used when the caller data refers to a feature that is not supported on callee side
50116	Conflicting Key Slot Identifier	400	N/A	Shall be used when there is a mismatch between the key id and the expected slot identifier in the trackKey request (e.g., slot identifier was used with another key id before)
50117	Version not supported	400/ 500	No	Shall be used when the version sent in the request is not supported by the recipient
50119	Invalid or expired Sharee Instance CA Certificate	400	No	Shall be used when Instance CA Certificate from Sharee has expired or is invalid
50120	Invalid or expired Device OEM Root Certificate	400	No	Shall be used when root certificate of the Device OEM has expired or is invalid
50121	Invalid or expired Vehicle OEM Root Certificate	400	No	Shall be used when root certificate of the Vehicle OEM has expired or is invalid
50122	Owner Pairing not possible: vehicle is not ready	400	No	Shall be used in case the vehicle is unmapped from account after owner pairing but before key tracking

Status Code	Description	HTTP Status Code	Caller should retry	Note
50123	Friend Tracking failed: Owner signature not as expected	400	No	Shall be used when owner signature is invalid
50124	Friend tracking failed: No owner key found or in invalid state	400	No	Shall be used when owner key is in invalid state

- 1 Wherever the specification allows multiple HTTP Status Codes, caller (e.g., Vehicle OEM)
2 server should choose the HTTP Status Code that fits the specific error (5xx for errors within
3 Vehicle OEM server, 4xx for errors on caller's side, e.g., Device OEM server).
- 4 Status code 50000 Unexpected error - shall not be used unless a more appropriate sub-status
5 code has not been defined. In this case concerned parties are encouraged to file Change Requests
6 against this specification to add a dedicated sub-status code for the new error and not use
7 "unexpected error" anymore. Callers shall accept even unknown sub-status codes and map them
8 to "unexpected error" internally to avoid the use of versioning to introduce new error codes.
- 9 The caller shall retry any request satisfying one or more of the following criteria:
- 10 • Received an HTTP status code that allows a retry (see table above)
- 11 • Request timed out or otherwise failed at the network level such that no HTTP response
12 was received or processed
- 13 Requests may be retried a maximum of three times. Retry attempts shall adhere to the following
14 guidelines:
- 15 • First retry shall occur a pseudorandom amount of time Δt after the original attempt,
16 where Δt is between 0s–1s inclusive.
- 17 • Second retry shall occur a pseudorandom amount of time Δt after the first retry attempt,
18 where Δt is between 1s–5s inclusive.
- 19 • Third retry shall occur a pseudorandom amount of time Δt after the second retry attempt,
20 where Δt is between 5s–30s inclusive.

21 **17.11 ECC Encryption for Device and Server**

22 This section defines the specifics of the ECIES_v1 algorithm for ECDHE encryption and
23 decryption.

24 **17.11.1 Frame/Packet Contents**

25 This following information shall be included in the frame/packet sent from the
26 encrypting/sending party to the decrypting/receiving party:

- 27 1. Algorithm Identifier

28 The identifier for the algorithm specified is "ECIES_v1".

1 2. Sender Key Agreement Public Key

2 The sender's key agreement public key shall be from an ephemeral key pair generated for
3 a single message.

4 The format for encoding the sender public key: concat(0x04, x, y)

5 (This will be 65 bytes total for a key generated based on named curve secp256r1 and its
6 OID value 1.2.840.10045.3.1.7.

7 3. Recipient Key Agreement Public Key Fingerprint

8 The recipient's key agreement public key fingerprint. The fingerprint shall be generated
9 using algorithm: SHA-256 hash of the uncompressed key agreement public key (i.e.,
10 concat(0x04, x, y)) of the receiving party used in ECIES_v1 Key Agreement.

11 *fingerprint = sha256Hash(toUncompressedRawECPublicKeyFormat(recipientKAPublicKey));*

12 4. Encrypted Data

13 The binary encrypted data.

14 **17.11.2 Encryption Process**

15 The encrypting system shall follow the steps below to encrypt the data:

16 1. Generate sender's ephemeral EC keypair on named curve ephemeral for the message.
17 secp256r1 and its OID 1.2.840.10045.3.1.7.

18 2. Using recipient's encryption public key and the ephemeral private key, generate the
19 shared secret using Elliptic Curve Diffie-Hellman key agreement algorithm with OID
20 1.3.132.1.12.

21 3. Derive the symmetric key as described in Key Derivation Function (see Section [17.11.4](#))

22 4. Derive the initialization vector for symmetric encryption as described in initialization
23 vector derivation (see Section [17.11.6](#)).

24 5. Encrypt data using all 128 bits of the derived symmetric key using AES-128 id-aes128-
25 GCM with its OID 2.16.840.1.101.3.4.1.6 with no padding, the initialization vector, and
26 no associated authentication data. The 16-byte GCM authentication tag shall be appended
27 to the cipher text.

28 **17.11.3 Decryption Process**

29 Receiving system shall follow the steps below to decrypt the encrypted data:

30 1. Validate that the sender's ephemeral public key is valid on the named curve secp256r1
31 and its OID 1.2.840.10045.3.1.7.

32 2. Using receiving system's encryption private key (identified by recipient fingerprint) and
33 the sender's ephemeral public key, generate the shared secret using Elliptic Curve Diffie-
34 Hellman key agreement algorithm with OID 1.3.132.1.12.

35 3. Derive the symmetric key as described in Key Derivation Function (see Section [17.11.4](#)).

36 4. Derive the initialization vector for symmetric encryption as described in Initialization
37 vector derivation (See Section [17.11.6](#)).

- 1 5. Decrypt the data using AES-128 id-aes128-GCM with its OID 2.16.840.1.101.3.4.1.6
2 with no padding, the initialization vector, and no associated authentication data. The 16-
3 byte GCM authentication tag shall be appended to the cipher text.

4 **17.11.4 Key Derivation Function**

5 This subsection defines the algorithm for deriving the symmetric key from a shared secret and
6 other parameters.

7 Derive the keying material according to the algorithm defined in Section 3.6 of [22], using the
8 ANSI X9.63 Key Derivation Function described in Section 3.6.1 of [22].

```
9       z = sharedSecretFromKeyAgreement(myECPrivateKey, otherPartyEC PublicKey);  
10      sharedInfo = computeSharedInfo();  
11      counter = 0x00000001; // 4 bytes  
12      keyingMaterial = Hash(z | counter | sharedInfo);
```

13 where

14 *senderKeyAgreementPrivateKey* is generated in item 2 of Section [17.11.1](#)

15 *recipientKeyAgreementPublicKey* is defined in item 3 of Section [17.11.1](#).

16 *computeSharedInfo* is defined in Section [17.11.4.1](#).

17 with the following parameters:

Input name	Value
H (Hash function)	SHA-256
keydatalen (Length of symmetric key)	16
Z (Shared secret)	The shared secret calculated using the ECDH key agreement keys
SharedInfo	The shared information containing the Shared Info sequence

18 **17.11.4.1 Shared Info**

Input name	Required	Value
Sender Key Agreement Public Key	Mandatory	The senders' ephemeral key agreement public key. Format: toUncompressedRawEC PublicKeyFormat(sender.ephemeralKeyAgreementPublicKey)
Recipient Key Agreement Public Key	Mandatory	The recipient's key agreement public key Format: toUncompressedRawEC PublicKeyFormat(recipient.keyAgreementPublicKey)

19 **17.11.5 Symmetric Encryption Key**

20 The first 16 bytes of the 32-byte keying material output from the KDF shall be used as the
21 symmetric key for encryption.

1 **17.11.6 Initialization Vector Derivation**

2 The last 16 bytes of the 32-byte keying material output from the KDF shall be used as IV.

3 **17.11.7 EC Public Key Point Encoding in Uncompressed Form**

4 To encode the EC public key point in uncompressed form, follow Elliptic-Curve-Point-to-Octet-
5 String Conversion in Section 2.3.3 of [\[22\]](#).

6 **17.12 Versioning**

7 **17.12.1 General**

8 Server APIs send and receive data that can originate from another server or from a device
9 connected to the other server (if other server is device OEM server).

10 A new column in the data description tables of the APIs indicate the domain version that a data
11 element depends on.

12 The negotiation of the API version used between the Device OEM Server and the Vehicle OEM
13 Server (DS-VS) is outside of this specification.

14 **17.13 Example**

15 This section shows the step-by-step output of the encryption of a sample message using sample
16 sender and recipient keys.

17 **17.13.1 Keys**

Party	Key Type	Value
Sender	Key Agreement Key	
	Private Key Hex (S)	53994c02ca9f6d1afbda1742f43d3c17dedfaf3367727208fe9c cc50a33824a0
	Public Key Hex (0x04 x y)	0474542541492424edc34f33ba94e61bf718f33ca393d1bf081 6f156e4a26687 3f59564aad1a95c9fd8971b527171784e390d9137d037fe2ae3 0490b1ed1d73aa3
Recipient	Key Agreement Key	
	Private Key (S)	6c300cac339b4c7084e34175e2e6f5e1d1b135d158bd578c2b 1af870facaeef17
	Public Key Hex (0x04 x y)	04ad2d126c3f8f85bf5796f6ab849b57a35133fed491eced025 4ed6a1169d928 1f98018f1aa71d222874d72f47b0b28d84dacb8801b96e815c 06f1151210cc7090
	Public Key Fingerprint Hex	ac0095a2121357c69d64213137973222d9873bc80dbcc6ea04 4d6db3ec5fbcb

1 **17.13.2 Message to encrypt**

2 { "id": "Hello", "value": "World" }

3 **17.13.3 Encryption process**

Data	Format	Value
Algorithm Identifier	String	ECIES_v1
Derived Shared Secret	Hex	a6c3021dc18ad03959768250e872818585264fbdd1b6de6ed32a7eb16f19f858
KDF Shared Info	Hex	0474542541492424edc34f33ba94e61bf718f33ca393d1bf0816f156e4a26687 3f59564aad1a95c9fd8971b527171784e390d9137d037fe2ae30490b1ed1d73a a304ad2d126c3f8f85bf5796f6ab849b57a35133fed491eced0254ed6a1169d9 281f98018f1aa71d222874d72f47b0b28d84dacb8801b96e815c06f1151210cc 7090
Keying Material	Hex	58a5f8b53ff010a62eb2e98303f7d2d088b35fa8b758797777dbafa81df3bc9f
Derived AES Key	Hex	58a5f8b53ff010a62eb2e98303f7d2d0
Initialization Vector	Hex	88b35fa8b758797777dbafa81df3bc9f
Encrypted Data	Hex	46e9fbefb564e5eba68c094da08172bac8299bd268749763224dea8a59313d548 963dafbc3dcc7ea206646edeb4469e5d6eed38

4 **17.13.4 Decryption process**

Data	Format	Value
Algorithm Identifier	String	ECIES_v1
Derived Shared Secret	Hex	a6c3021dc18ad03959768250e872818585264fbdd1b6de6ed32a7eb16f19f858
KDF Shared Info	Hex	0474542541492424edc34f33ba94e61bf718f33ca393d1bf0816f156e4a266873f59564aad1a95c9fd8971b527171784e390d9137d037fe2ae30490b1ed1d73a a304ad2d126c3f8f85bf5796f6ab849b57a35133fed491eced0254ed6a1169d9 281f98018f1aa71d222874d72f47b0b28d84dacb8801b96e815c06f1151210cc 7090
Keying Material	Hex	58a5f8b53ff010a62eb2e98303f7d2d088b35fa8b758797777dbafa81df3bc9f
Derived AES Key	Hex	58a5f8b53ff010a62eb2e98303f7d2d0
Initialization Vector	Hex	88b35fa8b758797777dbafa81df3bc9f

Decrypted Data	String	{ "id": "Hello", "value": "World" }
----------------	--------	-------------------------------------

18 SECURITY

18.1 SPAKE2+ Protocol Description

18.1.1 General

This section explains the principles of the SPAKE2+ protocol. Refer to Section [18.4](#) for implementation details.

The system uses SPAKE2+, which is an ECC-based pairing algorithm protocol, to mutually authenticate two entities based on the knowledge of a password provided on the client side (i.e., device) and a verifier, permanently stored in the server (i.e., vehicle). For more detailed information, see [\[10\]](#).

The NIST P-256 curve shall be used [\[8\]](#).

The Vehicle OEM Server should generate the password and provision the necessary elements (w_0, L ; see [18.1.2](#)) into the vehicle well before start of the owner pairing, so that pairing is possible even when the vehicle is offline at the moment of owner pairing.

The Scrypt key derivation is executed in the server and on the device, which allows the server and device to adapt the derivation \leftrightarrow parameters over time to counter increases in attacker performance.

The password pwd should be provided to the owner through the Vehicle OEM account, protected by the login credentials known only by the owner. The password is UTF-8 encoded and should be passed into the scrypt function in this coding.

All values are assumed to be in big-endian byte order. The x and y random generator shall have uniform distribution over the required range and be cryptographically secure.

18.1.2 Execution

The Vehicle OEM Server derives L and w_0 from the password as follows:

$z_0 = \text{left 40 bytes of Scrypt}(pwd, s, N_{\text{scrypt}}, r, p, dkLen)$

$z_1 = \text{right 40 bytes of Scrypt}(pwd, s, N_{\text{scrypt}}, r, p, dkLen)$

It shall derive w_0 and w_1 using the method FIPS 186-4, B.5.1 Per-Message Secret Number Generation Using Extra Random Bits:

$w_0 = (z_0 \bmod (n-1)) + 1$ with n being the order n of base point G as defined for NIST P-256

$w_1 = (z_1 \bmod (n-1)) + 1$ with n being the order n of base point G as defined for NIST P-256

w_0 and w_1 shall not equal.

Then, it shall calculate:

$L = w_1 \times G$

with G as defined for the chosen elliptic curve.

The scrypt algorithm is described in [\[11\]](#).

1 The Scrypt function is defined with the following parameters:

- 2 • **Salt**: 16 bytes, randomly generated for each new verifier as per [\[12\]](#)
- 3 • **Cost parameter Nscrypt**: 4096 or higher
- 4 • **Block size r**: 8
- 5 • **Parallelization parameter p**: 1
- 6 • **Output length dkLen**: 80

7 The Vehicle OEM Server then provides salt, L, and w0, which constitute the verifier, securely
8 and confidentially to the vehicle.

9 This specification does not specify the means to upgrade these parameters to faster hardware in
10 the future. A simple solution would be for the server to provide these parameters to the vehicle
11 too.

12 When the pairing starts or re-starts after an erroneous attempt, the vehicle shall randomly
13 generate a valid scalar y on the chosen curve, and then calculate:

14
$$Y = y \times G + w0 \times N$$

15 where N is a point on the used elliptic curve as specified in Section [18.1.5](#). When the maximum
16 number of failed pairing attempts is reached (see the usage of SPAKE2+ REQUEST command
17 in Section 5.1.2), a new password and verifier shall be generated to allow to retry pairing.

18 Y shall not be the point at infinity and shall be on the chosen curve. A new y shall be generated,
19 and a new Y shall be calculated until the requirements for Y are fulfilled before continuing the
20 protocol.

21 The vehicle shall transmit Y to the device together with the Scrypt configuration parameters.

22 When owner pairing is started, the password is provided to the device to be paired, which allows
23 it to calculate:

24
$$z0 = \text{left 40 bytes of Scrypt}(pwd, s, Nscrypt, r, p, dkLen)$$

25
$$z1 = \text{right 40 bytes of Scrypt}(pwd, s, Nscrypt, r, p, dkLen)$$

26 using s, Nscrypt, r, and p transmitted by the vehicle. dkLen is implicitly known on both sides.

27 It shall derive w0 and w1 using the method FIPS 186-4, B.5.1 Per-Message Secret Number
28 Generation Using Extra Random Bits:

29
$$w0 = (z0 \bmod (n-1)) + 1 \text{ with } n \text{ being the order } n \text{ of base point } G \text{ as defined for NIST}$$

30 P-256

31
$$w1 = (z1 \bmod (n-1)) + 1 \text{ with } n \text{ being the order } n \text{ of base point } G \text{ as defined for NIST}$$

32 P-256

33 Then the device shall randomly generate a valid scalar x on the chosen curve and then calculate:

34
$$X = x \times G + w0 \times M$$

35 where M is a point on the chosen elliptic curve as specified in Section [18.1.5](#). A new x shall be
36 generated and X shall be calculated on each new pairing attempt, within the limits of a maximum
37 number of retries until a new password needs to be generated by the server.

38 X shall not be the point at infinity and shall be on the chosen curve. A new x shall be generated,
39 and a new X shall be calculated until the requirements for X are fulfilled before continuing the
40 protocol.

1 Device and vehicle then shall exchange X and Y through the commands defined in Section 5.
2 On reception of X, the vehicle shall compute the shared secret curve points Z and V as follows:
3 $Z = y \times (X - w_0 \times M)$
4 $V = y \times L$
5 On reception of Y, the device shall compute the shared secret curve points Z and V as follows:
6 $Z = x \times (Y - w_0 \times N)$
7 $V = w_1 \times (Y - w_0 \times N)$
8 Both vehicle and device then shall calculate the shared secret K as follows:
9 $K = \text{SHA-256}(\text{len}(X) \parallel X \parallel \text{len}(Y) \parallel Y \parallel \text{len}(Z) \parallel Z \parallel \text{len}(V) \parallel V \parallel \text{len}(w_0) \parallel w_0)$
10 where $\text{len}(\text{str})$ denotes the length of a string in bytes, represented as an 8-byte little-endian
11 number.
12 The shared secrets CK and SK are derived from K:
13 $CK = K [0:128]$
14 $SK = K [128:128]$ ⁵
15 The LONG_TERM_SHARED_SECRET is derived based on Listing 18-9.

16 18.1.3 *Verification*

17 The verification of the derived keys shall be done by calculating a check value on either side
18 which are mutually exchanged and verified.

19 The vehicle shall calculate the verification value M[1] as

20 $K1 = \text{HKDF}(CK, \text{"ConfirmationKeys"} \parallel \text{TLV } 5B_h \parallel \text{TLV } 5C_h, [0:128])$, where
21 ConfirmationKeys is a defined static string (see [Listing 18-6](#)).

22 $M[1] = \text{C-MAC}(K1, X)$ using K1 as secret key, using CMAC-AES-128 as defined in
23 [RFC4493]

24 and shall send it over to the device.

25 The device shall verify M[1] using the received M[1] from the vehicle and, if successful, shall
26 calculate the verification value M[2] as

27 $K2 = \text{HKDF}(CK, \text{"ConfirmationKeys"} \parallel \text{TLV } 5B_h \parallel \text{TLV } 5C_h, [128:128])$ (see [Listing](#)
28 [18-6](#))

29 $M[2] = \text{C-MAC}(K2, Y)$ using K2 as secret key, using CMAC-AES-128 as defined in
30 [RFC4493]

31 and shall send it over to the vehicle.

32 The protocol succeeds when the vehicle successfully validates M[2].

33 18.1.4 *Summary*

34 To summarize, the following elements are part of the protocol:

- 35 • s (16 bytes): Salt generated by the Vehicle OEM Server

⁵ The notation used is [start_index: number_of_bits]

- 1 • **pwd**: Password generated by the Vehicle OEM Server, available in the owner's Vehicle
2 OEM account
- 3 • **z0** (40 bytes): Part of verifier including FIPS extra bits, derived from the password using
4 a one-way function
- 5 • **z1** (40 bytes): Value including FIPS extra bits derived from the password using a one-
6 way function
- 7 • **w0** (32 bytes): Part of verifier, derived z0
- 8 • **w1** (32 bytes): Value derived from z1
- 9 • **L**: curve point, part of verifier, calculated by the Vehicle OEM Server in uncompressed
10 format
- 11 • **x** (32 bytes): Random scalar generated by device on each pairing attempt
- 12 • **y** (32 bytes): Random scalar generated by vehicle on each pairing attempt
- 13 • **M**: Known constant curve point in uncompressed format
- 14 • **N**: Known constant curve point in uncompressed format
- 15 • **X**: Intermediate public curve point exchanged between device and vehicle in
16 uncompressed format
- 17 • **Y**: Intermediate public curve point exchanged between vehicle and device in
18 uncompressed format
- 19 • **Z**: Secret curve point generated on both sides from public values and constants in
20 uncompressed format
- 21 • **V**: Secret curve point generated on both sides from public values and constants in
22 uncompressed format
- 23 • **K** (32 bytes): Shared SPAKE2+ secret
- 24 • **CK** (16 bytes): Shared SPAKE2+ secret (as derived from K) to derive confirmation keys
- 25 • **SK** (16 bytes): Shared SPAKE2+ secret (as derived from K) to derive system keys
- 26 • **K1** (16 bytes): Vehicle-side secret key for evidence calculation M[1]
- 27 • **K2** (16 bytes): Device-side secret key for evidence calculation M[2]
- 28 • **Kenc, Kmac, Krmac**: Secure channel keys
- 29 • **LONG_TERM_SHARED_SECRET**: Long-term shared secret for further key
30 derivations

31 For all operations, all curve points shall be represented in x9.63 standard format as the byte
32 stream $0x04 \parallel \langle x \rangle \parallel \langle y \rangle$ format where $\langle x \rangle$ and $\langle y \rangle$ are each 32-byte integer in big-endian
33 representation (65 bytes).

34 x and y shall be newly generated after each failing pairing attempt.

35 18.1.5 SPAKE2+ Constant Definitions

36 The following NIST P-256 (SECP256r1) curve parameters are defined in [10]

37 M=

38 04

1 88 6e 2f 97 ac e4 6e 55 ba 9d d7 24 25 79 f2 99
2 3b 64 e1 6e f3 dc ab 95 af d4 97 33 3d 8f a1 2f
3 5f f3 55 16 3e 43 ce 22 4e 0b 0e 65 ff 02 ac 8e
4 5c 7b e0 94 19 c7 85 e0 ca 54 7d 55 a1 2e 2d 20
5
6 N=
7 04
8 d8 bb d6 c6 39 c6 29 37 b0 4d 99 7f 38 c3 77 07
9 19 c6 29 d7 01 4d 49 a2 4b 4f 98 ba a1 29 2b 49
10 07 d6 0a a6 bf ad e4 50 08 a6 36 33 7f 51 68 c6
11 4d 9b d3 60 34 80 8c d5 64 49 0b 1e 65 6e db e7

12 **18.2 Privacy Properties**

13 **18.2.1 NFC Transaction [WCC1/WCC2]**

14 The vehicle identifier is transmitted from the vehicle to the device in the AUTH0 command (see
15 Section [15.3.2.9](#)). It is recommended to change the vehicle identifier when the vehicle owner
16 changes to avoid any privacy issues.

17 **18.2.2 Vehicle OEM Server**

18 The Vehicle OEM Server shall be designed in a way to separate business-relevant data from the
19 key tracking data, which may be stored in a KTS for example.

20 Business data, such as number of purchased mobile keys, expiration of the mobile key
21 subscription, etc., may be exploited to enforce the Vehicle OEM business policy.

22 Data stored in the KTS shall be privacy protected. The data shall not be used for business
23 reasons.

24 When a vehicle is stolen (or in similar cases), only the KTS-data for the affected vehicle is to be
25 disclosed, on specific request.

26 **18.2.3 User Privacy**

27 The Vehicle OEM-controlled data going into the Digital Key instance shall be chosen and
28 protected in a way such that the privacy of device users is protected in all use cases.

29 **18.2.4 Multi-Vehicle OEM Deployment**

30 The device shall enforce that a specific Vehicle OEM application can only access Digital Key
31 data and functionality within the Digital Key applet instance that belongs to the same Vehicle
32 OEM.

1 **18.2.5 Error Handling**

2 On every attempt to execute the owner pairing flow, the device shall generate a new public key
3 pair (device.PK/device.SK).

4 **18.3 Cryptographic Algorithms**

5 All cryptographic elements that are stored in or received from the Digital Key applet shall
6 respect the formats defined in Section 4.

7

8 **18.4 Cryptographic Protocols**

9 **18.4.1 Server Password Generation**

10 The Server generates the password for vehicle and device in the following way:

11 Listing 18-1: Server Password Generation

```
1    input: Script parameters
2    output: w0, w1, L, s, pwd
3    begin
4       generate random salt s
5       generate random password pwd
6       z=Scrypt(pwd, s) with parameters as defined in Section 18.1.2
7       z0=left 40 bytes of z
8       z1=right 40 bytes of z
9       w0=(z0 mod (n-1)) + 1 with n being the order n of base point G as defined for NIST P-256
10      w1=(z1 mod (n-1)) + 1 with n being the order n of base point G as defined for NIST P-256
11      L=w1xG with G as defined for NIST P-256
12      provide pwd to device (through web interface or Vehicle OEM app)
13      send s, w0 and L securely to vehicle (through Vehicle OEM proprietary link)
14    end
```

12 **18.4.2 Vehicle-side public EC Point Generation**

13 Listing 18-2: Vehicle-side Public Point Generation

```
1    input: w0
2    output: Y, y
3    bBegin
4       generate random scalar y on chosen curve
5       Y=yxG+w0xN
6       return Y, which is sent to the device during owner pairing
7    end
```

14 **18.4.3 Device-side public EC Point Generation**

15 Listing 18-3: Device-side Public Point Generation

```
1    input: w0
2    output: X, x
3    begin
4       generate random scalar x on chosen curve
5       X=xxG+w0xM
6       return X, which is sent to the vehicle during owner pairing
```

7 | **end**

1 18.4.4 Vehicle-side computation of Shared Secret

2 *Listing 18-4: Vehicle-side Computation of Shared Secret*

```
1 input: X, w0, L, y
2 output: K
3 begin
4   Z=yx(X-w0xM)
5   V=yxL
6   K=SHA-256(len(X) || X || len(Y) || Y || len(Z) || Z || len(V) || V || len(w0) || w0)
7   where len(str) denotes the length of a string in bytes, represented as an 8-byte little-endian number.
8   OKM : [0:128] = CK, [128:128] = SK (all 16-byte)
9 end
```

3 18.4.5 Device-side computation of Shared Secret

4 *Listing 18-5: Device-side Computation of Shared Secret*

```
1 input: Y, w0, w1, x
2 output: CK, SK
3 begin
4   Z=xx(Y-w0xN)
5   V=w1x(Y-w0xN)
6   K=SHA-256(len(X) || X || len(Y) || Y || len(Z) || Z || len(V) || V || len(w0) || w0)
7   where len(str) denotes the length of a string in bytes, represented as an 8-byte little-endian number.
8   OKM : [0:128] = CK, [128:128] = SK (all 16-byte)
9 end
```

5 18.4.6 Derivation of Evidence Keys

6 *Listing 18-6: Derivation of Evidence Keys*

```
1 input: CK
2 output: K1, K2
3 begin
4   Compute the steps indicated by IETF RFC5869 \[13\] with the following mapping:
5   IKM : CK
6   L : 32
7   Salt : NULL
8   Info : "ConfirmationKeys" || TLV 5Bh || TLV 5Ch, see Table 5-4
9   Both TLVs include tag, length and data field.
10  Hash : SHA-256
11  OKM : [0:128] = K1, [128:128] = K2 (all 16-byte)
12  Notation is [start_index: number_of_bits]
13 end
```

7 18.4.7 Vehicle-side computation of Evidence

8 *Listing 18-7: Vehicle-side Computation of Evidence*

```
1 input: K1, X
2 output: M[1]
3 begin
4   compute M[1] = CMAC(K1, X) using K1 as secret key, using CMAC-AES-128 as defined in [RFC4493]
5   return M[1]
6 end
```

18.4.8 Device-side computation of Evidence

2 Listing 18-8: Device-side Computation of Evidence

```
1 input: K2, Y
2 output: M[2]
3 begin
4   compute M[2] = CMAC(K2, Y) using K2 as secret key, using CMAC-AES-128 as defined in [RFC4493]
5   return M[2]
6 end
```

18.4.9 Derivation of System Keys [WCC1/WCC2/WCC3]

4 Derivation of system keys Kenc, Kmac, Krmac for secure channel and of a long-term symmetric
5 secret for further key derivations. If the vehicle is WCC2 or WCC3, and if the device is either
6 WCC2 or WCC3 capable or device is capable of deriving Kble_intro and Kble_oob_master, then
7 additionally Kble_intro and Kble_oob_master shall be derived. This algorithm is not part of
8 SPAKE2+ but of the applicative part of the system.

9 Listing 18-9: Derivation of System Keys

```
1 input: SK, Flag_Kble_intro_Kble_oob_master_support
2 output: Kenc, Kmac, Krmac, LONG_TERM_SHARED_SECRET
3   If (Flag_Kble_intro_Kble_oob_master_support == true): Additional output: Kble_intro, Kble_oob_master
4   begin
5     Compute the steps indicated by IETF RFC5869 [13] with the following mapping:
6     IKM : SK
7     L : if (Flag_Kble_intro_Kble_oob_master_support == true) 96 else 64
8     Salt : NULL
9     Info : "SystemKeys"
10    Hash : SHA-256
11    OKM : [0:128] = Kenc, [128:128] = Kmac, [256:128] = Krmac, [384:128] = LONG_TERM_SHARED_SECRET
12      (all 16-byte)
13        if (Flag_Kble_intro_Kble_oob_master_support == true): [512, 128] = Kble_intro; [640, 128] = Kble_oob_master
14          (all 16 bytes)
15 Notation is [start_index: number_of_bits]
16 end
```

18.4.10 Generate Attestation Signature

11 The signature is calculated with ECDSA according to [8] using NIST P-256 curve and SHA-256.
12 See Listing 15-42.

18.4.11 Validate Attestation or Certificate

14 See Listing 15-43.

18.4.12 Secure Channel Command Encryption and Authentication

16 Listing 18-10: Secure Channel Command Encryption and Authentication

```
1 input: payload, Kmac, Kenc
2 output: encrypted_payload, mac
3 begin
4   Compute the steps as indicated in Section 6.2.6 of GPC_SPE_014 [7] with the following mapping:
5     Command Data Field - Plain Text : payload
6     Padded Counter Block : 00000000000000000000000000000000h || 1-byte counter (01h for first command, up to max
7     FFh)
```

```
7   S-ENC : Kenc
8   Ciphered Command Data Field : encrypted_payload
9   MACChainingValue : 00000000000000000000000000000000h on first command or
10    MAC Chaining value of previous command
11   S-MAC : Kmac
12   C-MAC : mac (8 bytes)
13   end
```

1 18.4.13 Secure Channel Response Encryption and Authentication

2 Listing 18-11: Secure Channel Response Encryption and Authentication

```
1 input: payload, Krmac, Kenc
2 output: encrypted_payload, mac
3 begin
4   Compute the steps indicated in Section 6.2.7 of GPC_SPE_014 [7] with the following mapping:
5   Response Data Field - Plain Text : payload
6   Padded Counter Block: 80000000000000000000000000000000h || 
7   1-byte counter_value used in command
8   S-ENC : Kenc
9   Ciphered Response Data Field : encrypted_payload
10  Response Data Field : encrypted_payload
11  MAC Chaining Value : 16-byte MAC Chaining Value from command
12  S-RMAC : Krmac
13  R-MAC : mac (8 bytes)
14  end
```

3 18.5 Error Handling [WCC1/WCC2/WCC3]

4 Vehicle and device should limit the time between starting of the pairing mode and the start of the
5 contactless transaction. After expiration of this time, the vehicle and device need to be brought
6 back to pairing mode by the user.

7 The overall time for the pairing transaction should be limited on vehicle and device side. When
8 the vehicle and device are brought into pairing mode and the pairing transaction starts (i.e.,
9 successful SELECT command/response exchange), the transaction time should be limited. Then,
10 either the pairing was successful, or the pairing process will be aborted by the vehicle and device
11 and needs to be restarted by the user. A new pairing password shall be entered again on the
12 device side.

13 The time to present the device again after a RF reset should be limited. The vehicle should abort
14 the pairing process if a device is not presented within this time limit after the restart of the reader
15 field.

16 18.6 Secure Element (SE)

17 **Secure Element (SE):** As defined within CCC, is embedded technology that provides a tamper-
18 resistant secure implementation aimed at providing isolation, integrity, confidentiality, secure
19 provisioning, secure storage, and secure processing for the Digital Key applet including its high-
20 value keying material while being isolated from the rich OS and application processor. An
21 implementation shall provide hardware and software protection against physical and logical
22 attacks and unsafe use during the full lifecycle of the secure implementation, the Digital Key
23 applet, and their assets.

1 **Tamper-resistant secure hardware:** Hardware designed to isolate and protect embedded
2 software and data by implementing appropriate security measures. The hardware and embedded
3 software must be resistant against high attack potential and must meet the requirements as
4 defined by the Digital Key certification which include resistance to physical tampering scenarios.
5 The acceptable scenarios are described in the latest Security IC Platform Protection Profile or in
6 other alternatives identified by the Digital Key certification.
7 The transport channel between a SE and the NFC component shall protect confidentiality and
8 integrity. The SE shall provide a secure binding between these components that shall be resistant
9 to manipulation after production. This is referred to as a hardware root of trust. It shall be
10 prevented that any code running on the application processor has access to or inject data on this
11 channel. The Secure Element shall have the ability to distinguish communication between wired
12 interface and contactless interface.

19 BLUETOOTH LOW ENERGY (LE) INTERFACE [WCC2/WCC3]

3 The Secure Ranging Service provides the Bluetooth LE framework necessary to discover,
4 manage, and control UWB-based ranging between a device and a vehicle. Bluetooth LE, Secure
5 Element, and UWB are core to the Digital Key solution. Bluetooth LE offers secure data
6 exchange between device and vehicle which then enables Secure Element to offer mutual
7 authentication and data sharing over a secure channel with the Vehicle. The Bluetooth LE
8 channel is also required to establish and or manage the Secure Ranging service.

19.1 Bluetooth LE Functional Requirements

10 The Bluetooth LE Controller and Host for both the device and vehicle shall be compliant to the
11 mandatory capabilities of the Bluetooth Core Specification 5.0 or later [30] and shall support the
12 following additional capabilities:

- 13 • LE L2CAP Connection Oriented Channel Support
- 14 • LE Privacy
- 15 • LE Secure Connections

16 The Bluetooth LE Controller and Host for both the device and vehicle may optionally support
17 the following additional capabilities:

- 18 • Long Range (LELR)
- 19 • LE Advertising Extensions (mandatory, if LE Long Range is supported)

20 Vehicles and devices shall use “LE security mode 1/Level 4” or/and “Secure Connections Only
21 mode,” see Vol 3, Part C, Section 10 of [30]. During a connection, the device and vehicle may
22 use the Feature Exchange Procedure to signal the LELR support based on the “LE Coded PHY”
23 bit defined in [30].

19.1.1 Vehicle

25 The Vehicle shall support the GAP Peripheral role.

26 The vehicle shall support privacy in the advertising state as defined in Volume 6, Part B, Section
27 6 of [30]. The vehicle shall use resolvable private addresses for advertising events as specified in
28 Volume 6, Part B, Section 6.2 of [30].

29 If the vehicle supports multiple Bluetooth LE controllers, all controllers shall use the same
30 identity address and resolution key. Each controller may have a different random resolvable
31 address at any time.

32 The vehicle shall support GATT in the server role.

19.1.2 Device

34 The device shall support the GAP Central role.

35 The device shall support resolvable private addresses used in advertisements by the vehicle.

- 1 The device shall initiate a connection to the vehicle when it receives the advertising packet from
- 2 a paired vehicle. A connection interval of 30ms is recommended when connecting to a vehicle.
- 3 The device shall support GATT in the client role.

4 **19.2 Bluetooth LE Procedures**

5 **19.2.1 Owner Pairing Connection Establishment**

6 This is the flow for owner pairing connection establishment using Bluetooth out-of-band (OOB)
7 pairing. The Bluetooth LE Setup is divided into the following two subsections:

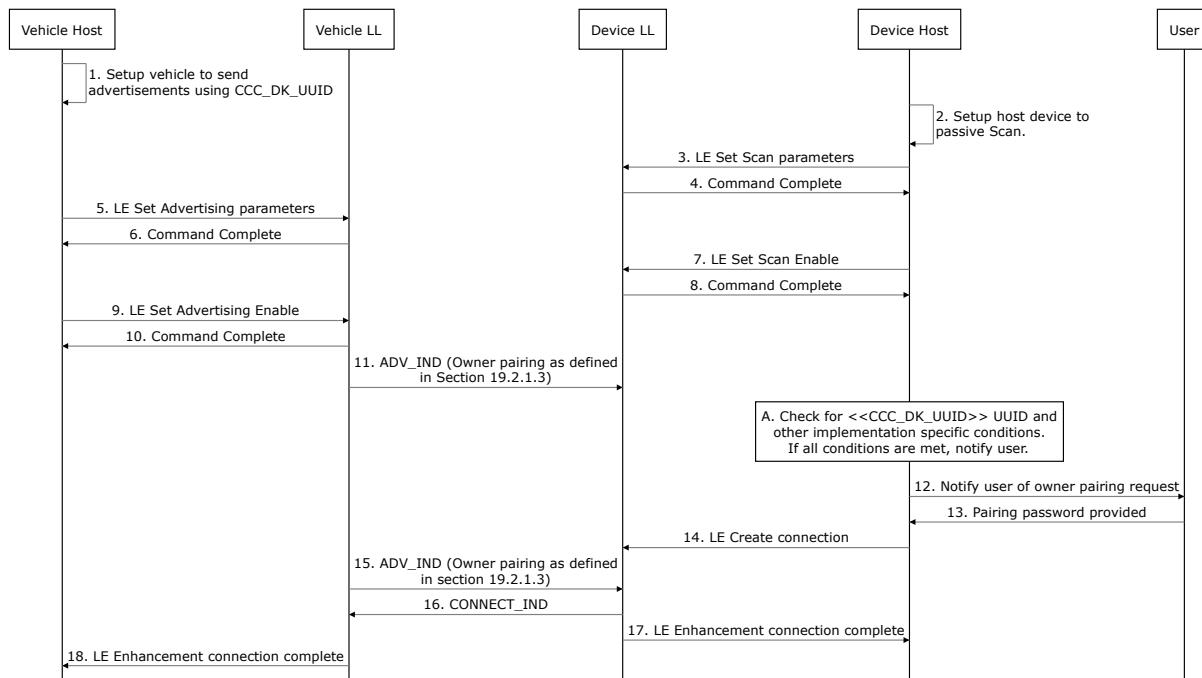
- 8 • Bluetooth LE Link Layer Connection Establishment & Bluetooth LE Owner Pairing
9 GATT Flow
- 10 • Bluetooth LE Pairing & Encryption Setup

11 *19.2.1.1 A vehicle that does not support secure ranging shall not allow owner pairing over the
12 Bluetooth LE interface and shall not broadcast the Owner Pairing Advertisement.
13 Owner pairing shall start over NFC as described in Section 6 and followed by
14 Bluetooth LE Activation flow as described in Section 19.5.10. Bluetooth LE Link Layer
15 Connection Establishment & Bluetooth LE Owner Pairing GATT Flow*

16 In [Figure 19-1](#), the vehicle begins by sending ADV_IND with CCC_DK_UUID as the
17 advertising payload (steps 11 and 15). The Vehicle LL shall be in the advertising state with its
18 filtering policy set to accept all connection requests. The device begins passive scanning. The
19 Device LL shall be in the scanning state with its filter policy set to accept all advertisements.
20 Once the Device LL receives an advertisement, it forwards that to the device host. The Device
21 Host shall check if the CCC_DK_UUID is contained in the advertisement payload (box A). If the
22 CCC_DK_UUID is contained in the advertising payload, the user is notified. If the user accepts
23 the owner pairing requests, the user provides the pairing password. The Device LL shall enter
24 initiating state with the filter policy set to the advertiser's address after step 14. Upon receiving
25 the next same advertisement from the Vehicle LL, the Device LL shall send a connection request
26 (CONNECT_IND as in step 16).

1

Figure 19-1: Bluetooth LE Link Layer Connection Establishment.

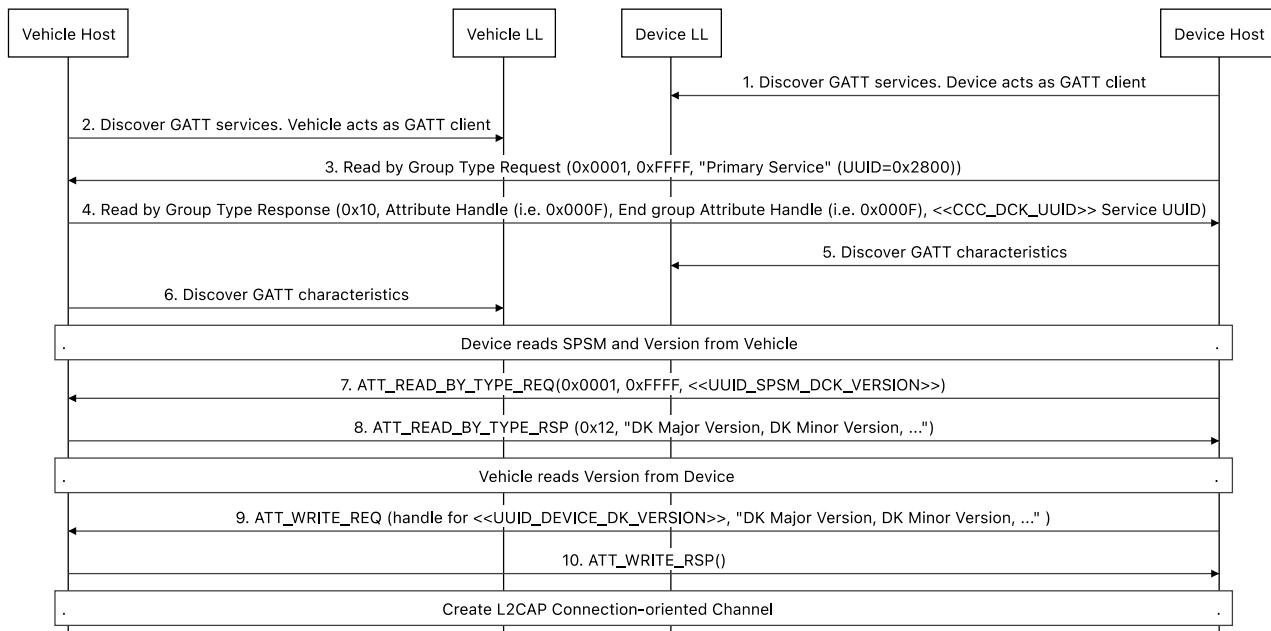


2

3 In [Figure 19-2](#), the Device Host (acting as the GATT client) initiates service discovery with the
4 Vehicle Host (acting as the GATT server) to get the DK Service and DK Service UUID_SPSM
5 DK_VERSION characteristic in order to establish the L2CAP connection for the DK Service.

6

Figure 19-2: L2CAP Connection-Oriented Channel.



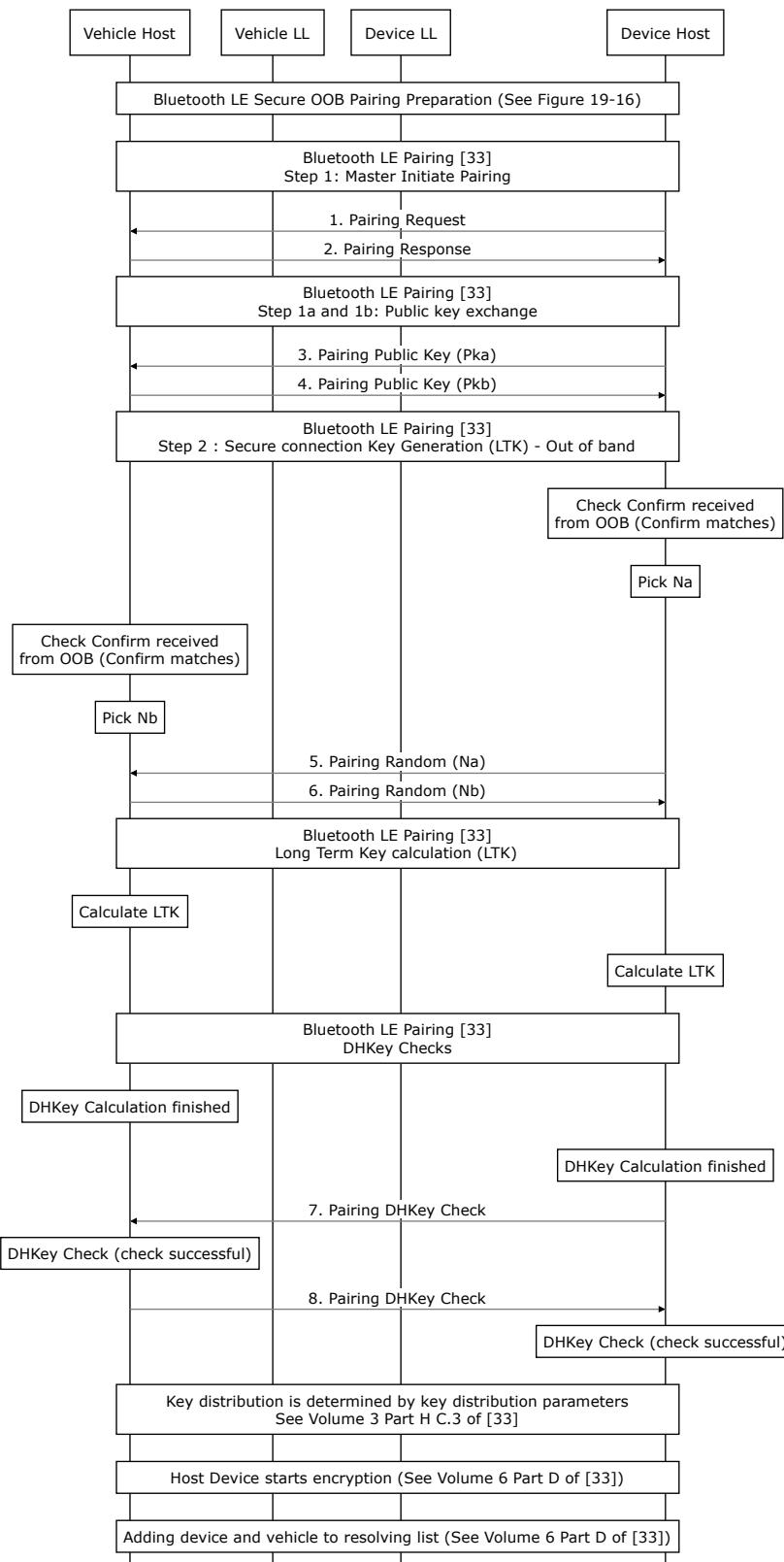
7

8 [Figure 19-2](#) is an example in which the device reads the SPSM value from the vehicle's GATT
9 Server. The device may read the SPSM value in other ways, as defined in the Volume 3 Part G
10 of [\[30\]](#).

- 1 For more details on Box B and the associated message flow, please refer to Volume 3 Part A
- 2 [\[30\]](#).

1 19.2.1.2 Bluetooth LE Pairing and Encryption Setup

2 Figure 19-3: Bluetooth LE Pairing and Encryption Setup.



3

In [Figure 19-3](#), the setup begins with the Bluetooth LE Secure OOB Pairing Preparation procedure as defined in Section [19.5.1](#) and shown in Figure 19-17.

During this procedure the device has generated its public key (PKa) and secret key (SKa) and has received the vehicle's Bluetooth address (BTAddrB), the vehicle's commitment value (Cb), and random number (rb) over a secured channel. The vehicle has generated its public key (PKb) and secret key (Skb) and received the device's Bluetooth address (BTAddrA) and the device's commitment value (Ca) and random number (ra) over a secured channel.

The procedure continues with the following:

- **Bluetooth LE Pairing: Master Initiated Pairing** (Pairing Phase 1, Step 1) as defined in [\[30\]](#). The Bluetooth LE Pairing Request PDU is sent from the device to the vehicle. The Bluetooth LE Pairing Response PDU is sent from the vehicle to the device.
- **Bluetooth LE Pairing: Public Key Exchange** (Pairing Phase 1, step 1a and 1b) Vol 2, Part H, Section 7.1 in [\[30\]](#). A Bluetooth LE Pairing Public Key PDU is sent from the device to the vehicle and a Bluetooth LE Pairing Public Key PDU is sent from the vehicle to the device. Both the device and vehicle begin their DHKey generation.
- **Bluetooth LE Pairing: Secure Connection Key Generation (LTK) – OOB** (Pairing Phase 2, Authentication Stage 1, Step 2) in [\[30\]](#). Both the device and vehicle verify if the confirm value received as part of the OOB Pairing Preparation procedure matches. Both the device and vehicle generate a random nonce (Na and Nb). The device sends its nonce (Na) to the vehicle using the Bluetooth LE Pairing Random PDU and the vehicle sends its nonce (Nb) to the device using the Bluetooth LE Pairing Random PDU.
- **Bluetooth LE Pairing: LTK calculation** (Pairing Phase 2, Authentication Stage 2) as described in Vol 3, Part H, Section 2.3.5.6.5 in [\[30\]](#). Once the DHKey generation is completed on the device and vehicle, the device and vehicle calculate their LTK.
- **Bluetooth LE Pairing: DHKey Checks.** The device sends the check value (Ea) using the Pairing DHKey Check PDU to the vehicle. The vehicle sends the check value (Eb) using the Pairing DHKey Check PDU to the device. Both the device and vehicle verify the values.
- **Bluetooth LE Pairing: Key Distribution** (Pairing Phase 3) as described in Vol 3, Part H, Section 2.4.3.2 in [\[30\]](#).
- **Device and vehicle enabling encryption** as described in Vol 3, Part H, Section 2.4.4.2 in [\[30\]](#). The device and vehicle add each other to their private address resolving list as described in Vol 6, Part B, Section 4.7 in [\[30\]](#).

The messages in the box “Host Device starts encryption” and “Adding device and vehicle to resolving list” are out of scope of this specification. These are shown for illustration purposes only. Please refer to Vol 6 Part D of [\[30\]](#).

19.2.1.3 Owner Pairing Advertising

For owner pairing, only the Legacy LE 1M PHY shall be used. The Advertisement (ADV_IND) for owner pairing shall follow the Section 2.3.1.1 of Volume 6 Part B of [\[30\]](#).

Event Type: Connectable and scannable undirected

The ADV_IND consists of Advertising Address and Advertising Data as shown in [Table 19-1](#) and [Table 19-2](#), respectively:

1 *Table 19-1: AdvA field of ADV_IND.*

Field	Length (bytes)
Advertising Address (AdvA)	6

2 *Table 19-2: AdvData field of ADV_IND.*

Field	Length (bytes)	Description	Value (hex)
Length	1	Length of AD Type and AD Data	0x03
AD Type	1	16-bit service UUID	0x03
AD Data	2	<<CCC_DK_UUID>>	0xFFFF
Length	1	Length of AD Type and AD Data	0x14
AD Type	1	Service data -128bit UUID	0x21
AD Data	16	CCCSERVICEIntent UUID	0x5810bbc0-b499-11e9-a2a3-2a2ae2dbcce4
AD Data	1	IntentConfiguration	0x1 (Default)
AD Data	2	Vehicle Brand Identifier	AS defined in Table 2-1 of [35]

3 The usage of CCC_DK_UUID: The Assigned Value is provided by Bluetooth SIG, Inc. and may
4 only be used by its members in compliance with all terms and conditions of use issued by
5 Bluetooth SIG, Inc. For more information visit
6 <https://www.bluetooth.com/specifications/assigned-numbers>.

7 The IntentConfiguration is used to allow separate user flows on the device, depending on
8 whether the user has started owner pairing in the vehicle or not. The byte is defined as described
9 in Table 19-3:

10 *Table 19-3: Definition of IntentConfiguration byte.*

Field	Description	Bit index
Intent	0: Intent from vehicle - user has actively started owner pairing in vehicle 1: No intent from vehicle	0
Reserved for future use	0: Set to 0 and shall be ignored by receiver	1:7

11 The vehicle brand identifier is specified in Table 2-1 in [35].

12 19.2.1.4 Pairing Request Definition:

13 The following fields are mandatory for the Pairing Request (see Volume 3, Part H, Section 3.5.1
14 of [30]):

15 *Table 19-4: Mandatory fields in Pairing Request.*

Field	Value (hex)	Definition
IO Capability	0x00 – 0xFF	See Vol 3, Part H, Table 3.4 in Section 3.5 of [30]
OOB data flag	1	OOB Authentication data from remote device present

Field	Value (hex)	Definition
AuthReq	1	Bonding

- 1 19.2.1.5 *Pairing response definition:*
 2 The following fields are mandatory for the Pairing Response (see Volume 3, Part H, Section
 3 3.5.2 of [30]):

4 *Table 19-5: Mandatory fields in Pairing Response.*

Field	Value (hex)	Definition
IO Capability	0x00 – 0xFF	See Vol 3, Part H, Table 3.4 in Section 3.5 of [30]
OOB data flag	0x1	OOB Authentication data from remote device present
AuthReq	0x1	Bonding
Maximum Encryption Key Size	0x10	
Initiator Key Distribution	0xF0	
Responder Key Distribution	0xF0	

- 5 19.2.1.6 *DK Service:*
 6 The vehicle shall support the DK Service. This shall be a primary service and there shall only be
 7 one instance of this service.

8 *Table 19-6: DK Service UUID.*

Field	Value
DK Service UUID	<<CCC_DK_UUID>>

- 9 19.2.1.7 *Vehicle PSM Characteristic:*
 10 This characteristic shall return the Simplified Protocol/Service Multiplexer (SPSM) used for the
 L2CAP channel from the vehicle.
 12 The device shall read the SPSM from the vehicle upon connection. The vehicle manufacturer
 shall pick an SPSM. For LE Credit-based Connections, a dynamically allocated SPSM value
 (i.e., the non-SIG assigned SPSM value in 0x0080-0x00FF) is used.
 15 During passive entry and owner pairing, the device shall upon Bluetooth LE connection:
 16 • Read the characteristics by UUID (with the UUID_SPSM or UUID_SPSM_DK_VERSION).
 The SPSM value shall use big-endian byte order.
 18 • Write the characteristics by UUID (for UUID_DEVICE_DK_VERSION)

19 *Table 19-7: SPSM Characteristic declaration.*

Attribute type	Attribute value		Attribute permission
0x2803 – UUID for «Characteristic»	Characteristic Properties = 0x02	0xMMMM = Handle of Characteristic Value D3B5A130-9E23-4B3A-8BE4- 6B1EE5F980A3 – UUID_SPSM	Read only, No Authentication, No Authentication

1

Table 19-8: SPSM Characteristic value declaration.

Attribute type	Attribute value	Attribute permission
D3B5A130-9E23-4B3A-8BE4-6B1EE5F980A3 – UUID_SPSM	uint16 – SPSM value	Read only, No Authentication, No Authentication

2 19.2.1.8 DK Version Characteristic:

3 *Table 19-9: Vehicle SPSM and DK version Characteristic Declaration*

Attribute type	Attribute value	Attribute permission
0x2803 – UUID for «Characteristic» 0x02	Characteristic Properties = 0x02 of Characteristic Value	Handle AE285B91-6D23-23F1-CA12-6B1EE5B780A3 – UUID_SPSM_DK_VERSION

4 *Table 19-10: Vehicle SPSM and DK Version Characteristic Value Declaration*

Attribute type	Attribute value	Attribute permission
D3B5A130-9E23-4B3A-8BE4-6B1EE5B780A3 – UUID_SPSM_DK_VERSION	Refer to Table 19-13 (Attribute value definition for Vehicle_SPSM_And_DK_Version_Characteristic)	Read only, Authenticated encryption required

5 *Table 19-11: Device Selected DK Version Characteristic Declaration*

Attribute type	Attribute value	Attribute permission
0x2803 – UUID for «Characteristic» 0x02	Characteristic Properties = 0x02 Handle of Characteristic Value	BD4B9502-3F54-11EC-B919-0242AC120005 – UUID_DEVICE_DK_VERSION

6 *Table 19-12: Device Selected DK Version Characteristic Value Declaration*

Attribute type	Attribute value	Attribute permission
BD4B9502-3F54-11EC-B919-0242AC120005 – UUID_DEVICE_DK_VERSION	Refer to Table 19-14 (Attribute value definition for Device_Selected_DK_Version_Characteristic)	Write only, Authenticated encryption required

7

8 *Table 19-13: Attribute Value Definition for Vehicle SPSM and DK Version Characteristic*

Attribute value	Length (Bytes)	Description
SPSM value	2	SPSM Value.
Supported_DK_Protocol_Version_Len	1	Length byte for supported DK Protocol versions.
Supported_DK_Protocol_Version	2 × n	DK_Protocol_Version is 2 Byte field (Major:Minor); minor version byte changes

		<p>whenever there is a change to DK protocol flows or message definitions. Major version byte changes when backward compatibility is broken.</p> <p>The vehicle shall return all supported DK Protocol Version using two bytes per version in big endian coding.</p> <p>n is the number of protocol versions supported by the vehicle</p> <p>The DK Protocol Version shall be ordered from the highest to the lowest version.</p> <p>In this specification the vehicle shall support the DK Protocol Version 1.0 (coded 0100_h). The device shall select the highest DK Protocol Version commonly supported by device and vehicle as described in Selected_DK_Protocol_Version attribute (See Table 19-14)</p> <p>The first supported DK Protocol Version is 0100_h.</p>
Features Supported length	1	1 byte is the length in bytes for the Features Supported Field per DK protocol version
Features Supported	k	<p>k is the number of bytes needed to convey the bitfield for the list of features supported.</p> <p>Refer to Table 19-15.</p>

1

2

Table 19-14: Device Selected DK Version Characteristic

Attribute value	Length (Bytes)	Description
Selected_DK_Protocol_Version	2	Device-selected highest DK Ranging Protocol Version commonly supported by device and vehicle. In this specification, the device shall support the DK Ranging Protocol Version 1.0 (coded 0100 _h). Attribute may not be included if Supported_DK_Ranging_Protocol_Version_Len = 0x00. Refer to Table 19-13.
Features Supported length	1	Length in bytes for the Features Supported Field
Features Supported	k	<p>k is the number of bytes needed to convey the bitfield for the list of features supported in common by vehicle and device.</p> <p>Refer to Table 19-15.</p>

- 1 The device shall respond with the latest version supported. DK Protocol Major Version and DK
2 Protocol Minor Version attribute values shall be encoded as big endian.
- 3 DK Protocol Major Version may not be backward compatible with previous DK Protocol Major
4 Versions.
- 5 DK Protocol Minor Version shall be compatible with previous DK Protocol Minor Version for
6 the same DK Protocol Major Version.
- 7 If UUID_SPSM_DK_VERSION is not available on vehicle, it shall be interpreted that the
8 vehicle supports DK Protocol Version 1.0 (i.e., DK Protocol Major Version is 1 and DK Protocol
9 Minor Version is 0).
- 10 If the device supports DK Protocol version 1.0, it shall request the UUID_SPSM from the
11 vehicle.
- 12 If the device supports DK Protocol Version higher than 1.0:
- 13 • Device shall request the UUID_SPSM_DK_VERSION characteristic from the vehicle. If
14 the vehicle doesn't support "UUID_SPSM_DK_VERSION", the device shall ask for
15 "UUID_SPSM" since the vehicle only supports DK Protocol version 1.0.
 - 16 • Device shall use the highest version commonly supported by both vehicle and device.
 - 17 • If vehicle supports V-D-BT version higher than 1.0, device shall write the
18 Selected_DK_Protocol_Version and Features_Supported to the Vehicle UUID_DEVICE_
19 DK_VERSION characteristic.
- 20 If the vehicle supports DK Protocol Version 1.0, it shall implement UUID_SPSM. If the vehicle
21 that does not support UUID_SPSM_DK_VERSION and the UUID_DEVICE_DK_VERSION
22 characteristic receives a request for the UUID_SPSM_DK_VERSION or
23 UUID_DEVICE_DK_VERSION from a device, it shall respond with the error code "Attribute
24 not found" as indicated by Bluetooth specification (refer to Bluetooth core specification v5.0 Vol
25 3, Part F 3.4.4.1).
- 26 If the vehicle supports DK Protocol Version higher than 1.0:
- 27 • Vehicle shall implement the UUID_SPSM and the UUID_SPSM_DK_VERSION
28 characteristic
- 29 The total size of the attribute value for UUID_SPSM and UUID_SPSM_DK_Version shall be
30 less than 512 bytes. If this value is exceeded, it shall keep the most recent anchor versions
31 followed by additional supported versions to make it fit within 512 Bytes.
- 32 **Features Supported definition:**
- 33 Features Supported shall be defined as follow:

34 *Table 19-15: Supported Feature Definition Bitmap*

Attribute type/ Feature Supported	Attribute value	Description
Bit[0]	0 or 1	1 _b : TimeSync Procedure 0 Supported Optional for vehicle, See 19.4.4.1
Bit[1]	0 or 1	1 _b : TimeSync Procedure 1 Supported

		Optional for vehicle, See 19.4.4.2
Bit[2]	0 or 1	1b: URSK derivation via Standard Transaction for other purposes Optional for device and vehicle, See 19.5.6.1
Bit[3]	0 or 1	1b: Head Unit Pairing Optional for device and vehicle, See 19.5.2
Bit[4]	0 or 1	1b: LE Coded PHY Passive Entry Advertising Optional for device and vehicle, See 19.2.3.1

1

2 *19.2.1.9 Vehicle Bluetooth Tx Power Level Characteristic:*

3 The Vehicle Bluetooth Tx Power Level characteristic is an optional attribute provided by the
4 vehicle. The Vehicle Bluetooth Tx Power Level characteristic shall return the current radiated
5 transmit power of a Bluetooth LE module in dBm [see GATT specification, supplement version
6 4].

7 *Table 19-16 Vehicle Bluetooth Tx Power Level Characteristics*

Attribute type	Data Type	Size (in octets)
0x2A07 – UUID for «Tx Power Level»	sint8	1

8

9 The device should read the Vehicle Bluetooth Tx Power level from the vehicle upon connection.
10 During passive-entry, the device should read the characteristics by UUID (with the UUID for
11 “Tx power level”) at reconnection. The Bluetooth LE module on the vehicle should keep the
12 same radiated transmit power during the entire time of the Bluetooth LE connection.

13 *19.2.1.10 Vehicle Antenna Identifier Characteristic:*

14 The Vehicle antenna identifier represents a unique identifier for every physical Bluetooth LE
15 antenna located in the vehicle that establishes a connection to a device. The antenna identifier is
16 defined as a unique value that correspond to a unique physical Bluetooth antenna in the vehicle.
17 This shall remain a unique antenna identifier even if a different device connects to that antenna.
18 The device should read the “Vehicle Antenna Identifier” from the vehicle upon connection.

19 In the case a Bluetooth module has multiple antennas, it can be considered as single unique
20 single antenna. Also, if multiple Bluetooth modules share multiple antennas, then it can also be
21 considered as a single antenna.

22 The GATT characteristic instance should be unique per client connecting to the vehicle if the
23 vehicle supports multiple concurrent Bluetooth LE connections.

24 During passive-entry, the device should read the characteristics by UUID (with the UUID_
25 AntennaIdentifier) at reconnection.

26 *Table 19-17: Vehicle Antenna Characteristic declaration*

Attribute type	Attribute value	Attribute permission
----------------	-----------------	----------------------

0x2803 – UUID for «Characteristic»	Characteristic Properties = 0x02	0xMMMM = Handle of Characteristic Value	c6d7d4a1-e2b0-4e95- b576-df983d1a5d9f – UUID_AntennaIdentifier	Read only, No Authentication, No Authorization
--	--	---	--	--

1

2

1

2 *Table 19-18: Vehicle Antenna Characteristic value declaration*

Attribute type	Attribute value	Attribute permission
c6d7d4a1-e2b0-4e95-b576-df983d1a5d9f – UUID_AntennaIdentifier	Uint16 – Vehicle Antenna Identifier	Read only, No Authentication, No Authorization

3 **19.2.2 Bluetooth Encryption**

4 The following requirements apply to Bluetooth encryption:

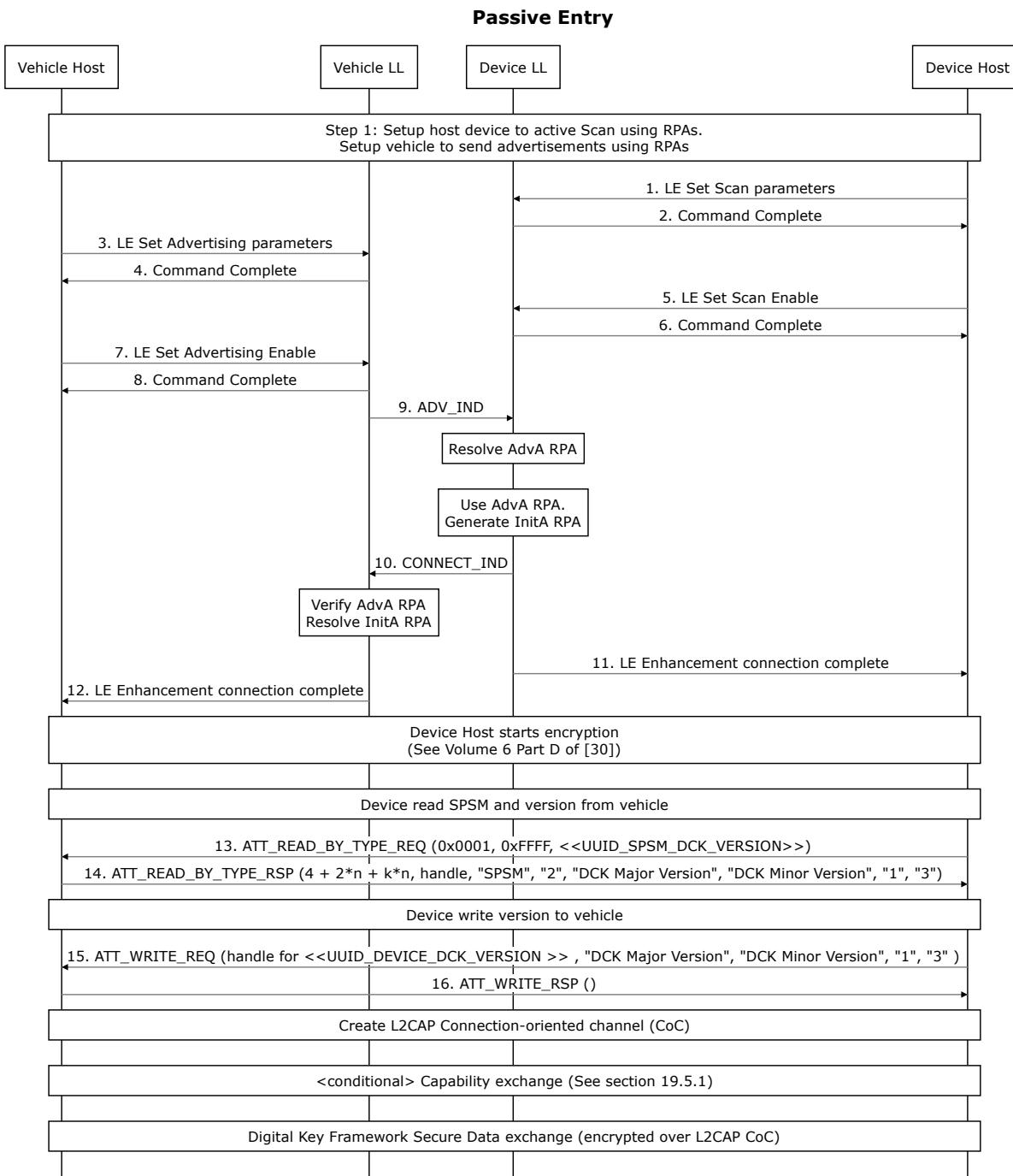
- 5 • The device (central) shall request encryption and encryption shall successfully complete
6 before L2CAP connection establishment is initiated, except for owner paring and first friend
7 approach.
- 8 • The vehicle shall trigger a disconnect 5 seconds after an unencrypted L2CAP connection
9 establishment if no First_Approach_RQ (see Section [19.3.4.1](#)) or Request_owner_pairing
10 Command Complete SubEvent notification (see Table 19-72) has been received during the
11 first approach and owner pairing, respectively.
- 12 • The device and vehicle should not terminate the L2CAP connection used for DK Service
13 while the BLE connection is alive. In case the L2CAP connection used for DK Service is
14 terminated while the Bluetooth LE connection is alive, the device shall attempt to re-establish
15 the DK Service L2CAP connection once conditions permit.

16 **19.2.3 Passive Entry: Bluetooth LE Setup:**

17 Once the Bluetooth LE Pairing and Encryption of the Bluetooth LE setup is completed, the
18 subsequent connection to the vehicle shall use the passive entry flow.

1

Figure 19-4: Passive Entry Flow Diagram.



2

- 3 For passive entry, Capability Exchange is conditional upon whether either the vehicle or device has an updated DK Protocol Version, UWB Configuration Identifier, PulseShape Combinations, or a combination of these parameters since the last Capability Exchange, as described in Section 4 5 6 [19.5.1](#).

1 *19.2.3.1 Passive Entry Advertising:*

2 The vehicle shall advertise on the LE 1M PHY.

3 When advertising on the LE 1M PHY, the connectable and scannable undirected event shall be
4 used, see Volume 6, Part B, Section 4.4.2.7 in [30]. The advertising interval shall be 42.5ms.

5 The vehicle may advertise on both the LE Coded PHY with S=2 coding and on the LE 1M PHY.
6 If both PHYs are used, the advertising for each shall occur consecutively.

7 When advertising on the LE Coded PHY, the connectable undirected event shall be used, see
8 Volume 6, Part B, Section 4.4.2.3 in [30]. The advertising interval shall be 84ms.

9 For both advertisements, the ADV_IND shall contain AdvA but no advertising data (AdvData).

10 A paired device shall be able to connect to both the owner pairing advertising and the passive
11 entry advertising to perform the passive entry flows. For known devices connecting to passive
12 entry as well as owner pairing advertisement, the vehicle shall be able to perform the Passive
13 Entry Flow.

14 *19.2.3.2 Passive Entry L2CAP Connection*

15 Upon establishment of the LE link layer connection, the device host shall establish an L2CAP
16 connection using the protocol specified in Section [19.2.1.1](#).

17 *19.2.3.3 Connection Performance Recommendations*

18 In this section, the connection performance parameters and their values are recommended. The
19 exact values are refined and finalized in the certification/test plan specification, which form the
20 certification requirement. All TBD values in this section will be finalized/measured during
21 interoperability event.

22 • **Device requirement:**

23 During the approach of the user toward the vehicle, the last Bluetooth message before the
24 UWB pre-poll should be sent by 3 (TBD_1) meters for 95 (TBD_2) percentiles. This shall be
25 based on the following assumptions:

- 26 ○ The maximum approach walk speed is 2.1m/s and the test start at 6 (TBD_3) meters.
- 27 ○ The conditions defined in certification/test plan specification.
- 28 ○ Device starts Bluetooth LE scanning at the start of test.

29 • **Transmitter requirements:**

- 30 ○ Bluetooth LE Advertisement packet (from vehicle), measured with reference
31 measurement setup, at 10 meters (TBD_5) should be equal or greater than -80 dBm
32 (TBD_4) with a free-space LOS path loss and common test case condition (see note
33 below).
- 34 ○ The radiated power output power level (including antenna gain) of a CONNECT_IND
35 packet from mobile device should be equal or greater than -80 dBm (TBD_6) at 10
36 meters (TBD_5) with a free-space LOS path loss and common test case condition (see
37 note below).

38 *Note:* Reference measurement setup and common test case condition are defined in
39 certification/test plan specification.

40

1 • **Receiver requirements:**

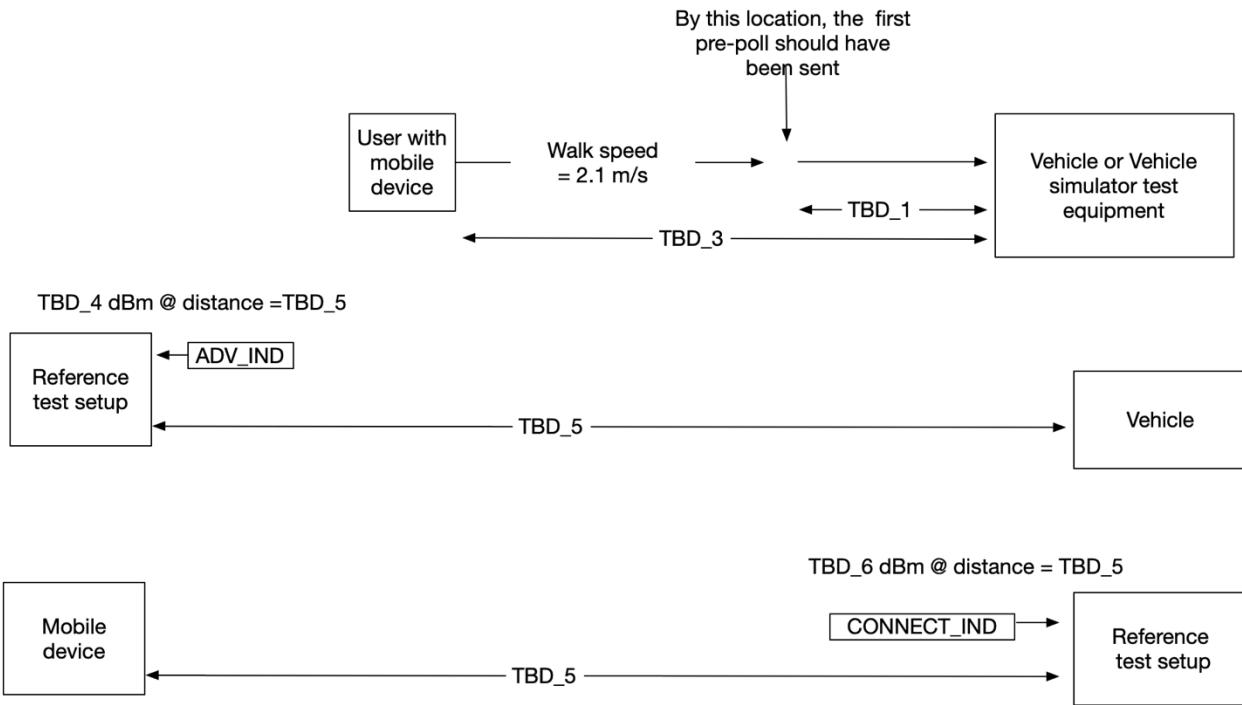
- 2 The actual sensitivity level, as defined in Version 5.2 Vol 6 Part A Section 4 of [30]:
3
 - o Should be equal or better than -85dBm for un-coded PHYs and -88dBm for LE Coded
4 PHY with S=2 coding for the device (see note below).
 - o Should be equal or better than -85dBm for un-coded PHYs and -88dBm for LE Coded
6 PHY with S=2 coding for the vehicle (see note below).

7 *Note:* Reference measurement setup and common test case condition are defined in
8 certification/test plan specification.

9 • **General requirements:**

10 Test should be done using the optimal ranging setup flow with a previously derived URSK
11 Any other losses in the Bluetooth link such as vehicle body or interior should be kept at a
12 minimum or at best avoided.

13 *Figure 19-5: Connection performance in relationship with device, transmitter, and receiver requirements.*



15 19.2.3.4 *Vehicle Requirements*

16 **Maximum advertising intervals:**

17 Maximum number of advertising events per time:

18 Locked vehicles shall transmit within 504ms (6 periods of 84ms) up to a maximum of

- 19
 - (a) 24 LE 1M + 12 LELR or
 - (b) 36 LE 1M + 6 LELR or
 - (c) 48 LE 1M (zero LELR)

22 advertising events (See Section 19.2.3.1).

- 1 Unlocked vehicles shall transmit within 504ms (6 periods of 84ms) up to a maximum of
2 • (a) 48 LE 1M + 24 LELR or
3 • (b) 72 LE 1M + 12 LELR or
4 • (c) 96 LE 1M (zero LELR)
5 advertising events (See Section 19.2.3.1).

6 **19.3 DK Message Format**

7 The DK messages shall be exchanged over L2CAP using the DK Service SPSM.
8 The device shall retrieve the DK Service SPSM from the vehicle's GATT server and establish an
9 LE L2CAP credit-based connection to the vehicle's SPSM for use by the DK Protocol (See
10 [Figure 19-2](#)).

11 The DK message format is shown in [Table 19-19](#).

12 *Table 19-19: DK Message Format.*

Message Field	Location
Message Header	Byte 0
Payload Header	Byte 1
Length	Byte [3:2]
Data	N

13 The Message Header field is one byte long and indicates the Message Type of the message being
14 sent. For example, if the message being sent is a Select APDU wrapped within DK_APDU_RQ
15 (see Section [19.3.2.1](#) for more details), the Message Header field indicates whether this message
16 is an SE message for standard transaction or Framework message for owner pairing, for which
17 the associated message processing differs by the type of message. On the other hand, the Payload
18 Header field is 1 byte long and communicates the message such as DK_APDU_RQ.

19 The Message Header is defined in [Table 19-20](#) and the Message Type is defined in [Table 19-21](#).
20 Bit [7:6] of the Message Header are reserved for future use and shall be set to 0.

21 *Table 19-20: Message Header definition.*

Message Header	Bit field
Message Type	Bit [5:0]
RFU	Bit [7:6]

22 *Table 19-21: Message Type definition.*

Message Type definition Bit [5:0]	Value (decimal)
Framework message	0
SE message	1
UWB Ranging Service message	2

Message Type definition Bit [5:0]	Value (decimal)
DK Event Notification	3
Vehicle OEM App message	4
Supplementary Service message	5
Head Unit Pairing message	6
Reserved	7–63

1 The Payload Header field is defined in Table 19-22:

2 *Table 19-22: Payload Header definition.*

Payload Header	Bit field
Message ID	Bit [7:0]

3 The Length field is two bytes long and indicates the size of the Data field in the message coded in Big Endian format. All the DK message fields shall be encoded in Big Endian format, except for data type fields already defined by Bluetooth Specification or otherwise explicitly defined in this specification. Those data types ordering defined by Bluetooth Specification shall remain unchanged.

4 The Data field is variable in length and its data structure definition is defined in Sections [19.3.1](#) to [19.3.8](#) for each message ID.

5 Table 19-23 shows the categorization of DK message under different Message Type.

6 *Table 19-23: Message Type and its associated messages.*

Message Type	Message	APDUs
Framework	DK_APDU_RQ DK_APDU_RS (As defined in Section 19.3.2)	SELECT, SPAKE2+ REQUEST, SPAKE2+ VERIFIER, WRITE DATA, GET DATA, GET RESPONSE, OP CONTROL FLOW, See Table 5-1
SE	DK_APDU_RQ DK_APDU_RS (As defined in Section 19.3.2)	SELECT, AUTH0, AUTH1, EXCHANGE CONTROL FLOW CREATE RANGING KEY See Table 15-1
UWB Ranging Service	Ranging_Capability_RQ Ranging_Capability_RS Ranging_Session_RQ	N/A

Message Type	Message	APDUs
	Ranging_Session_RS Ranging_Session_Setup_RQ Ranging_Session_Setup_RS Ranging_Suspend_RQ Ranging_Suspend_RS Ranging_Recovery_RQ Ranging_Recovery_RS Configurable_Ranging_Recovery_RQ Configurable_Ranging_Recovery_RS (As defined in Section 19.3.1)	
DK Event Notification	Ranging_Event (As defined in Section 19.3.8)	N/A
Vehicle OEM App	Pass_Through (As defined in Section 19.3.6)	N/A
Supplementary Service	Time_Sync (As defined in Section 19.3.3) First_Approach_RQ First_Approach_RS (As defined in Section 19.3.4) RKE_Auth_RQ RKE_Auth_RS (As defined in Section 19.3.5)	N/A
Head Unit Pairing	Head_Unit_Pairing_Preparation Head_Unit_Pairing_RQ Head_Unit_Pairing_RS (As defined in Section 19.3.7)	N/A

1 Below is an example on how to encode an APDU message before transporting it over Bluetooth
2 LE by using DK Message. The same can also be used to decode the incoming message.

```

3 Message: Select APDU message (See Section 15.3.2.1) over Bluetooth LE with
4 AID as 0xA000000809434343444B417631.
5 Message Type in Message Header: 0x01 (SE)
6 Message ID in Payload Header: 0x0B (DK_APDU_RQ, 11)
7 Data: 0x00A404000DA000000809434343444B41763100
8 (SELECT APDU command)
9 DK Bluetooth LE Payload: 0x010B001300A404000DA000000809434343444B41763100

```

10 *19.3.1 UWB Ranging Service Message*

11 This section details the Ranging Service messages to negotiate, initiate and complete UWB
12 ranging between the device and the vehicle.

1 **19.3.1.1 Ranging Capability Request (RC-RQ)**

2 This message is sent by the vehicle to the device during ranging capability exchange. The
3 Ranging Capability Request message and its parameters are defined in Table 19-24 and Table
4 19-25, respectively.

5 *Table 19-24: Ranging_Capability_RQ message and its parameters.*

Message	Message ID	Parameters
Ranging_Capability_RQ	0x01	Supported_DK_Protocol_Version_Len, Supported_DK_Protocol_Version, Supported_UWB_Config_Id_Len, Supported_UWB_Config_Id Supported_PulseShape_Combo_Len Supported_PulseShape_Combo

6 *Table 19-25: Definition of the parameters for Ranging_Capability_RQ.*

Parameters	Length (bytes)	Description
Supported_DK_Protocol_Version_Len	1	Length byte for supported DK protocol versions
Supported_DK_Protocol_Version	$2 \times n$	DK_Protocol_Version is 2 Byte field (Major:Minor); as defined in Table 19-13. If a DK_Protocol_Version was selected as part of the GATT version negotiation, this field shall contain only the selected DK_Protocol_Version. The first supported DK Protocol Version is 0100 _h
Supported_UWB_Config_Id_Len	1	Length byte for supported UWB Configuration Identifiers
Supported_UWB_Config_Id	$2 \times m$	UWB_Config_Id is a 2 Byte field which is an identifier for the supported UWB configuration. This configuration includes the supported PHY layer parameters. See Section 21.4 for the supported UWB PHY layer configurations. m is the number of UWB configs supported by vehicle
Supported_PulseShape_Combo_Len	1	Length byte for the supported PulseShape_Combos
Supported_PulseShape_Combo	$1 \times l$	PulseShape_Combo is a 1 Byte field which is an identifier for the supported initiator/response, transmit/receive pulse shape combinations. See Section 21.5 for the supported subset. l is the number of PulseShape_Combos supported by the vehicle.

7 **19.3.1.2 Ranging Capability Response (RC-RS)**

8 The message is sent by the device to the vehicle in response to a Ranging Capability Request
9 message (RC-RQ) during ranging capability exchange. The Ranging Capability Response
10 message and its parameters are defined in Table 19-26 and Table 19-27, respectively.

11 *Table 19-26: Ranging_Capability_RS message and its parameters.*

Message	Message ID	Parameters
Ranging_Capability_RS	0x02	Selected_DK_Protocol_Version, Selected_UWB_Config_Id Selected_PulseShape_Combo

1

Table 19-27: Definition of the parameters for Ranging_Capability_RS.

Parameters	Length (bytes)	Description
Selected_DK_Protocol_Version	2	Highest DK Protocol Version commonly supported by device and vehicle defined in Table 19-14. The first supported DK Protocol Version is 0100h. For DK Protocol Version higher than version 1.0, please refer to Section 19.2.1.7. If supported, the DK Protocol Version selected during GATT negotiation shall be used by device.
Selected_UWB_Config_Id	2	Device-selected UWB config id from the list of supported common UWB config ids between device and vehicle.
Selected_PulseShape_Combo	1	Device-selected single pulse shape combination from the list of common supported ones between device and vehicle.

2 **19.3.1.3 Ranging Session Request (RS-RQ)**

3 This message is sent by the vehicle to the device to start the ranging session setup handshake procedures as defined in Section [20.5.2](#). The Ranging_Session_RQ message and its parameters are defined in Table 19-28 and Table 19-29, respectively

4 The vehicle shall use the Selected_DK_Protocol_Version, Selected_UWB_Config_id and Selected_PulseShape_Combo as selected by the device during ranging capability exchange (see Figure 19-18).

5 If the vehicle has an updated DK Protocol Version or UWB Configuration Identifier or PulseShape Combinations or combination of these parameters, the vehicle shall perform Capability Exchange with the device again as described in Section [19.5.1](#).

12 *Table 19-28: Ranging_Session_RQ message and its parameters.*

Message	Message ID	Parameters
Ranging_Session_RQ	0x03	Selected_DK_Protocol_Version, Selected_UWB_Config_id, UWB_Session_Id, Selected_PulseShape_Combo, Channel_Bitmask

13 *Table 19-29: Definition of the parameters for Ranging_Session_RQ.*

Parameters	Length (bytes)	Description
Selected_DK_Protocol_Version	2	Selected protocol version from ranging capability exchange (RC-RQ, RC-RS)
Selected_UWB_Config_Id	2	Selected UWB config id from ranging capability exchange (RC-RQ, RC-RS)
UWB_Session_Id	4	The least significant 4 bytes of the transaction_identifier field. For more details on transaction_identifier, See Table 15-27. Vehicle selects the transaction_identifier based on which pre-derived URSK it chooses to use for establishing UWB ranging session. The least significant 4 bytes of that transaction_identifier is then used as UWB_Session_Id. Device uses UWB_Session_Id for URSK retrieval and session management.
Selected_PulseShape_Combo	1	Device-selected single pulse shape combination from the list of common supported ones between device and vehicle.

Parameters	Length (bytes)	Description
Channel_Bitmask	1	A bitmap equaling the bitwise “OR” of supported UWB channels Bit 0: UWB channel 5 supported <u>Bit 1: UWB channel 9 supported</u> <u>Bit 2: RFU (shall be set to 0)</u> <u>Bit 3: RFU (shall be set to 0)</u> <u>Bit 4: RFU (shall be set to 0)</u> <u>Bit 5: RFU (shall be set to 0)</u> <u>Bit 6: RFU (shall be set to 0)</u> <u>Bit 7: RFU (shall be set to 0)</u>

- 1 *19.3.1.4 Ranging Session Response (RS-RS)*
 2 This message is sent by the device according to Section [20.5.2](#). The Ranging Session Response
 3 message and its parameters are defined in Table 19-30 and Table 19-31, respectively. If the
 4 device has an updated DK Protocol Version or UWB_Config_Id or PulseShape Combinations or
 5 combination of these parameters which is different than the ones it received in Ranging Session
 6 Request, the device shall respond with DK Event Notification with Subevent_Category set to
 7 0x01_h and the corresponding command status code set to 0x02_h, to initiate Capability Exchange
 8 as described in Section [19.5.1](#).

9 *Table 19-30: Ranging_Session_RS message and its parameters.*

Message	Message ID	Parameters
Ranging_Session_RS	0x04	RAN_Multiplier, Slot_BitMask, SYNC_Code_Index_BitMask, Selected_UWB_Channel, Hopping_Config_Bitmask

10 *Table 19-31: Definition of the parameters for Ranging_Session_RS.*

Parameters	Length (bytes)	Value	Description
RAN_Multiplier	1	Range = 1 to 255 T_Block RAN = RAN_Multiplier × 96ms Time Range = 96ms to 24480 ms	RAN multiplier for setting the ranging block duration. See Section 20.5.2 for the definition ranging block.
Slot_BitMask	1	A bitmap equaling the bitwise “OR” combination of 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80.	Bitmap of supported values of Slot durations as a multiple of T _{Chap} , N _{Chap_per_Slot} as defined in Section 20.5.2 . Each “1” in this bit map corresponds to a specific value of N _{Chap_per_Slot} where: 0x01 = “3” 0x02 = “4” 0x04 = “6” 0x08 = “8” 0x10 = “9” 0x20 = “12” 0x40 = “24” and 0x80 is reserved. Note that 0x40 is used for testing only

Parameters	Length (bytes)	Value	Description
SYNC_Code_Index_BitMask	4	A bitmap equaling the bitwise “OR” combination of 0x00000001, 0x00000002, 0x00000004, ... 0x40000000, 0x80000000	Bitmap of SYNC code indices that can be used. The position of each “1” in this bit pattern corresponds to the index of a SYNC code that can be used, where: 0x00000001 = “1” 0x00000002 = “2” 0x00000004 = “3” 0x00000008 = “4” ... 0x40000000 = “31” 0x80000000 = “32” Refer to [31] and Section 21.4.1 for SYNC code index definition.
Selected_UWB_Channel	1	“0x01 for channel 5” and “0x02 for channel 9”	Selected UWB channel for requested session. See Section 21.7.1 .
Hopping_Config_Bitmask	1	0-255	[b7 b6 b5] bitmask of hopping modes, the device offers to use in this ranging session 100 No Hopping 010 Continuous Hopping 001 Adaptive Hopping The mandatory mode is 010 corresponding to Continuous Hopping mode. The support of 100 or 001 is optional. [b4 b3 b2 b1 b0] bit mask of hopping sequences the device offers to use in this ranging session. b4=1 is always set since the default hopping sequence and support thereof is mandatory and defined in Appendix A. b3=1 is set when the optional AES-based hopping sequence, which is defined in Appendix G, is supported. All remaining bits reserved for future hopping sequences.

- 1 *19.3.1.5 Ranging Session Setup Request (RSS-RQ)*
 2 This message is sent by the vehicle as part of the ranging session setup handshake according to
 3 Section [20.5.2](#). The Ranging Session Setup Request message and its parameters are defined in
 4 [Table 19-32](#) and [Table 19-33](#), respectively.

5 *Table 19-32: Ranging_Session_Setup_RQ message and its parameters.*

Message	Message ID	Parameters
Ranging_Session_Setup_RQ	0x05	Session_RAN_Multiplier, Number_Chaps_per_Slot, Number_Responders_Nodes, Number_Slots_per_Round, SYNC_Code_Index, Selected_Hopping_Config_Bitmask See Section 20 for the definition of these parameters.

1

Table 19-33: Definition of the parameters for Ranging_Session_Setup_RQ.

Parameters	Length (bytes)	Value	Description
Session_RAN_Multiplier	1	Range = 1 to 255 T_Block_S = Session_RAN_Multiplier × 96 ms Time Range = 96ms to 24480 ms	Session specific RAN multiplier. Value selected by vehicle must be greater than or equal to RAN_Multiplier sent by mobile device.
Number_Chaps_per_Slot	1	3, 4, 6, 8, 9, 12, or 24 Note that the last value is used only for testing	Selected Slot duration as a multiple of Tchap, NChap_per_Slot as defined in Section 20 .
Number_Responders_Nodes	1	1-255	Number of logical responder nodes participating in this ranging session as selected by the vehicle. See [31] for the definition of a logical node.
Number_Slots_per_Round	1	6,8,9,12,16,18,24,32,36,48,72, or 96	Selected number of slots per round
SYNC_Code_Index	4	A bitmap equaling the bitwise “OR” combination of 0x00000001, 0x00000002, 0x00000004, ... 0x40000000, 0x80000000	Bitmap of SYNC code indices that the vehicle can use (this may be a smaller set of the SYNC codes supported by the vehicle). The position of each “1” in this bit pattern corresponds to the index of a SYNC code that can be used, where: 0x00000001 = “1” 0x00000002 = “2” 0x00000004 = “3” 0x00000008 = “4” ... 0x40000000 = “31” 0x80000000 = “32” See [31] and Section 21 for SYNC code index definition.
Selected_Hopping_Config_-Bitmask	1	0-255	Vehicle shall send a bitmask indicating the selected hopping mode and the selected hopping sequence (in case of continuous or adaptive hopping). This bitmask shall meet the following rules: - For [b7 b6 b5], only one of those bits shall be set to 1 - For [b5 b4 b3 b2 b1 b0], at most one of the bits is set to 1 to signal the selected hopping sequence and the rest of the bits are set to 0. Examples of a valid Selected_Hopping_Config_BitMask [10000000] no hopping mode [01010000] continuous hopping with default hopping sequence [00110000] adaptive hopping with default hopping sequence [01001000] continuous hopping with AES-based optional hopping sequence [00101000] adaptive hopping with AES-based optional hopping sequence

1 **19.3.1.6 Ranging Session Setup Response (RSS-RS)**

2 This message is sent by the device to the vehicle to complete the ranging session setup
3 handshake according to Section [20.5.2](#). The Ranging Session Setup Response message and its
4 parameters are defined in Table 19-34 and Table 19-35, respectively.

5 *Table 19-34: Ranging_Session_Setup_RS message and its parameters.*

Message	Message ID	Parameters
Ranging_Session_Setup_RS	0x06	STS_Index0, UWB_Time0, HOP_Mode_Key, SYNC_Code_Index See Section 20 for the definition of these parameters

6 *Table 19-35: Definition of the parameters for Ranging_Session_Setup_RS.*

Parameters	Length (bytes)	Value	Description
STS_Index0	4	0 - 0x3FFFFFFF	Starting STS index
UWB_Time0	8	0 - 0xFFFFFFFFFFFFFF	Starting time reference (resolution 1 microsecond) on UWB Device Clock of ranging session (see Section 20.3).
HOP_Mode_Key	4	0 - 0xFFFFFFFF	Key to generate default or AES based hopping sequence. In case of no hopping mode, this parameter shall be set to 0.
SYNC_Code_Index	1	1-32	Selected SYNC code index

7 **19.3.1.7 Ranging Suspend Request Message (RSD-RQ)**

8 This message is used to suspend an active ranging session for a given UWB_Session_Id. The
9 Ranging Suspend Request message and its parameters are defined in Table 19-36 and Table
10 19-37, respectively.

11 *Table 19-36: Ranging_Suspend_RQ message and its parameter.*

Message	Message ID	Parameters
Ranging_Suspend_RQ	0x07	UWB_Session_Id

12 *Table 19-37: Definition of the parameter for Ranging_Suspend_RQ.*

Parameters	Length (bytes)	Description
UWB_Session_Id	4	The UWB Session ID of the currently active UWB ranging session.

13 **19.3.1.8 Ranging Suspend Response Message (RSD-RS)**

14 This message is used as a successful response to Ranging Suspend Request (RS-RQ). The
15 Ranging Suspend Response message and its parameters are defined in Table 19-38 and Table
16 19-39, respectively.

17 *Table 19-38: Ranging_Suspend_RS message and its parameter.*

Message	Message ID	Parameters
Ranging_Suspend_RS	0x08	Suspend_Response

1 Table 19-39: Definition of the parameter for Ranging_Suspend_RS.

Parameters	Length (bytes)	Description
Suspend_Response	1	0x00 - For accepted suspend request 0x01 - To delay the suspend and keep ranging active

2 19.3.1.9 Ranging Recovery Request Message (RR-RQ)

3 This message is sent by vehicle to recover a suspended ranging session for a given
4 UWB_Session_Id. The Ranging Recovery Request message and its parameters are defined in
5 Table 19-40 and Table 19-41, respectively. The device and vehicle shall use the
6 Session_RAN_Multiplier negotiated prior to ranging suspension (see Section [19.3.1.5](#)).

7 Table 19-40: Ranging_Recovery_RQ message and its parameter.

Message	Message ID	Parameters
Ranging_Recovery_RQ	0x09	UWB_Session_Id

8 Table 19-41: Definition of the parameter for Ranging_Recovery_RQ.

Parameters	Length (bytes)	Description
UWB_Session_Id	4	The UWB Session ID of a previously suspended UWB ranging session.

9 19.3.1.10 Ranging Recovery Response Message (RR-RS)

10 This message is sent by device as a successful response to vehicle's Ranging Recovery Request
11 (RR-RQ). The Ranging Recovery Response message and its parameters are defined in Table
12 19-42 and Table 19-43, respectively.

13 Table 19-42: Ranging_Recovery_RS message and its parameter.

Message	Message ID	Parameters
Ranging_Recovery_RS	0x0A	STS_Index0, UWB_Time0

14 Table 19-43: Definition of the parameter for Ranging_Recovery_RS.

Parameters	Length (bytes)	Description
STS_Index0	4	Starting STS index
UWB_Time0	8	Starting time reference (resolution 1 microsecond) on UWB Device Clock of ranging session (see Section 20.3).

15 19.3.1.11 Configurable Ranging Recovery Request Message (CRR-RQ)

16 This message is sent by vehicle to recover a suspended ranging session for a given
17 UWB_Session_Id. The Configurable Ranging Recovery Request Message and its parameters are
18 defined in Table 19-44 and Table 19-45, respectively. This message enables the vehicle to
19 request a different RAN Multiplier value than the one selected prior to the suspension (i.e.
20 Session_RAN_Multiplier). For example, the vehicle may request an aggressive frequency to
21 support higher ranging rate for critical use cases such as door handle touch sensor.

1 *Table 19-44: Configurable_Ranging_Recovery_RQ message and its parameter.*

Message	Message ID	Parameters
Configurable_Ranging_Recovery_RQ	0x12	UWB_Session_Id Requested_RAN_Multiplier

2 *Table 19-45: Definition of the parameter for Configurable_Ranging_Recovery_RQ.*

Parameters	Length (bytes)	Description
UWB_Session_Id	4	The UWB Session ID of the currently active UWB ranging session.
Requested_RAN_Multiplier	1	Requested RAN Multiplier for the ranging session to be recovered. The Requested_RAN_Multiplier shall be different than the Session_RAN_Multiplier prior to ranging suspension (see Section 19.3.1.5).

3 **19.3.1.12 Configurable Ranging Recovery Response Message (CRR-RS)**

4 This message is sent by device as a response to CRR-RQ. The Configurable Ranging Recovery
5 Response message and its parameters are defined in Table 19-46 and Table 19-47, respectively.

6 *Table 19-46: Configurable_Ranging_Recovery_Response message and its parameter.*

Message	Message ID	Parameters
Configurable_Ranging_Recovery_RS	0x13	Selected_RAN_Multiplier STS_Index0, UWB_Time0

7 *Table 19-47: Definition of the parameter for Configurable_Ranging_Recovery_Response.*

Parameters	Length (bytes)	Description
Selected_RAN_Multiplier	1	Selected RAN Multiplier for the ranging session to be recovered. The Selected_RAN_Multiplier shall be equal or greater than the Requested_RAN_Multiplier
STS_Index0,	4	Starting STS index
UWB_Time0	8	Starting time reference (resolution 1 microsecond) on UWB Device Clock of ranging session (see Section 20.3).

8 The Selected_RAN_Multiplier shall be used during the recovered ranging session and shall
9 remain valid until the ranging session ends or the ranging session is suspended. When the
10 ranging session is recovered again, the RAN_Multiplier value for the given ranging session shall
11 revert to the default Session_RAN_Multiplier for that ranging session.

12 **19.3.2 SE Message**

13 **19.3.2.1 APDU Command Encapsulation**

14 The DK_APDU_RQ message is an encapsulation of the APDU command specified as available
15 on the wired interface in Table 15-1. The message and its parameters are defined in Table 19-48
16 and Table 19-49, respectively.

Table 19-48: *DK_APDU_RQ* message and its parameter.

Message	Message ID	Parameter
DK_APDU_RQ	0x0B	APDU command

Table 19-49: Definition of the parameter for DK_APDU_RQ.

Parameter	Length (bytes)	Description
APDU command	Variable	<p>List of APDU commands are described in Table 19-23.</p> <p>Allowed class byte values are limited to 00_h (for SELECT command), 80_h (for commands with non-secure messaging) or 84_h (for commands with secure messaging).</p>

3 If the Digital Key framework has selected the Digital Key applet over a supplementary logical
4 channel, it shall update the class byte of the APDU command accordingly before forwarding the
5 APDU command to the Digital Key applet.

6 19.3.2.2 APDU Response Encapsulation

The DK_APDU_RS message is an encapsulation of the APDU response. Each DK_APDU_RQ message requires a corresponding DK_APDU_RS message. The message and its parameter are defined in Table 19-50 and Table 19-51, respectively.

Table 19-50: *DK_APDU_RS* message and its parameter.

Message	Message ID	Parameter
DK_APDU_RS	0x0C	APDU response

Table 19-51: Definition of the parameter for DK_APDU_RS.

Parameter	Length (bytes)	Description
APDU response	Variable	The APDU response corresponding to the APDU command

19.3.3 Supplementary Service Message - Time Sync

13 This message is used by device to provide Bluetooth LE Timesync payload which includes, the
14 DeviceEventCount, the UWB Device Time timestamp and the UWB Device Time uncertainty.

15 The following two conditions shall trigger a timestamp (see Section 19.4):

- 16 • **Procedure 0:** After a CONNECT_IND, the device shall send a Time_Sync message
17 • **Procedure 1:** Once Bluetooth LE connection has been established, the vehicle may trigger a
18 new Bluetooth LE event-based timestamp by requesting “LE Set PHY”. Device shall use the
19 anchor point of the connection event used for “LL_PHY_UPDATE_IND” Link Layer
20 message to generate a new time synchronization timestamp and send it to vehicle using this
21 message.

Table 19-52 and Table 19-53 describe the parameters meaning based on which condition the timesync message was triggered.

Table 19-52: Time_Sync message and its parameters.

Message	Message ID	Parameters
Time_Sync	0x0D	DeviceEventCount,

Message	Message ID	Parameters
		UWB_Device_Time, UWB_Device_Time_Uncertainty, UWB_Clock_Skew_Measurement_available, Device_max_PPM, Success, RetryDelay

1

Table 19-53: Definition of the parameter for Time_Sync.

Parameters	Length (bytes)	Value	Description
DeviceEventCount	8	0 – 0xFFFFFFFFFFFFFFFE	Procedure 0: Event count is 0 or in case of retransmission, close to 0. Procedure 1: Event count value used for the LL_PHY_UPDATE_IND Link layer message. A value of 0xFFFFFFFFFFFFFF indicates DeviceEventCount is unavailable on the device.
UWB_Device_Time	8	0 – 0xFFFFFFFFFFFFFFFE	UWB Ranging clock value in microseconds. This is running clock and shall remain valid for the entirety of the session.
UWB_Device_Time_Uncertainty	1	0 – 0xFF	Time uncertainty between the anchor point of connection event and UWB Clock domain mapping. Format is log 2 accuracy of unsigned value in microseconds divided by 8: Value [usec] = $2^{(UWB_Device_Time_Uncertainty / 8)}$ The value range which can be represented is 1μsec ($=2^0$ μsec) to 1.094 hour ($=2^{255/8}$ μsec)
UWB_Clock_Skew_Measurement_available	1	0 – 0x1	This is used to inform the vehicle if the device has a clock system which allows the vehicle to estimate and correct for the UWB clock skew, based on DeviceEventCount and UWB_Device_Time information. In particular this requires that the device UWB clock is available at Procedure 0 and Procedure 1. Value = 1: UWB clock skew measurement is available Value = 0: UWB clock skew measurement is NOT available.
Device_max_PPM	2	0-0xFFFF	Worst case clock skew of device UWB clock (with respect to nominal time). Format: Unsigned value representing (single sided) clock skew. Shall be interpreted that clock skew could be either positive or negative. Units is PPM (parts per million).
Success	1	0-0x2	0: The “Bluetooth LE Timesync” procedure has failed and the timesync message parameters are invalid. The vehicle should ask again to perform the “Bluetooth LE Timesync” after RetryDelay 1: The “Bluetooth LE Timesync” procedure on device side is successfully able to map the UWB clock and all parameters in the timesync message are valid. 2: The “Bluetooth LE Timesync” procedure has failed and the Procedure 0 is not available. This only applies to Procedure 0.

Parameters	Length (bytes)	Value	Description
RetryDelay	2	0-0xFFFF	Unsigned 16bits. Value in milliseconds for minimum delay required by device until the vehicle should trigger a new “Bluetooth LE Timesync”.

1 19.3.4 *Supplementary Service Message - First Approach*
 2 First Approach messages enable Bluetooth LE OOB Secure LE pairing for both owner and friend
 3 devices. These messages are used to share OOB Bluetooth LE data via application level
 4 encrypted channel through Kble_intro and Kble_oob as described in Bluetooth LE Secure OOB
 5 Pairing Prep (see Figure 19-17).

6 19.3.4.1 *First Approach Request Message (FA-RQ)*
 7 This message is sent by the device to the vehicle to share device OOB data. The device OOB
 8 data comprises a combination of two encrypted payloads i.e. E1_Payload and E2_Payload. The
 9 E1_Payload and E2_Payload are encrypted using Kble_intro and Kble_oob, respectively. The
 10 message and its parameters are defined in Table 19-54 and Table 19-55, respectively.

11 *Table 19-54: First_Approach_RQ message and its parameters.*

Message	Message ID	Parameters
First_Approach_RQ	0x0E	E1_Payload IV1, Tag1, E2_Payload IV2, Tag2

12 *Table 19-55: Definition of the parameter for First_Approach_RQ.*

Parameters	Length (bytes)	Description
E1_Payload	8	E1_Payload is an encrypted (AES-128-CCM) DK_Identifier using Kble_intro key. If slot_identifiers are provided by the vehicle, the DK_Identifier is an 8-byte zero padded slot_identifier based on the slot_identifiers provided by the vehicle. If slot identifiers are provided online, the DK_Identifier is an 8-byte zero padded ot_identifier based on the random slot_identifier generated by the owner device during key_sharing.
IV1	8	IV used by device for encrypting E1_Payload using Kble_intro
Tag1	4	AES-CCM authentication tag for E1_Payload using Kble_intro
E2_Payload	38	E2_Payload is encrypted (AES-128-CCM) BTAddrA (6 Byte) Ca (16 Byte) ra (16 Byte) using Kble_oob as shared secret. The BTAddrA, Ca and ra are Bluetooth address of the device at connection establishment, Calculated Value on the device, and Random Value on the device, respectively (see Figure 19-17).
IV2	8	IV used by device for encrypting E2_Payload using Kble_oob
Tag2	4	AES-CCM authentication tag for E2_Payload using Kble_oob

1 19.3.4.2 First Approach Response Message (FA-RS)

2 This message is sent by the vehicle to the device to share vehicle OOB data which is encrypted
3 by Kble_oob key. The message and its parameters are defined in Table 19-56 and Table 19-57,
4 respectively.

5 *Table 19-56: First_Approach_RS message and its parameters.*

Message	Message ID	Parameters
First_Approach_RS	0x0F	E_Payload IV, Tag

6 *Table 19-57: Definition of the parameter for First_Approach_RS.*

Parameters	Length (bytes)	Description
E_Payload	38	E_Payload is encrypted (AES-128-CCM) BTAddrB (6 Byte) Cb (16 Byte) rb (16 Byte) using Kble_oob as shared secret. The BTAddrB, Cb and rb, are the Bluetooth address of the vehicle at connection establishment, Calculated Value on the vehicle, and Random Value on the vehicle, respectively (see Figure 19-17).
IV	8	IV used by vehicle for encrypting vehicle OOB data using Kble_oob.
Tag	4	AES-CCM authentication tag for E_Payload using Kble_oob

7 19.3.5 Supplementary Service Message - RKE

8 19.3.5.1 RKE_Auth_RQ

9 This message is sent by vehicle upon receiving RKE action SubEvent (See Section [19.5.9](#)).
10 The content of the RKE_Auth_RQ and its parameter are shown in Table 19-58 and Table 19-59,
11 respectively.

12 *Table 19-58: RKE_Auth_RQ message and its parameter.*

Message	Message ID	Parameter
RKE_Auth_RQ	0x14	RKE Challenge

13 *Table 19-59: Definition of the parameter for RKE_Auth_RQ.*

Parameter	Length (bytes)	Description
RKE Challenge	16	Vehicle unique challenge generated per RKE request to avoid replay attack.

14 19.3.5.2 RKE_Auth_RS

15 This message is sent by the device in response to the RKE_Auth_RQ.
16 The content of the RKE_Auth_RS is shown in Table 19-60.

17 *Table 19-60: RKE_Auth_RS message and its parameter.*

Message	Message ID	Parameters
RKE_Auth_RS	0x15	All fields contained Table 15-61 .

1 Note 1: The content of the arbitrary_data field (Tag 58_h) in [Table 15-56](#) is described in Section
2 [19.5.9](#).

3 Note 2: The content of the usage field (Tag 93_h) in [Table 15-56](#) is set to D074DA4F_h or
4 FC6F4C17_h.

5 19.3.6 Vehicle OEM App Message

6 This message is used by the vehicle OEM's app or the vehicle to exchange messages with each
7 other over the same L2CAP channel used in the Digital Key Service. Digital Key framework is
8 agnostic of this message and its data structure. It is simply acting as pass-through. The message
9 and its parameters are defined in Table 19-61 and [Table 19-62](#), respectively.

10 *Table 19-61: Pass_through message and its parameter.*

Message	Message ID	Parameter
Pass_through	0x10	Payload

11 *Table 19-62: Definition of the parameter for Pass_through.*

Parameter	Length (bytes)	Description
Payload	Variable	Proprietary payload known to vehicle OEM's app and vehicle. Length of the payload depends on the device/vehicle payload size over Bluetooth LE limitations.

12 19.3.7 Head Unit Pairing Message

13 Head Unit Pairing messages enable head unit pairing between vehicle and device as part of the
14 owner pairing as described in Section [19.5.1](#).

15 19.3.7.1 Head_Unit_Pairing_Preparation Message (HU-PP)

16 The message and its parameters are defined in Table 19-63 and Table 19-64, respectively.

17 *Table 19-63: HU_PP message and its parameters.*

Message	Message ID	Parameter
Head_Unit_Pairing_Preparation (HU-PP)	0x16	BT_Pairing_Configuration, BD_ADDR_Head_Unit, Vehicle capabilities

18 *Table 19-64: Definition of the parameter for HU_PP.*

Parameter	Length (bytes)	Description
BT_Pairing_Configuration	1	BitMask for supported capabilities: OOB_Configuration: Bit[0] : Value = 0. The vehicle head unit only supports OOB communication from vehicle head unit to device. Bit[0] : Value = 1. The vehicle head unit supports OOB communication for both directions. Bit[1-7]: Reserved
BD_ADDR_Head_Unit	6	Bluetooth Device Address of the head unit as defined in Volume 2 Part H of [30]
Vehicle capabilities	2	BitMask for supported capabilities on vehicle

Parameter	Length (bytes)	Description
		Bit [1]: Value = 0: Head unit Bluetooth Classic (A2DP and HFP) pairing not supported Bit [1]: Value = 1: Head unit Bluetooth Classic (A2DP and HFP) pairing supported Bit [2]: Value = 0: Apple Carplay not supported Bit [2]: Value = 1: Apple Carplay Supported Bit [3]: Value = 0: Android Auto Not Supported Bit [3]: Value = 1: Android Auto Supported Bit[4:15]: Reserved

1 **19.3.7.2 Head Unit Pairing Request Message (HUP-RQ)**

2 This message is sent by the device to the vehicle. The message and its parameters are defined in
3 Table 19-65 and Table 19-66, respectively.

4 *Table 19-65: HUP_RQ message and its parameters.*

Message	Message ID	Parameter
Head_Unit_Pairing_RQ (HUP-RQ)	0x17	User Intent, BD_ADDR_Device, Device Capabilities, PairingDataR192, PairingDataR256, PairingDataC192, PairingDataC256

5 *Table 19-66: Definition of the parameter for HUP_RQ.*

Parameter	Length (bytes)	Description
User Intent	2	BitMask for user intent coming from device. Bit [0]: Value =0: Intent from device Bit [0]: Value =1: No intent from device Bit [1]: Value = 0: Head unit Bluetooth Classic usage not intended from user. Bit [1]: Value = 1: Head unit Bluetooth Classic usage intended from user Bit [2]: Value = 0: Apple Carplay usage not intended from user Bit [2]: Value = 1: Apple Carplay usage intended from user Bit [3]: Value = 0: Android Auto usage not intended from user Bit [3]: Value = 1: Android Auto usage intended from user Bit [4-15]: Reserved
BD_ADDR_Device	6	Bluetooth Device Address of the device as defined in Volume 2 Part H of [30] .
Device Capabilities	2	BitMask for supported capabilities on device Bit [1]: Value = 0: Head unit Bluetooth Classic (A2DP and HFP) not supported Bit [1]: Value = 1: Head unit Bluetooth Classic (A2DP and HFP) supported Bit [2]: Value = 0: Apple Carplay not supported Bit [2]: Value = 1: Apple Carplay supported Bit [3]: Value = 0: Android Auto not supported Bit [3]: Value = 1: Android Auto supported Bit[4-15]: Reserved
PairingDataR192	16	Simple Pairing Randomizer R-192*
PairingDataR256	16	Simple Pairing Randomizer R-256*

Parameter	Length (bytes)	Description
PairingDataC192	16	Simple Pairing Hash C-192*
PairingDataC256	16	Simple Pairing Hash C-256*
* See volume 2 Part H Section 7.2.2 of [30]		

1 *19.3.7.3 Head Unit Pairing Response Message (HUP-RS)*

2 This message is sent by the vehicle to the device. The message and its parameters are defined in
3 Table 19-67 and Table 19-68, respectively.

4 *Table 19-67: HUP_RS message and its parameters.*

Message	Message ID	Parameter
Head_Unit_Pairing_RS (HUP-RS)	0x18	BD_ADDR_Head_Unit, PairingDataR192, PairingDataR256, PairingDataC192, PairingDataC256

5 *Table 19-68: Definition of the parameter for HUP_RS.*

Parameter	Length (bytes)	Description
BD_ADDR_Head_Unit	6	Bluetooth Device Address of the head unit as defined in Volume 2 Part H of [30] .
PairingDataR192	16	Simple Pairing Randomizer R-192 *
PairingDataR256	16	Simple Pairing Randomizer R-256 *
PairingDataC192	16	Simple Pairing Hash C-192 *
PairingDataC256	16	Simple Pairing Hash C-256 *

* See volume 2 Part H Section 7.2.2 of [\[30\]](#)

6 *19.3.8 DK Event Notification*

7 DK Event Notifications are used to encapsulate events that are generated by the device or vehicle
8 participating in the ranging exchange. The message and its parameters are defined in Table 19-69
9 and Table 19-70, respectively.

10 *Table 19-69: DK Event Notification message and its parameters.*

Message	Message ID	Parameter
DK Event Notification	0x11	Subevent_Category, Subevent_Code

11 *Table 19-70: Definition of the parameters for DK Event Notification.*

Parameter	Length (bytes)	Description
Subevent_Category	1	DK SubEvent category which are listed in the table below
Subevent_Code	Variable	SubEvent code as defined below per category.

1 *19.3.8.1 DK SubEvent Category*

2 Table 19-71 defines categories of DK SubEvents supported.

3 *Table 19-71: DK SubEvents Category and its parameters.*

DK SubEvent Category	SubEvent_Category	Parameter
Command Complete SubEvent	1	Command_Status
Ranging Session Status Changed SubEvent	2	Session_Status
Device Ranging Intent SubEvent	3	DR_Intent
Vehicle Status Change SubEvent	4	Vehicle_Status
RKE Request Subevent	5	Requested_Action
Head Unit Pairing SubEvent	6	Headunit_Pairing_Status
RFU	0x07 - 0xFF	None

4 *19.3.8.2 DK SubEvent Codes Per Category*

5 **Command Complete SubEvent**

6 The Command Complete SubEvent is used to send appropriate code upon processing of last message. Table 19-72 and Table 19-73 detail the Command_Status codes applicable to the ‘Command Complete SubEvent’ category. For detailed flow on Command_Status SubEvents, see Section [19.7.1](#).

10

11 *Table 19-72: Definition of Command_Status and its Command_Status codes.*

Parameter	Length (bytes)	Value	Description
Command_Status	1	0x00 - 0xFF	See Table 19-73 for all command status codes

12

Table 19-73: List of Command_Status for Command Complete SubEvent.

Command_Status	Code	Description
Deselect_SE	0x00	Vehicle shall send this SubEvent to deselect the Digital Key applet. Upon receiving this, the device shall deselect the Digital Key applet. For each secure element transaction over Bluetooth LE (see SE message in Section 19.3.2), it shall be terminated by this command regardless of the transaction is successful or fail.
BLE_pairing_ready	0x01	Vehicle shall send this SubEvent code when its ready for Bluetooth LE pairing during Owner Pairing
Require_capability_exchange	0x02	Device shall send this SubEvent to signal a mismatched ranging capability or an update to the device capability. Upon receiving this, the Vehicle shall trigger ranging capability exchange (see Section 19.7.1.1)
Request_standard_transaction	0x03	Device shall send this SubEvent to signal the vehicle in all cases during owner pairing and first friend approach where a standard transaction is expected. (e.g. if key_tracking_receipt is available in OP). Upon Receiving this, the vehicle should start a standard transaction. If a vehicle has not received the Request_standard_transaction SubEvent (e.g. due to transmission error) or if the vehicle is busy (e.g. vehicle is still

Command_Status	Code	Description
		busy processing the preceding action), then the vehicle should start the standard transaction after a timeout or when the processing is done, respectively. If the vehicle initiates a standard transaction but the device becomes not ready, the device may respond with a Device_busy SubEvent or may not respond during the command timeout. The vehicle shall retry the standard transaction until the device responds. A timeout shall be applied if the device does not respond.
Request_owner_pairing	0x04	Device shall send this SubEvent to request owner pairing
Reserved	0x05- 0x7F	Reserved for future use
General_error	0x80	Device or vehicle may send this SubEvent to signal an unknown error. Upon receiving this, the vehicle or device may retry or abort.
Device_SE_busy	0x81	Device may send this SubEvent to signal the SE is temporarily unavailable. The device may retry internally to get SE resources before sending this. Upon receiving this, the vehicle may retry the digital key flow.
Device_SE_transaction_state_lost	0x82	Device may send this SubEvent to signal standard transaction state was lost. Upon receiving this, the vehicle may retry the transaction
Device_busy	0x83	Device may send this SubEvent to signal device temporal not able to respond to the sent request. Upon receiving this, the vehicle may retry the action.
Command_temporarily_blocked	0x84	Device shall send this SubEvent when the device UA state and UA policy do not allow the requested command.
Unsupported_channel_bitmask	0x85	Device shall send this SubEvent if the channels provided in the channel bitmask are currently not supported by the device
OP_Device_not_inside_vehicle	0x86	Vehicle shall send this SubEvent when the device was not found inside the vehicle during owner pairing. This error shall cause the vehicle and device to abort owner pairing
Device_resource_available	0x87	The device sends this subevent to the vehicle that it recovered from a state where access to device internal resources was blocked. The vehicle may use this information to retry a flow that failed due to that previous device state.
Reserved	0x88- 0xFB	Reserved for future use
OOB_mismatch	0xFC	Device or vehicle may send this SubEvent to signal the incompatible OOB data exchanged during Bluetooth LE pairing. This is an error code for debugging purposes.
BLE_pairing_failed	0xFD	Device or vehicle may send this SubEvent to signal the failure in Bluetooth LE pairing. This is an error code for debugging purposes.
FA_crypto_operation_failed	0xFE	Device or vehicle may send this SubEvent to signal the failure in First approach due to cryptography. This is an error code for debugging purposes.
Wrong_parameters	0xFF	Device or vehicle may send this SubEvent to signal the use of unsupported message or format. This is an error code for debugging purposes.

1 Ranging Session Status Changed SubEvent

2 The Ranging Session Status Changed SubEvent is generated to indicate the start, progress and
3 completion of an UWB related action. These SubEvents can be generated by both the device and
4 the vehicle. The Session_Status codes are described in Table 19-74 and Table 19-75.

1 *Table 19-74: Definition of Session_Status and its Session_Status codes.*

Parameter	Length (bytes)	Value	Description
Session_Status	1	0x00 - 0xFF	See Table 19-75 for all session status codes.

2 *Table 19-75: List of Session_Status for Ranging Session Status Changed SubEvent.*

Session_Status	Code	Description
Ranging_session_URSK_refresh	0x00	Vehicle may send this SubEvent to request clean-up for pre-derived URSKs.
Ranging_session_URSK_not_found	0x01	Device shall send this SubEvent to signal, the device failed to find URSK for this UWB_Session_Id
Ranging_session_not_required	0x02	Device shall send this SubEvent when it detects the user does not intend to approach the vehicle e.g device is moving away from the vehicle. Upon receiving this subEvent, the vehicle may immediately suspend the ranging session.
Ranging_session_secure_ranging_failed	0x03	Vehicle shall send this SubEvent to signal the Vehicle failed to establish secure ranging.
Ranging_session_terminated	0x04	Vehicle or device may send this SubEvent if it has stopped the ranging session (e.g. due to URSK TTL expiration)
Ranging_session_recovery_failed	0x06	Device shall send this SubEvent to signal it failed to recover ranging for this UWB_Session_Id
Ranging_session_suspended	0x07	Vehicle or device shall send this SubEvent only if it is suspending the current ranging session without allowing the responder to delay the suspension (e.g. UWB has temporarily become unavailable). When vehicle or device receives this, it shall suspend its current ranging session as well without sending a Ranging_Suspend_RQ or a Ranging_session_suspended SubEvent.
Reserved	0x08-0xFF	Reserved for future

3 **Device Ranging Intent SubEvent Codes**

4 This set of SubEvents are generated by the device to notify the vehicle that the user may be
5 approaching the vehicle and ranging may be required. The DR_Intent codes are described in
6 [Table 19-76](#) and [Table 19-77](#).

7 *Table 19-76: Definition of DR_Intent and DR_Intent codes.*

Parameter	Length (bytes)	Value	Description
DR_Intent	1	0x00-0xFF	Low_approach_confidence, Medium_approach_confidence, High_approach_confidence, Reserved

8 *Table 19-77: List of DR_Intent for Device Ranging Intent SubEvent.*

DR_Intent	Code	Description
Low_approach_confidence	0x00	Device notifies vehicle of user approaching with low confidence
Medium_approach_confidence	0x01	Device notifies vehicle of user approaching with medium confidence
High_approach_confidence	0x02	Device notifies vehicle of user approaching with high confidence
Reserved	0x03- 0xFF	Reserved for future use

1 The device shall use the following definition of Confidence levels to select DR_Intent. The
2 setting of the DR_Intent is up to the device OEM implementation. When the vehicle is in low
3 power mode, the vehicle may decide to ignore ranging intent with low or medium approach
4 confidence. When the vehicle is not in low power mode, and if the vehicle is ready to perform
5 the secure ranging operation (e.g., no vehicle internal conflict), the vehicle shall initiate the
6 secure ranging regardless of the DR_Intent confidence level.

7 To enable secure ranging based on a Device Ranging Intent, the device shall send a Device
8 Ranging Intent SubEvent with at least low_approach_confidence between the point in time a
9 Bluetooth LE connection with the vehicle is established and the point in time the vehicle triggers
10 secure ranging (e.g. user touches the door handle).

11 The performance requirements from Section 19.2.3.3 shall apply when the vehicle starts the
12 Ranging Session Setup flow after receiving a Device Ranging Intent SubEvent

13 *Approach detection probability:*

- 14 • Describes the probability an intent was sent in the last 2 to 20 seconds before the
15 customer touched the door
- 16 • $P(\text{Approach detection}) = P(\text{in last [2s,20s]} \mid \text{an intent was sent} \mid \text{Customer touches door}$
17 handle)

18 *False approach detection probability:*

- 19 • Describes the probability that a ranging intent was sent, but the customer has not arrived
20 at the vehicle touching the door handle within the following 20s
- 21 • $P(\text{False approach detection}) = P(\text{Customer has not touched the door handle within the last}$
22 $20s \mid \text{ranging intent received 20s ago})$

23 *Definition of Confidence levels:*

- 24 • Low_approach_confidence:
 - 25 1. Approach detection probability: >90%, and
 - 26 2. False approach detection probability: as low as possible
- 27 • Medium_approach_confidence:
 - 28 3. Approach detection probability: as high as possible, and
 - 29 4. False approach detection probability: <75%
- 30 • High_approach_confidence:
 - 31 5. Approach detection probability: as high as possible, and
 - 32 6. False approach detection probability: <50%

33 **Vehicle Status Changed SubEvent**

34 The Vehicle Status Changed SubEvent is used to inform the device about state changes of
35 vehicle functions as shown in [Table 19-78](#). The Digital Key framework on the device shall store
36 the latest values of received function state changes as long as the BLE connection is established.

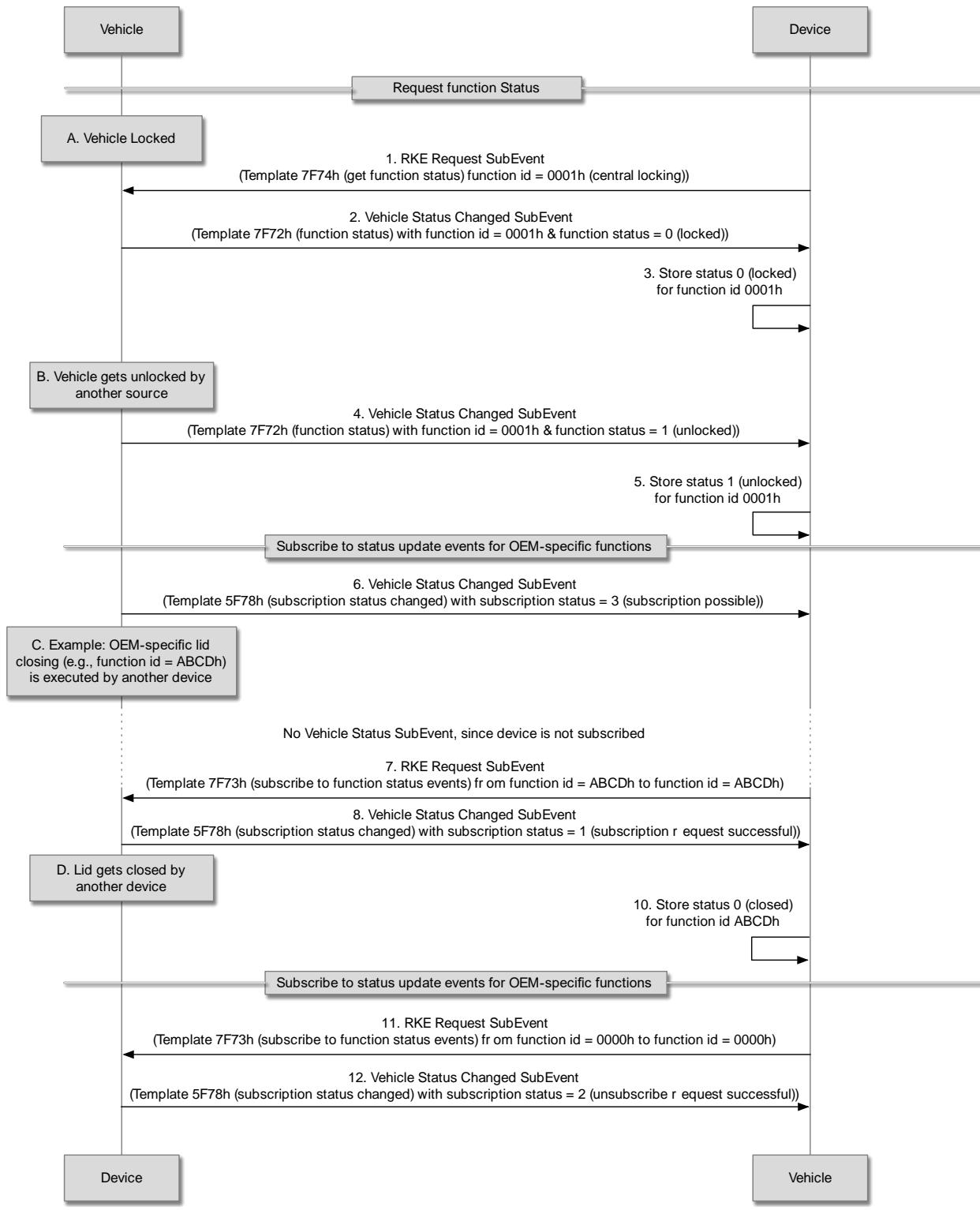
1

Table 19-78: List of Vehicle Status Changed SubEvent Tags.

The content of the Vehicle Status Changed SubEvent message is encoded as TLV format as defined in [1]				Length (bytes)	Description	Field is
7F71 _h				10	Last requested action execution status for function / action id. Tag is only present when an RKE function was triggered beforehand or an automatic action was triggered due to the location / movement of the device (e.g., walk away locking).	C
	80 _h			2	Function id	M
	81 _h			1	Action id	M
	82 _h			1	Execution Status 00 _h : Function successfully executed 01 _h : Execution started 02 _h : Execution stopped successfully (upon user stop request) 03 _h : Execution ended (e.g., limit reached) 04 _h -0F _h : Reserved 10 _h : Execution not possible or cancelled - unspecific reason 11 _h : Execution not possible due to vehicle state 12 _h : Authentication Error 13 _h : Device response timeout 14 _h : Device out of execution range 15 _h -4F _h : Reserved 50 _h -EF _h : Execution not possible or cancelled with OEM specific meaning. The meaning of these unsuccessful status codes is proprietary and may be used to send a more specific error reason to the vehicle OEM app. The definition is up to the vehicle OEM. F0 _h -FF _h : Standardized error codes defined in Table 19-80.	M
7F72 _h			variable		Vehicle function status summary. The tag is present, when the vehicle sends at least one status message.	C
	30 _h		variable		Sequence of vehicle function status	M
	A0 _h		variable		Vehicle function status	M
		80 _h	2		Function id	M
		83 _h	1		Function status	M
		C0 _h	variable		Vehicle OEM proprietary data (max. 64 bytes)	O
7F75 _h			variable		Request confirmation for enduring RKE action. Only present, when enduring RKE action is in progress.	C
	80 _h		2		Function id	M
	81 _h		1		Action id	M
	87 _h		variable		Arbitrary data with a maximum size of 64 bytes.	O
5F78 _h			1		Subscription status changed, only present, when the vehicle changes the subscription state 0: Unsubscribed all, no subscription possible 1: Subscription request successful 2: Unsubscribe request successful 3: Subscription possible/resubscription required	5F78 _h

1

Figure 19-6: RKE SubEvent Subscription Process



2
3
4

1 Tag 7F71_h shall always be sent together with 7F72_h in a single SubEvent message to the device
2 that triggered the action causing the vehicle state to change. An exception is when the vehicle
3 state takes more than 250ms to change after the action received. In this situation, 7F71_h and
4 7F72_h can be sent separately in two different subevent messages. When sent together, the
5 templates order shall be always 7F71_h followed by 7F72_h. In case of automatic actions triggered
6 due to the location of the device, the device that triggered a state change (i.e., ‘central locking’ or
7 ‘lock and secure’) will receive 7F71_h and 7F72_h as a single SubEvent message with the
8 appropriate action id 0x50(lock) and 0x51(unlock). Other BLE connected devices (not used in
9 decision making) will only receive the 7F72_h template in the SubEvent notification. The vehicle
10 can make an exception for devices in the vicinity as described below.

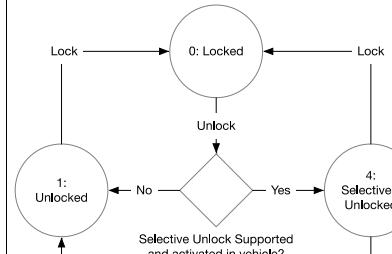
11 For central locking or lock and secure, the template 7F71_h can be included with 7F72_h to inform
12 devices that are in the vicinity of the vehicle but did not directly cause the event. This shall only
13 be done for devices located in zones relevant for passive entry (‘vicinity’).

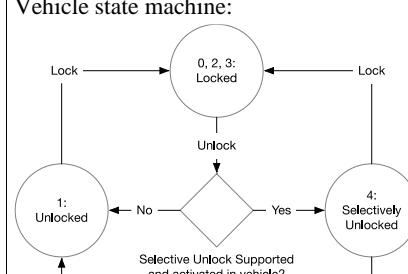
14 The 7F71_h template shall not be included for below conditions:

- 15 • Lock upon car in motion
- 16 • Car lock/unlock from inside by the user

17 Table 19-79 lists the functions and its corresponding action ids and/or status values. Depending
18 on the usage of the particular ID, it may not provide the corresponding action id or status value.
19 For certain actions that represent automatically executable functions, e.g., due to device position
20 or movement, the vehicle may send an execution status without an explicit RKE trigger. An
21 example of this is using it to signal when execution of an automatic action is unsuccessful (e.g.,
22 walk away lock with vehicle state not allowing to lock the vehicle).

23 *Table 19-79: Definition of Function ids and their corresponding Action ids.*

Function id	Function	Possible action ids	Possible function status values (Note 1)	Event-based or enduring action	Device (Note 2)	Vehicle (Note 2)
0000_h-000F_h Central locking category (vehicle may support up to one of the following options)						
0001 _h	Central locking	0: Lock 1: Unlock 50: Vehicle-triggered Lock (only for execution status) 51: Vehicle-triggered Unlock (only for execution status)	0: Locked 1: Unlocked 	Lock, Unlock: Event; Walk Away Lock, Approach Unlock: only execution status	M	C, only available if 0002 _h is not available
0002 _h	Lock and secure	0: Lock and secure 1: Unlock 2: Lock with partial arming of alarm system	0: Locked and secured 1: Unlocked 2: Locked with partially armed alarm system 3: Locked with partial arming available	Lock and secure, Unlock, Lock with partial arming of alarm system: Event; Walk Away	M	O

Function id	Function	Possible action ids	Possible function status values (Note 1)	Event-based or enduring action	Device (Note 2)	Vehicle (Note 2)
		50: Vehicle-triggered Lock & Secure (only for execution status) 51: Vehicle-triggered Unlock (only for execution status)	4: Selectively Unlocked, Only driver door unlocked (with global unlock available on unlock action) Vehicle state machine: 	Lock, Approach Unlock: only execution status		
0003 _h ... 000F _h	Reserved for standardization					
0010 _h	Driving Readiness		0: No Driving Readiness 1: Driving Readiness (e.g., engine "running" for ICE vehicles)	Event	-	M
0011 _h	Vehicle low power Mode		0: Normal power mode 1: Low power mode	Event	-	O
0012 _h	Low Fuel Status		0: Fuel not low 1: Fuel Low	-	-	O
0013 _h ... 001F _h	Reserved for standardization					
0100 _h	Remote Climatization	0: Stop Climatization 1: Start Climatization	0: Remote Climatization not active 1: Remote Climatization active	Event	O	O
0101 _h	Panic alarm	0: Mute panic alarm 1: Trigger panic alarm	0: Panic alarm not active 1: Panic alarm active	Event	M	O
0102 _h	Fuel Lid	1: Release	0: Closed 1: Open	Event	O	O
0103 _h ... 010F _h	Reserved for standardization					
0110_h-011F_h Rear Vehicle Storage Variants category (the vehicle should support one of the following options if the vehicle has a trunk)						
0110 _h	Manual Trunk/Decklid/Liftgate/Tailgate Controls	1: Release	0: Closed 1: Open	Event	O	O
0111 _h	Power Trunk/Decklid/Liftgate/Tailgate Controls (with confirmation for close)	0: Close 1: Open	0: Closed 1: Open	Open: enduring without confirmation, Close: enduring with confirmation	O	O

Function id	Function	Possible action ids	Possible function status values (Note 1)	Event-based or enduring action	Device (Note 2)	Vehicle (Note 2)
0112 _h	Power Trunk/Decklid/ Liftgate/ Tailgate Controls (without confirmation for close)	0: Close 1: Open	0: Closed 1: Open	enduring without confirmation	O	O
0113 _h ... 011F _h	Reserved for standardization					
0120_h-012F_h Front Vehicle Storage Variants category (the vehicle should support one of the following options if the vehicle has a trunk)						
0120 _h	Manual “Frunk” Front Trunk Controls Traditional vehicle body style	1: Release	0: Closed 1: Open	Event	O	O
0121 _h	Power “Frunk” Front Trunk Controls with confirmation Traditional vehicle body style	0: Close 1: Open	0: Closed 1: Open	Open: enduring without confirmation, Close: enduring with confirmation	O	O
0122 _h	Power “Frunk” Front Trunk Controls without confirmation Traditional vehicle body style	0: Close 1: Open	0: Closed 1: Open	Open/Close: enduring without confirmation	O	O
0123 _h ... 012F _h	Reserved for standardization					
0130_h-0140_h Vehicle Power Doors Variants category (if supported by vehicle)						
0130 _h	Power Front Left with Confirmation	0: Close 1: Open	0: Closed 1: Open	enduring with confirmation	O	O
0131 _h	Power Front Left without confirmation	0: Close 1: Open	0: Closed 1: Open	enduring without confirmation	O	O
0132 _h	Power Front Right with Confirmation	0: Close 1: Open	0: Closed 1: Open	enduring with confirmation	O	O
0133 _h	Power Front Right without confirmation	0: Close 1: Open	0: Closed 1: Open	enduring without confirmation	O	O

Function id	Function	Possible action ids	Possible function status values (Note 1)	Event-based or enduring action	Device (Note 2)	Vehicle (Note 2)
0134 _h	Power Rear Left with Confirmation	0: Close 1: Open	0: Closed 1: Open	enduring with confirmation	O	O
0135 _h	Power Rear Left without confirmation	0: Close 1: Open	0: Closed 1: Open	enduring without confirmation	O	O
0136 _h	Power Rear Right with Confirmation	0: Close 1: Open	0: Closed 1: Open	enduring with confirmation	O	O
0137 _h	Power Rear Right without confirmation	0: Close 1: Open	0: Closed 1: Open	enduring without confirmation	O	O
0138 _h ... 013F _h	Reserved for standardization					
0140_h-0150_h Vehicle Power Windows / Roof (if supported by vehicle)						
0140 _h	Power Windows	0: Close 1: Open	0: Closed 1: Open	enduring with confirmation	O	O
0141 _h	Power roof	0: Close 1: Open	0: Closed 1: Open	enduring with confirmation	O	O
0142 _h ... 05FF _h	Reserved for standardization					
0600 _h to FFFF _h	Vehicle OEM-proprietary functions					
Note 1: Possible function status values: Range from 00 _h to EF _h , for values F0 _h to FF _h (see Table 19-80) Note 2: Device and Vehicle codes: Mandatory, Optional, Conditional						

1 Generic function status values shall be used to indicate errors or the absence of a certain
2 function. These function status values are defined in Table 19-80.

3 *Table 19-80: Definition of Standardized Function/Execution Status Values to Indicate the (Temporary)
4 Unavailability of Function/Execution or Errors.*

Function status value	Possible function status values
F0 _h	Function not supported by vehicle
F1 _h	Action not supported by vehicle
F2 _h	Function temporarily not available
F3 _h	Action temporarily not available
F4 _h	Function not purchased
F5 _h	Action not purchased
F6 _h	Insufficient entitlements

Function status value	Possible function status values
F7 _h	Use of wrong RKE security policy for requested action
F8 _h to FD _h	Reserved
FE _h	Generic error, no further description available
FF _h	Reserved, shall be used if action is supported but function has no explicit status

1 RKE Request SubEvent

2 The RKE Request SubEvent's content, described in Table 19-81, is encoded as TLV. The RKE
3 Request SubEvent may be sent:

- 4 1. in response to user's request to perform one of the defined RKE features (tag 7F70_h and
5 tags 7F76_h/7F77_h for enduring actions).
- 6 2. to subscribe to vehicle function status updates (tag 7F73_h).
- 7 3. to get the current vehicle functions' status (tag 7F74_h).

8 *Table 19-81: Definition of RKE Request SubEvent Templates.*

Tag	Length (bytes)	Description	Field Is
7F70 _h	7	Request RKE action. Only present, when RKE action shall be triggered.	C
80 _h	2	Function id	M
81 _h	1	Action id	M
7F73 _h	variable	Subscribe to vehicle function status events. Unsubscribing from all status updates is possible by setting "from function id" and "to function id" to 0. Tag is only present, when subscription change shall be performed.	C
30 _h	variable	Sequence of function ids. Only present, if vehicle has subscription change to the list of function status	C
84 _h	2	From function id	M
85 _h	2	To function id	M
86 _h	0	If present, vehicle shall send an immediate update of supported function states in this range.	C
7F74 _h	variable	Get status update for function ids. Only present, when explicit status updates shall be requested.	C
30 _h	variable	Sequence of function ids. Only present, if status for a list of function ids shall be requested.	C
80h	variable	Function id	M
86 _h	0	If present, vehicle shall send a full status update of all supported functions and their corresponding states.	C
7F76 _h	variable	Continue enduring RKE action. Only present, when enduring RKE action is in progress.	C
80 _h	2	Function id	M
81 _h	1	Action id	M
88 _h	variable	Arbitrary data (max. 64 byte)	O
7F77 _h	7	Stop enduring RKE action. Only present, when the enduring RKE action in progress shall be stopped.	C
80 _h	2	Function id	M
81 _h	1	Action id	M

- 1 **Head Unit SubEvents**
2 The Head Unit SubEvent is generated to indicate the final success or failure of the transaction or
3 to signal application-specific codes to the device. The Head Unit SubEvent codes are described
4 in Table 19-82 and Table 19-83.

5 *Table 19-82: Definition of Headunit_Pairing_Status and its Session_Status.*

Parameter	Length (bytes)	Value	Description
Headunit_Pairing_Status	1	0x00 - 0xFF	See Table 19-83 for all head unit pairing status codes

6 *Table 19-83: List of Session_Status for Head Unit SubEvent.*

Session_Status	Code	Description
Headunit_pairing_success	0x00	Head Unit Pairing is successful
Headunit_pairing_fail	0x01	Head Unit Pairing fails
Reserved	0x02-0xFF	Reserved for future

7 **19.4 Time Synchronization**

8 The time synchronization shall be mandatory for the device. The time synchronization support is
9 strongly recommended for the vehicle due to performance gains such as improved latency and
10 power saving benefit for the vehicle.

11 **19.4.1 Definitions**

12 The following definitions are used for this subsection:

13 *19.4.1.1 Device UWB Clock and UWB Device Time*

14 Device UWB Clock is defined as the UWB clock of the device as it appears to the vehicle.

15 UWB Device Time is defined as the timestamp on the Device UWB Clock.

16 **The appearance of the Device UWB Clock to a vehicle can be defined or undefined:**

- 17 1. “Defined”
 - 18 o Device UWB Clock becomes “defined” to the vehicle, once a successful Bluetooth LE
19 Timesync procedure with valid UWB_Device_Time has been performed
 - 20 o Device UWB Clock becomes “defined” to the vehicle, once an UWB packet (of a
21 ranging session with this device) has been received
 - 22 o Once defined, vehicle expects that UWB device clock is operated in a consistent manner
23 and the characteristics and requirements below hold
- 24 2. “Undefined”
 - 25 o Device UWB Clock appears “undefined” to the vehicle prior to a successful Bluetooth
26 LE Timesync.
 - 27 o Note: This is the case if the Device UWB clock has not been started.
 - 28 o Device UWB Clock becomes “undefined”, if there is no UWB and no Bluetooth LE
29 connection for more than 30 seconds.

- 1 ○ Device UWB Clock becomes “undefined”, if a ranging session is suspended.
- 2 ○ Note: A ranging session may be suspended automatically, if Bluetooth LE connection is lost for more than 30 seconds.
- 3 ○ Device UWB Clock becomes “undefined”, if a ranging session is terminated.
- 4 ○ Note: Session is terminated if 12 hours lifetime is exceeded or the STS-Index range is exhausted.

7 Characteristics of Device UWB Clock

- 8 1. UWB clock is UWB ranging session specific.

9 *Note:* Device may use a common clock for all its ranging sessions and maintain this clock even if a session is suspended.

- 11 2. The Device UWB Clock may have any value when entering “defined” state (within its specified range). For example, the device may set UWB_Device_Time=0 upon first valid Bluetooth LE Timesync procedure.
- 14 3. During an active ranging session, the Device UWB Clock is identical to the initiator clock as specified in Section [20.2](#).
- 16 4. Suspending a ranging session shall put the UWB clock into an undefined state and reset it to an arbitrary value for session recovery.

18 *Note:* UWB_Time0 for resuming a ranging session may be smaller than UWB_Device_Time at ranging session suspend.

- 20 5. Device UWB clock should have a granularity of 1 μ sec or smaller, considering that the format in the timesync message and the RSS-RS / RR-RS/ CRR-RS message is defined with 1 μ sec resolution.
- 23 6. The range of the UWB clock is limited to 0 ... $2^{64}-1$ μ sec due to the format in the timesync message and the RSS-RS / RR-RS/ CRR-RS message.
- 25 7. Device UWB clock does not depend on the existence of a BT connection as long the ranging session is not suspended.

27 *Note:* A short Bluetooth reconnection that does not trigger a ranging suspension during an active ranging session shall trigger Bluetooth LE Timesync procedure 0. But the short Bluetooth reconnection shall not impact the operational state of Device UWB clock (while in “defined” mode).

31 Requirements for Device

- 32 1. Device shall be able to map a timing reference (see Section [19.3.3](#)) onto the Device UWB Clock and generate a UWB_Device_Time timestamp. The term “to map” here means, to take a timestamp from the Device UWB Clock in the moment of the Bluetooth LE Timesync defined timing reference on the air interface, or equivalently, compute the value that was held by the Device UWB Clock during the timing reference by means of an OEM specific method.
- 38 2. Device shall support at least one successful Bluetooth LE Timesync procedure before session start/resume.

- 1 3. Device shall support all vehicle triggered timesync procedures (Bluetooth LE Timesync
2 procedure 1) during an active session.
- 3 4. All (valid) UWB_Device_Time values provided in the timesync messages shall be consistent
4 clock states in accordance with the MAC grid as specified in Section [20.2](#).
 - 5 *Note:* In particular, the value of the UWB clock at session start shall be UWB_Time0 (as
6 announced in the RSS-RS/RR-RS message).
- 7 5. UWB_Device_Time shall be presented as the clock state of a 64-bit counter with 1μsec
8 resolution; the 64-bit counter shall wrap around at $2^{64}-1$.

9 **Tolerance of Device UWB Clock frequency:**

- 10 1. Before session start (or before session resume):
 - 11 ○ Maximum tolerance of UWB clock frequency is Device_max_PPM, as included in
12 timesync message
- 13 2. After session start (during active ranging session):
 - 14 ○ Maximum tolerance of UWB clock frequency is Device_max_PPM, as included in
15 timesync message and shall be less than ±100 ppm as specified in Section 6.9.7.2 of [\[33\]](#).

16 *19.4.1.2 Terminology: Clock Offset, Skew, Drift*

17 **The terminology used in this section follows Section 10 of [\[37\]](#).**

18 *Clock state:* Value of a clock x at time t=T

19 *Clock offset:* Clock offset is the difference in clock states between a clock x and the true time at
20 an arbitrary point in time T.

21 *Relative clock offset:* Relative clock offset is the difference in clock states between a clock x and
22 a clock y, at an arbitrary point in time T.

23 *Clock skew:* Clock skew is the difference in clock frequency to the nominal clock frequency.

- 24 • Clock skew can be viewed as first derivative in time of clock offset.
- 25 • Clock skew normalized to the nominal frequency and multiplied by 10^6 is “clock skew in
26 ppm”.

27 *Relative clock skew:* Relative clock skew is the difference in clock frequencies of clock x and
28 clock y.

- 29 • Relative clock skew can be viewed as the first derivative in time of relative clock offset (the
30 difference in clock frequencies at certain point in time).

31 *Clock drift:* Clock drift is the derivative of clock skew (the variation of clock frequency over
32 time). Clock drift is assumed to be negligible for the purpose of this specification.

33 **Figures of merit for timesync**

- 34 • Vehicle UWB Clock (the UWB clock in a vehicle anchor = responder) is clock x.

- Device UWB Clock is the clock y, and is the reference clock that defines the UWB MAC grid, and to which the responder needs to synchronize.
- Timesync procedure is used to determine the relative clock offset between Device UWB clock and Vehicle UWB clock for the time of the BLE event on the air interface
- The worst-case clock skew `Device_max_PPM` of the device UWB clock (relative to true time) and the worst-case clock skew of the vehicle can be used to determine an upper bound for the relative clock offset of a future event (like session start).
- Vehicle may estimate the relative clock skew and compensate for it, to minimize the relative clock offset of a future event (like session start); for the time interval of interest the clock drift is assumed to be zero.

19.4.2 General Description

The Bluetooth connection serves as a common time domain between the vehicle and device and is used to synchronize the Bluetooth-connected UWB transceiver by mapping a shared BT event into the UWB clock domain. Synchronization methods are shown in Figure 19-7.

UWB Packet reception:

The reception of an UWB packet by the vehicle should allow to perform a better synchronization than the Bluetooth LE Timesync synchronization method described in this chapter.

Because this gives an anchor the precise position within the MAC grid (see Section 20.3), which it may be used to adjust the relative clock offset between device and anchor. This subsection describes the “Bluetooth LE Timesync” synchronization which happens before the reception of an UWB packet by the vehicle.

Note: An anchor may set its UWB-RX into scanning mode and wait for the reception of a packet without using any “Out of band” synchronization. This is always possible, at the cost of increased power consumption at the vehicle and other potential constraints.

Bluetooth LE Timesync:

In the absence of UWB Packet reception, this section describes a time synchronization over Bluetooth.

Time synchronization over Bluetooth

Time synchronization over Bluetooth, referred as “Bluetooth LE Timesync”, is an out-of-band synchronization method that is carried out between the device and one particular anchor. There are two types of “Bluetooth LE Timesync” procedure (see to Section 19.4.4):

- “Bluetooth LE Timesync at Bluetooth connection”, also referred as “Procedure 0”
- “Bluetooth LE Timesync triggered by vehicle”, also referred as “Procedure 1”

References in this chapter to “Bluetooth LE Timesync” apply to both “Bluetooth LE Timesync at Bluetooth connection” and “Bluetooth LE Timesync triggered by vehicle”

Vehicles may use “Bluetooth LE Timesync” to synchronize on the start of the UWB ranging session.

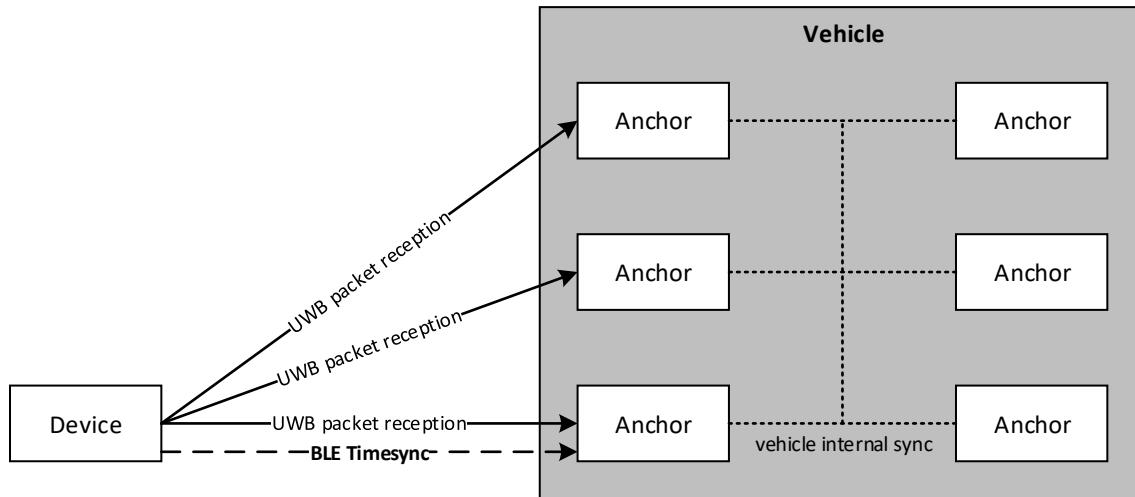
After start of the session the vehicle may use “Bluetooth LE Timesync” to re-synchronize on the MAC grid if UWB connection was lost for a longer period.

1 **Vehicle internal synchronization**

2 Vehicle internal synchronization may be used to distribute device synchronization (achieved by a
3 particular anchor based on UWB packet reception or “Bluetooth LE Timesync”) between the
4 anchors. The mechanism to achieve the vehicle internal synchronization is implementation
5 specific and is out of scope for this specification.

6 The different synchronization methods are shown in Figure 19-7.

7 *Figure 19-7: Synchronization Methods.*



8

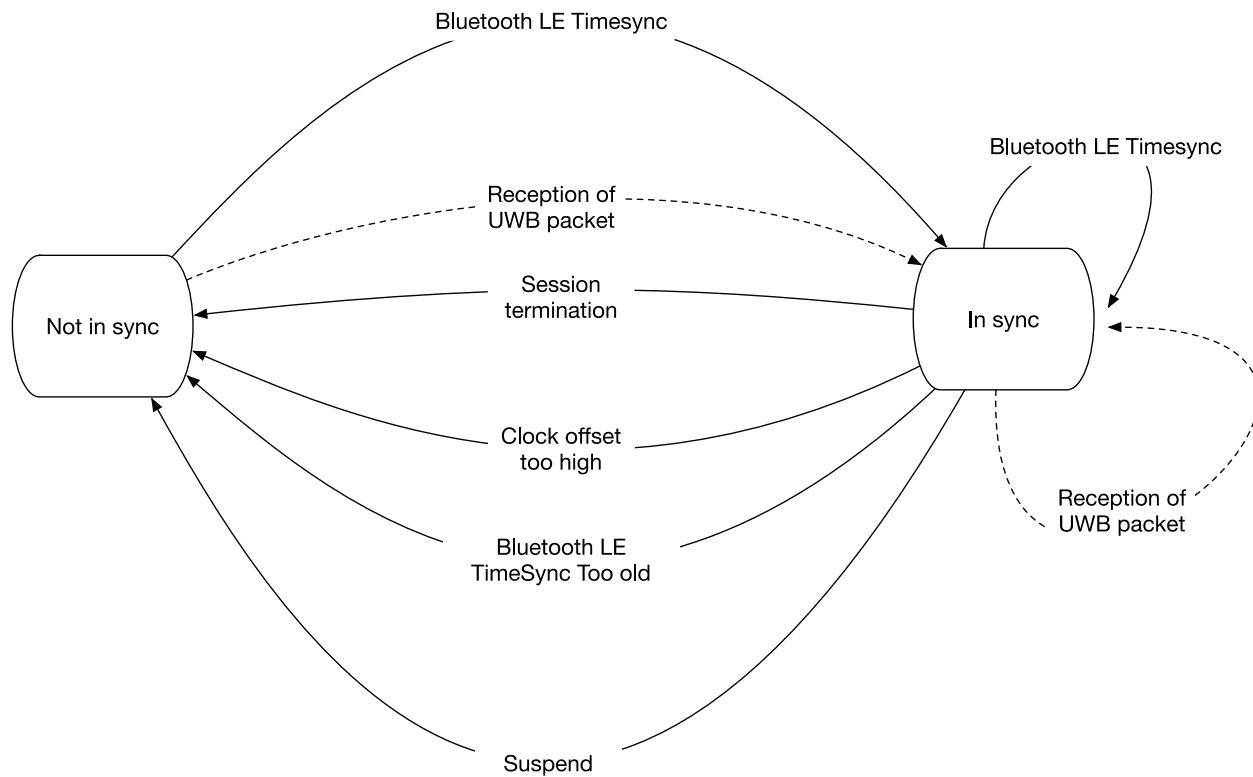
9 There are two states for the time synchronization between device and anchor as shown in Figure
10 19-8. Those states are only known by the vehicle.

- 11 • “not in sync”: Clock offset of anchor UWB clock vs device UWB clock is unknown, or
12 potential offset is larger than 1ms
- 13 • “in sync”: Clock offset of anchor UWB clock vs device UWB clock is better than 1ms

14 *Note:* The limit for the relative clock accuracy that defines the “in sync” state is up to the vehicle
15 implementation.

1

Figure 19-8: Synchronization state between device and anchor.



2

3 Note: This state diagram defines the pre-conditions for time synchronization and assumes a
4 vehicle implementation which establishes this time synchronization in all anchors. In a real
5 implementation there could be further boundary conditions, that might hinder time
6 synchronization, although the conditions from this state diagram are given.

7 Note: Reception of an UWB packet shall lead immediately to the “In sync” state. This
8 synchronization via reception of an UWB packet is indicated as dashed line in the state diagram
9 is referred as “In-band” UWB synchronization. However, the scope of this Section is “Out-of-
10 band” UWB synchronization via Bluetooth LE Timesync.

11 If “in sync”, the vehicle shall be able to predict events like session start or a position within the
12 MAC grid (for an active ranging session) with some uncertainty.

13 The uncertainties are caused by clock skew between device and vehicle UWB clock,
14 UWB_Device_Time uncertainty in case of “Bluetooth LE Timesync”, and uncertainties on
15 vehicle side.

16 If no ranging session was started or recovered within 10 seconds after a successful Bluetooth LE
17 Timesync, the Device UWB Clock becomes “Not in sync”.

18 The vehicle shall only trigger “Bluetooth LE Timesync” if conditions in condition set A or
19 condition set B is met:

20 Condition Set A:

- 1 1. Condition A.1: A ranging session was needed in the last 150s or is currently running.
2 This can be signaled by Device Ranging Intent as sent by device to vehicle, user physical
3 intent on vehicle (e.g. touching door handle, engine start), or when Session Ranging
4 Setup is to be performed (e.g. during owner pairing, first friend approach, passive entry).
5 2. Condition A.2: Previous successful “Bluetooth LE TimeSync” was done (either by device
6 or vehicle) more than 150s ago.

7 Condition Set B:

- 8 1. Condition B.1: A ranging session is needed. This can be signaled by Device Ranging
9 Intent as sent by device to vehicle, user physical intent on vehicle (e.g. touching door
10 handle, engine start), or when Session Ranging Setup is to be performed (e.g. during
11 owner pairing, first friend approach, passive entry).
- 12 2. Condition B.2: No active ranging session is on-goingNone of the vehicle’s UWB anchors
13 produced a valid ToF measurement within 10 seconds since the last “Bluetooth LE
14 TimeSync”
- 15 3. Condition B.3: Previous successful “Bluetooth LE TimeSync” was done (either by device
16 or vehicle) more than 10s ago.

17 Device clock skew is upper bounded by the parameter Device_max_PPM.

18 The vehicle may estimate the relative UWB clock skew (UWB clock frequency offset between
19 vehicle and device) using the information from the most recent and one or more previous
20 timesync procedures. In this case, the vehicle may use the estimated clock skew and compensate
21 for it, instead of applying an upper bound with Device_max_PPM.

22 After the first successful UWB packet is received by vehicle, the vehicle should use this to
23 resynchronize the UWB clocks between device and vehicle. Therefore, the vehicle should apply
24 worst-case PPM based on the smallest duration between:

- 25 • Latest successful UWB packet exchange until now
- 26 • Latest time sync until now

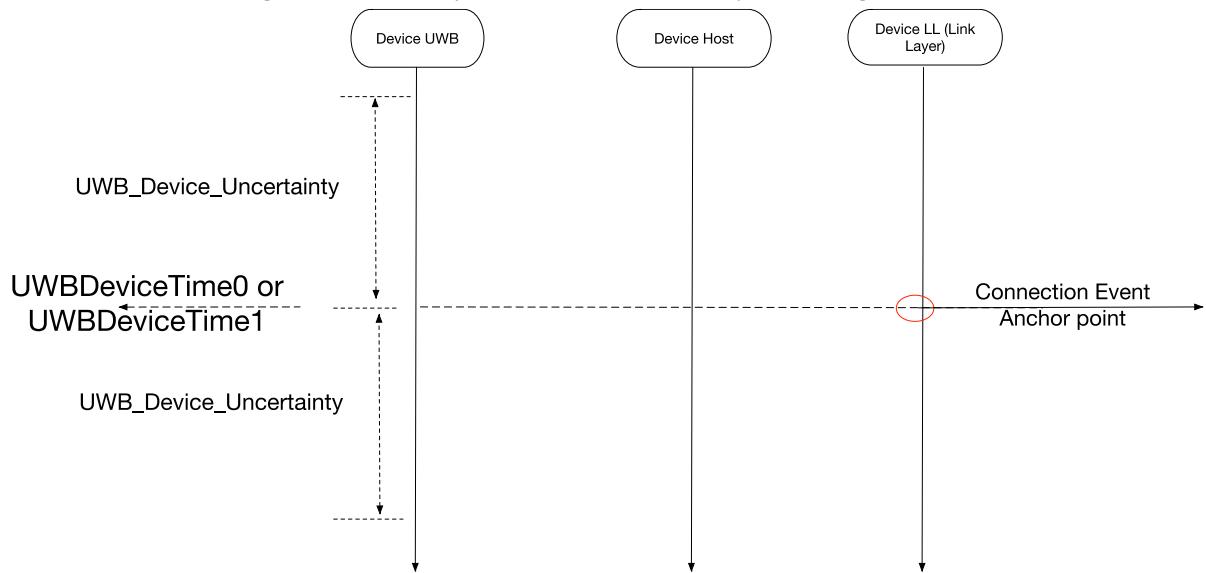
27 **19.4.3 *Device Uncertainty***

28 UWB_Device_Uncertainty is the time uncertainty between the connection event Anchor point
29 and UWB Clock domain mapping (see Section [19.3.3](#) for definition)

30 Figure 19-9 shows the uncertainties on the device side.

1

Figure 19-9: Time Synchronization Uncertainty Flow Diagram.



2

3 19.4.4 “Bluetooth LE Timesync” Procedures.

4 In this subsection, the following times are defined:

- 5 • *UWBDeviceTime0*: UWB time clock domain captured at Procedure 0.
- 6 • *UWBDeviceTime1*: UWB time clock domain captured at Procedure 1.
- 7 • *DeviceEventCount0*: Event count for the connection interval after Bluetooth connection
8 establishment. This is the first connection interval, and the value shall be 0.
9 *UWBDeviceTime0* is taken at the anchor point of this connection interval.
- 10 • *DeviceEventCount1*: Event count for the connection interval containing the
11 *LL_PHY_UPDATE_IND* message sent from device to the vehicle during the “Bluetooth LE
12 TimeSync” procedure 1. *UWBDeviceTime1* is taken at the anchor point of this connection
13 interval.

14 19.4.4.1 Procedure 0: Bluetooth LE Timesync at Bluetooth Connection

15 The host (device) shall capture the anchor point of the first available connection event after the
16 Bluetooth connection establishment. The device shall map this point in time also into its UWB
17 clock domain, (which corresponds to *UWBDeviceTime0*) and the device shall send a timesync
18 message over Bluetooth LE. In case of retransmission for the connection event 0, the vehicle
19 should consider only the latest successful LL message.

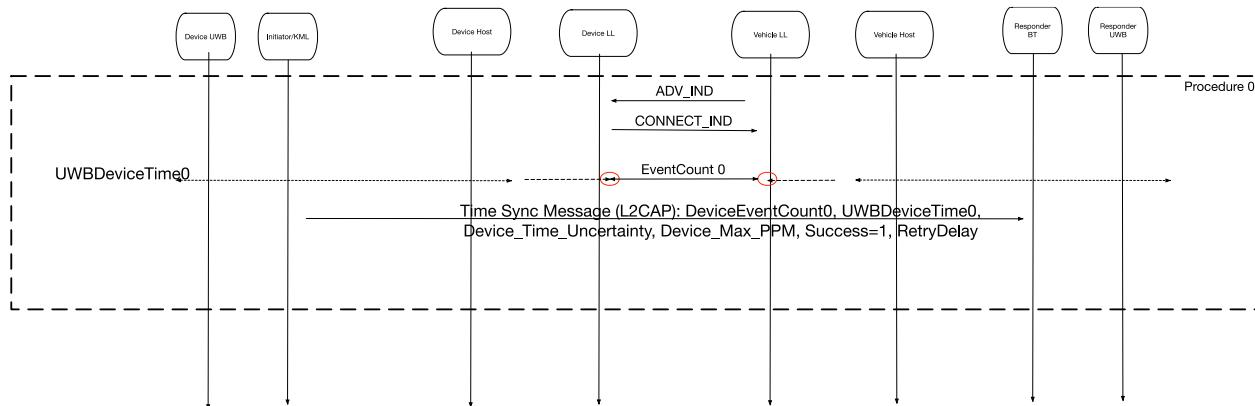
20 The timesync message is defined in Section [19.3.3](#).

21 The timesync message parameter *DeviceEventCount* should have a value of 0 or close to it.

22 A successful “procedure 0” shall return a timesync message parameter *Success* value of 1. The
23 timesync message parameter *RetryDelay* has no meaning.

1

Figure 19-10: Successful Bluetooth LE timeSync at BT connection (Procedure 0).



2

3 19.4.4.2 Procedure 1: Bluetooth LE Timesync Triggered by Vehicle

4 When the vehicle determines a time synchronization is required, the host (vehicle) shall send an
5 “LE Set PHY” command which triggers sending the “LL PHY REQ” to the device. The device
6 should respond to the “LL PHY REQ” by sending “LL_PHY_UPDATE_IND”.

7 The vehicle may trigger the procedure 1 before or after the ranging session start. The vehicle
8 shall trigger the procedure 1 only after Bluetooth connection is established and encryption is
9 completed.

10 The time synchronization process on the device side is initiated once the device receives the
11 message “LL_PHY_REQ” sent from the vehicle. The time synchronization shall happen at the
12 anchor point of the connection event that contains the response message
13 “LL_PHY_UPDATE_IND” of the device. This time as DeviceEventCount1.

14 The device shall map this point in time into its UWB clock domain (UWBDDeviceTime1). The
15 exact mechanism of mapping the timing reference is out of scope for this specification. In case of
16 retransmission for “LL_PHY_UPDATE_IND”, the vehicle should consider only the latest
17 successful “LL_PHY_UPDATE_IND” LL message.

18 After mapping the timing reference into its UWB clock domain, the device shall send a timesync
19 message over Bluetooth LE. The timesync message is defined in Section 19.3.3.

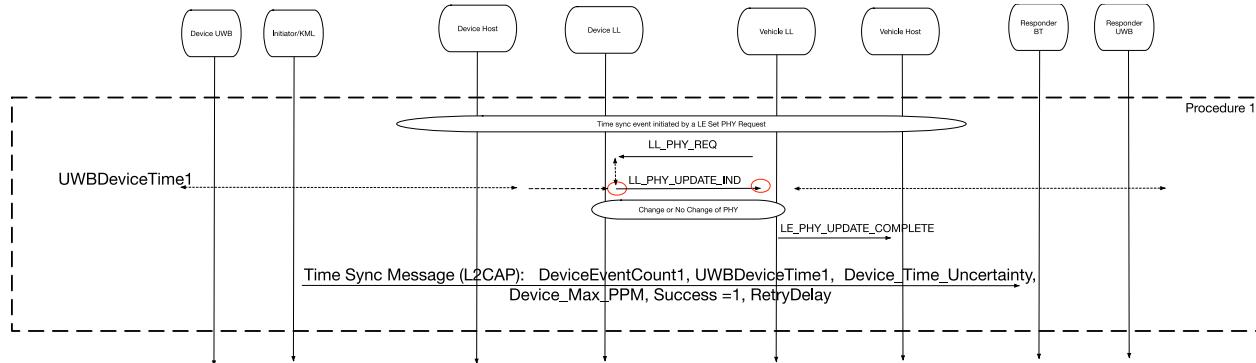
20 The vehicle should map the timing reference (anchor point) of a connection event into its UWB
21 clock domain (UWBVehicleTime1).

22 The vehicle maps its UWB clock in two different ways:

- 23 1. **Same connection event as device:** Vehicle has mapped the anchor point of the same
24 connection event as referenced in the timesync message.
 - 25 a. In this case the vehicle may directly use UWB_Device_Time from the timesync message
26 to determine the synchronization offset:
 - 27 i. $\text{SyncOffset} = \text{UWBDDeviceTime1} - \text{UWBVehicleTime1}$
 - 28 b. The use of DeviceEventCount is not necessary.
- 29 2. **Different connection event as device:** Vehicle has mapped the anchor point of a different
30 connection event VehicleEventCount1
 - 31 a. In this case the vehicle shall use DeviceEventCount to correct for the different points in
32 time and determine the synchronization offset:

- 1 i. SyncOffset = UWBDeviceTime1 + (VehicleEventCount1 – DeviceEventCount1) ×
2 Connection Interval – UWBVehicleTime1
- 3 b. Clock skew between device UWB and BLE clock domain can be neglected, if the delta of
4 the event counts for mapping is small.

5 *Figure 19-11: Successful Bluetooth LE timeSync triggered by vehicle (Procedure 1).*



6 **19.4.4.3 Unsuccessful “Bluetooth LE Timesync”**

An Unsuccessful “Bluetooth LE Timesync”, shown in Figure 19-12, is defined when the success field in the Time_Sync message (see Section 19.3.3.) is set to 0.

10 The device may return an unsuccessful “Bluetooth LE Timesync” for the following reasons:

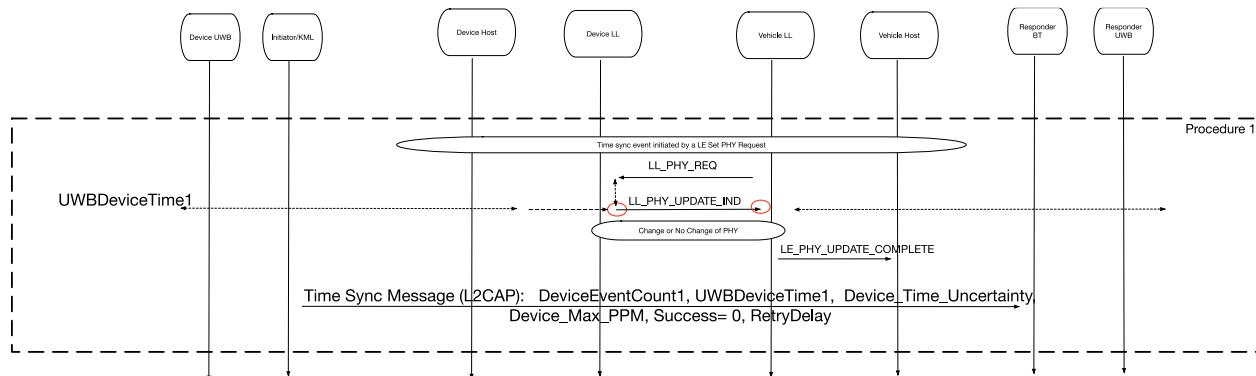
- 11 • Device UWB Clock is not available
- 12 • Device is unable to map the timing reference to its UWB clock domain.

13 If the success field is 0, the vehicle should initiate the “Bluetooth LE Timesync” procedure after
14 the delay stated in the field “RetryDelay”.

15 In this case, the following parameters in the time Sync message are undefined or not available:

- 16 • UWB_Device_time
- 17 • DeviceEventCount1
- 18 • UWB_Device_time_uncertainty
- 19 • Device_max_PPM

20 *Figure 19-12: Unsuccessful Timesync message.*



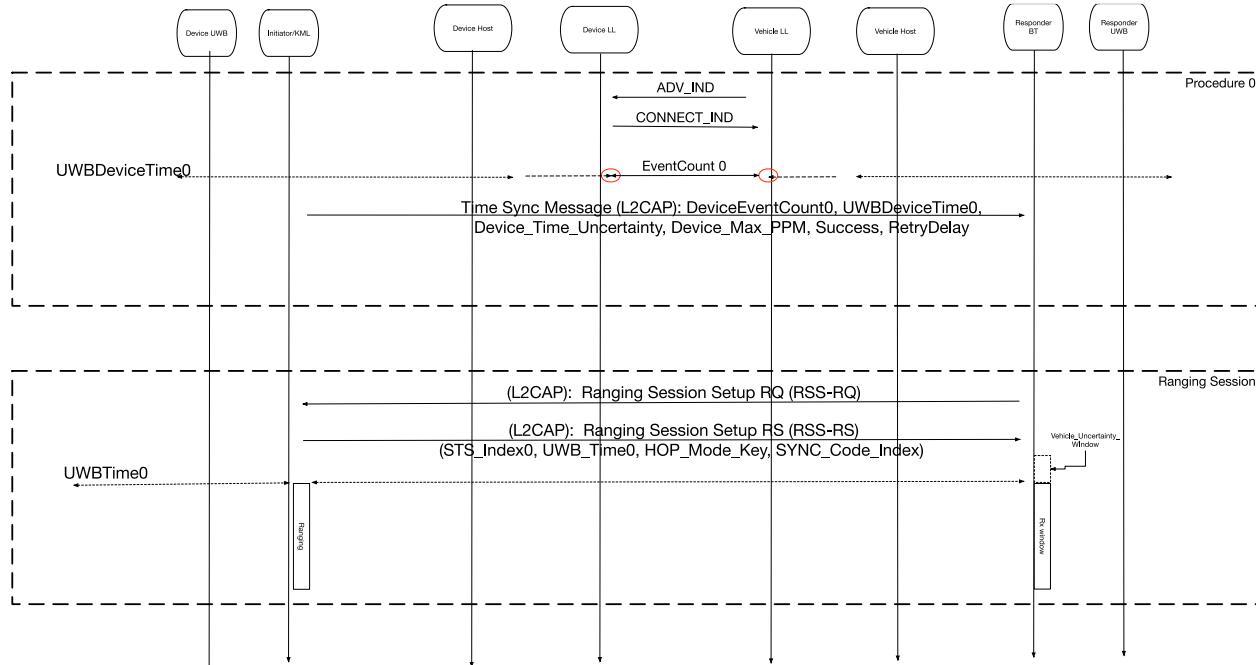
1 19.4.5 “Bluetooth LE Timesync” message flow examples

2 There are multiple combinations possible for Procedure 0 and Procedure 1 before a ranging
3 session is setup.

4 19.4.5.1 Example 1: “Bluetooth LE Timesync” with Procedure 0 only

5 In Example 1, shown in Figure 19-13, the procedure 0 is successful and procedure 1 is omitted.

6 Figure 19-13: Bluetooth LE Timesync message flow example 1.



7 19.4.5.2 Example 2: “Bluetooth LE Timesync” with Procedure 0 and Procedure 1

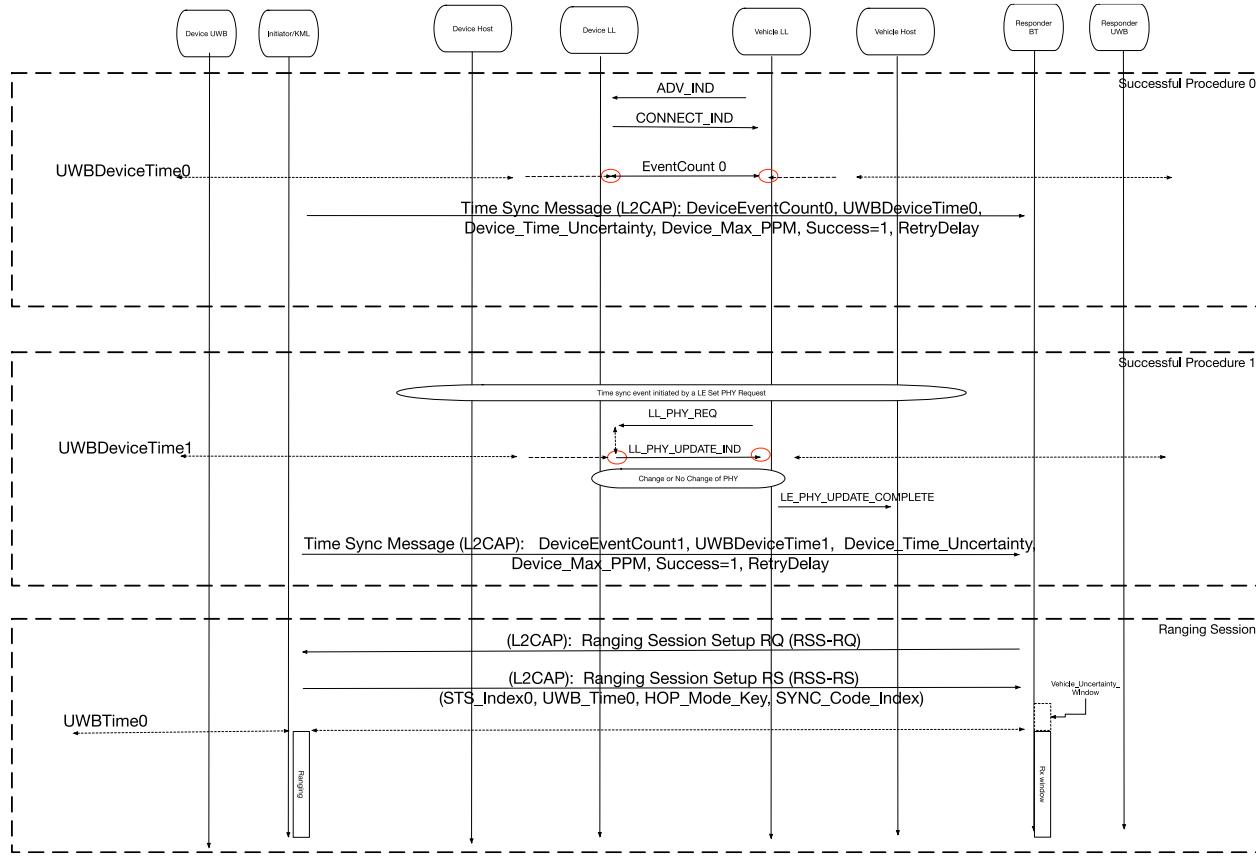
8 In Example 2, both Procedure 0 and Procedure 1 are successful.

9 The internal operations of the device and vehicle for obtaining UWB timestamps according to
10 the BLE timestamps are implementation specific.

11 Figure 19-14 describes the “Bluetooth LE Timesync” message flow with an example of internal
12 operations. In this figure, the ranging session is described as a block as illustration only. It is
13 possible for the vehicle to initiate Procedure 1 any time during the ranging session.

1

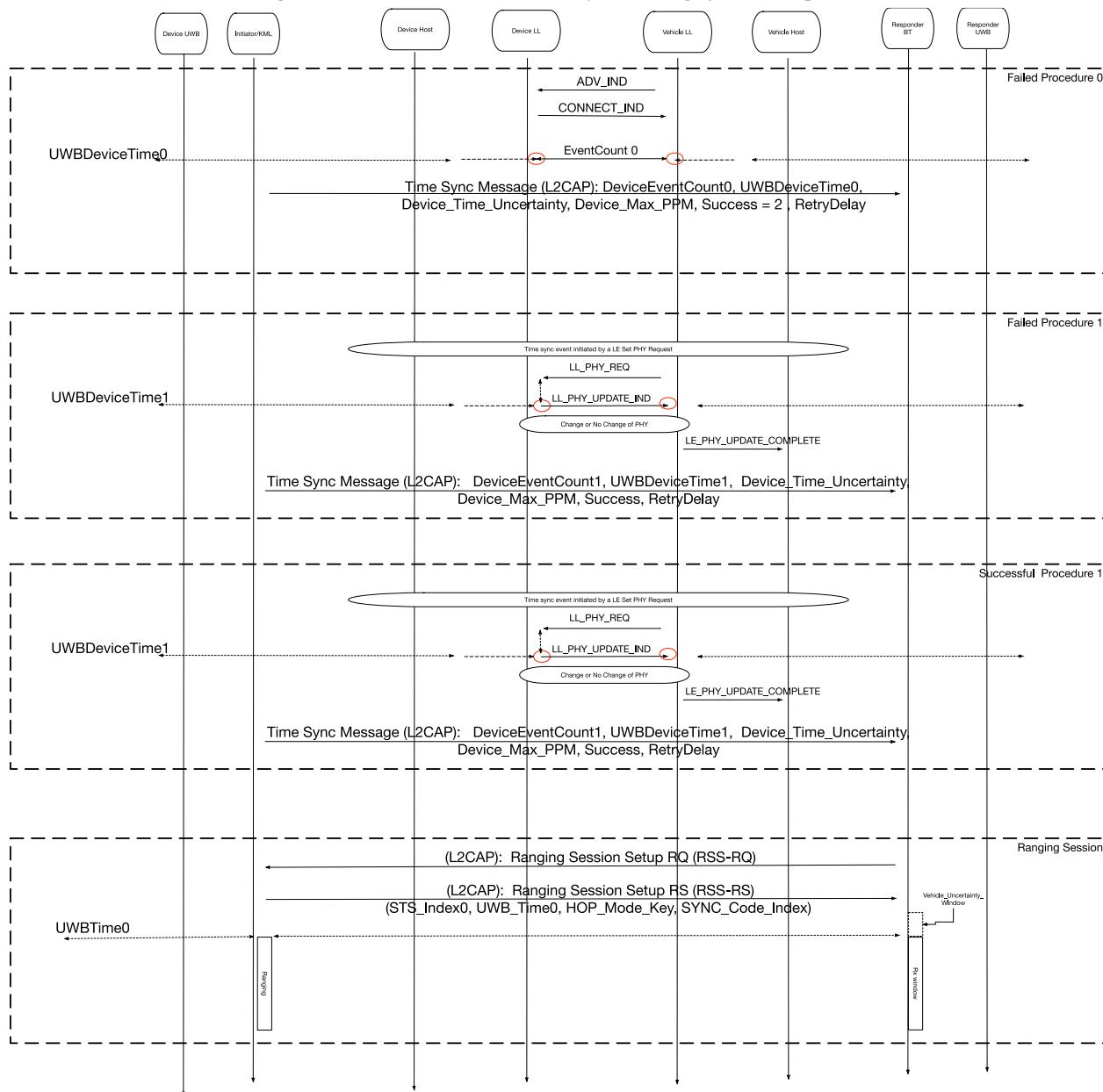
Figure 19-14: Time Synchronization flow diagram example 2.



- 3 19.4.5.3 Example 3: “Bluetooth LE Timesync” with Procedure 1 only
- 4 In Example 3, shown in Figure 19-15, the Procedure 0 is unsuccessful and only the second
- 5 Procedure 1 is successful.
- 6 The second Procedure 1 shall be initiated after the retryDelay provided in the first Failed
- 7 Procedure 1.
- 8 If the device supports Procedure 0, it should reply with Success = 2 (the “Bluetooth LE
- 9 Timesync” procedure has failed and the Procedure 0 is not available) to avoid confusion for the
- 10 vehicle.

1

Figure 19-15: Bluetooth LE Time Sync message flow example 3.



2

3 19.4.6 Conditions for a “Bluetooth LE Timesync” procedure

4 A successful “Procedure 1” is defined as the grouping of the following LL messages and DK messages:

- 5 • LL_PHY_REQ initiated by vehicle
- 6 • LL_PHY_UPDATE_IND sent by device
- 7 • Time_Sync message with success field = 1 (see Section [19.3.3](#)).

1 **Frequency of Procedure 1:**

- 2 • If consecutive successful “Procedure 1” is equal or more than 60 seconds apart, the device
3 shall support it.
4 • If consecutive successful “Procedure 1” is less than 60 seconds apart, device support is
5 optional.

6 There should be at least one successful procedure before the Ranging Session Setup Request and
7 Ranging Recovery Request messages.

8 UWB_Time0 (provided in the Ranging Session Setup Response or Ranging Recovery Response
9 message) shall be consistent in time to the reference UWB_Device_Time. Since Device UWB
10 clock enters "defined" mode after successful Procedure 1, any Bluetooth reconnection after the
11 LL_PHY_UPDATE_IND and before the ranging shall not impact the UWB_Time0.

12 **19.5 Digital Key – Flows**

13 **19.5.1 Owner Pairing Flow**

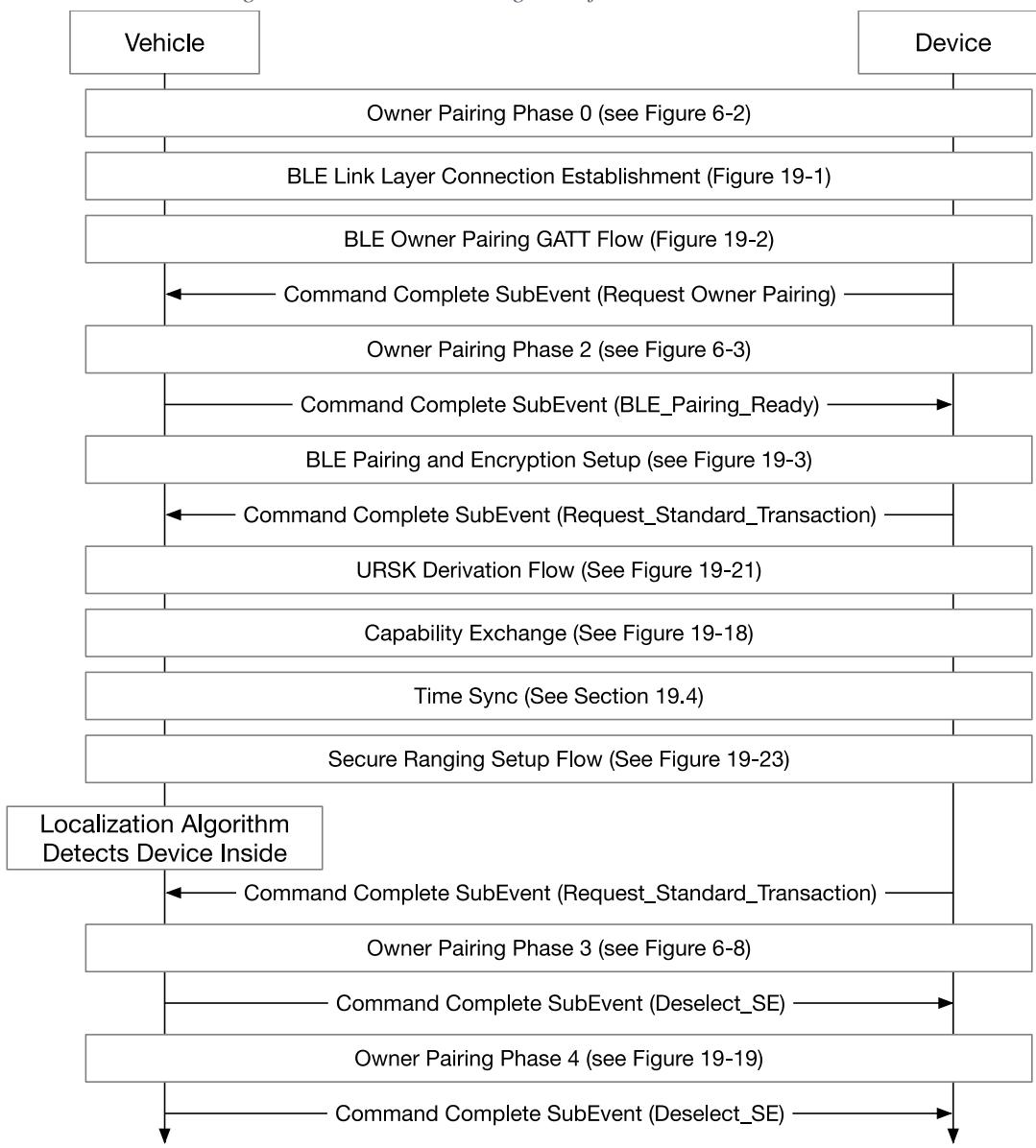
14 Owner pairing can either be triggered automatically by the vehicle initiating Owner Pairing
15 Connection Establishment as defined in Section [19.2.1](#) or manually by user going into the
16 vehicle system menu and initiating owner pairing. In either case, before initiating owner pairing
17 process, vehicle ensures that all of its policies have been met. Below are some of the examples
18 for what these policies might include:

- 19 1. Physical key fob inside the vehicle
20 2. No owner device paired with vehicle
21 3. Digital Key feature has been enabled

22 Owner pairing over Bluetooth LE leverages similar exchange that takes place over NFC with
23 minor modifications and additional steps to the flow as described in this section and shown in
24 Figure 19-16. NFC related procedures (NFC polling and link setup, NFC link teardown, NFC
25 reset) in Owner Pairing Phase 2, Phase 3 and Phase 4 shall not apply to owner pairing over
26 Bluetooth LE. For Bluetooth LE/NFC, the owner pairing may start over NFC as described in
27 Section 6 and followed by Bluetooth LE Activation flow as described in Section [19.5.10](#). Owner
28 pairing shall be performed over NFC if the vehicle or device does not support secure ranging.

1

Figure 19-16: Owner Pairing Flow for Bluetooth LE/UWB.



2

3

4 **Owner Pairing Phase 0: Preparation**

5 This phase prepares device and vehicle for owner pairing as described in Section [6.3.1](#).

6 **Bluetooth LE Link Layer Connection Establishment**

7 In this setup process, the device goes through Bluetooth LE connection, discover DK services
8 and characteristics and setup L2CAP channel as described above in Figure 19-1.

9 **Bluetooth LE Owner Pairing GATT Flow**

10 In this setup process, device goes through Bluetooth LE Owner Pairing GATT Flow as described
11 in Figure 19-2.

Request Owner Pairing

After connection to the vehicles Bluetooth LE owner pairing advertisement, the device shall send the “Request_Owner_Pairing” SubEvent notification (see Table 19-73) to start owner pairing.

Owner Pairing Phase 2: SPAKE2+ Flow and Certificate Exchange

This phase enables SPAKE2+ based secure channel and certificate exchange between vehicle and device. For more details on Phase 2, refer to Section 6.3.3. To perform Phase 2 exchange over Bluetooth LE, each APDU command and response shall be encoded as below.

Message Type: Framework message

Message:

APDU Command: DK_APDU_RQ (see Section 19.3.2.1)

APDU Response: DK_APDU_RS (see Section 19.3.2.2)

In Owner Pairing Phase 2, both device and vehicle go through SPAKE2+ flow (see Figure 6-3 and Figure 6-4) to derive system keys and OOB Bluetooth LE keys as defined below. More details on SPAKE2+ and system keys derivation (HKDF functions) see Section 18.

15 Below is the set of derived system keys.

```
16    Keys = HKDF-SHA256(96, K, "SystemKeys")      // (output_len,
17                                // input_keying_material, info)
18    Kenc = Keys[0:15]                            // first 16 bytes
19    Kmac = Keys[16:31]                           // second 16 bytes
20    Krmac = Keys[32:47]                          // third 16 bytes
21    LONG_TERM_SHARED_SECRET = Keys[48:63]        // fourth 16 bytes
22    Kble_intro = Keys[64:79]                     // fifth 16 bytes
23    Kble_oob_master = Keys[80:95]                // sixth 16 bytes
```

Out of these keys, Kble_oob_master and Kble_intro are UWB solution specific keys:

1. Kble_oob_master is used as shared secret between device and vehicle which is used to derive Kble_oob to encrypt OOB pairing data during First Approach
 2. Kble_intro is used to encrypt DK Identifier during First Approach

In addition, the vehicle shall provide its “Wireless Capabilities” as part of 7F49 template defined in Table 19-84. The following wireless capability combinations (WCC) as utilized for DK Vehicle and Device operation are defined:

- WCC1: NFC only
 - WCC2: NFC and Bluetooth LE (NFC + RKE Functions)
 - WCC3: NFC, Bluetooth LE, and UWB (NFC + RKE Functions + Passive/Location-based Functions)

Devices and Vehicles may contain any combination of radios not used for DK operations.

- 1 Device shall store Kble_oob_master, Bluetooth Random Static Address of the vehicle and IRK
2 of vehicle in its database. This information shall be used during owner pairing, NFC to UWB
3 migration of owner device and friend sharing.

4 *Table 19-84: 7F49 Template.*

Template		Length (bytes)	Description	Field is												
7F49 _h		variable	7F49 template is used to provide wireless capability (NFC/UWB) and related data; this template shall be included by vehicle.	M												
	5F50 _h	0	This tag shall be included if and only if the vehicle supports NFC. Vehicle and Device shall always support NFC capability. If this tag is present, tags 5F51 and/or 7F52 may be present.	M												
	5F51 _h	0	This tag shall be included if and only if the vehicle supports UWB. If this tag is present, tags 5F50 and 7F52 shall be present.	O												
	7F52 _h	variable	<p>This tag (Bluetooth LE) shall be included if and only if either of the WCC shown below are supported by the vehicle:</p> <table border="1"> <tr> <td>WCC</td> <td>NFC</td> <td>UWB</td> <td>BLE</td> </tr> <tr> <td>WCC2</td> <td>Supported</td> <td>Not Supported</td> <td>Supported</td> </tr> <tr> <td>WCC3</td> <td>Supported</td> <td>Supported</td> <td>Supported</td> </tr> </table> <p>Length for usage during Owner pairing: 28 bytes Length for usage during Friend sharing: 64 bytes</p>	WCC	NFC	UWB	BLE	WCC2	Supported	Not Supported	Supported	WCC3	Supported	Supported	Supported	C
WCC	NFC	UWB	BLE													
WCC2	Supported	Not Supported	Supported													
WCC3	Supported	Supported	Supported													
	D0 _h	16	IRK of Bluetooth LE of the vehicle. If 7F52 tag is present, this tag shall be included	C												
	D1 _h	6	Bluetooth Random Static Address of the vehicle. If 7F52 tag is present, this tag shall be included	C												
	D2 _h	0	This Tag shall be included if vehicle supports LELR.	O												
	D3 _h	16	<p>Kble_oob is a 16-byte key used to encrypt/decrypt (AES-128-CCM) Bluetooth LE Secure OOB Pairing data which is exchanged as part of First_Approach_RQ/RS. This key is derived by owner device and vehicle and unique per Digital Key.</p> <p>This tag shall not be used during owner pairing, when transferring the “Wireless capabilities” from vehicle to device. This tag shall only be used for friend sharing.</p>	C												
	D4 _h	16	Kble_intro is a 16-byte key used to encrypt/decrypt (AES-128-CCM) DK_Identifier exchanged as part of First_Approach_RQ. This key is derived by both device and vehicle as part of owner pairing flow (see Section 19.5.1) and is same across all friends. This tag shall only be used for friend sharing.	C												

- 5 Additional tags sent by the vehicle in 7F49 template shall be ignored by the device.
6
7 The vehicle that supports Bluetooth LE/UWB shall also set the corresponding bits in Tag 46_h and
8 Tag 47_h as defined in Table 15-14.

9 Bluetooth LE Pairing & Encryption Setup Procedure

10 The Bluetooth LE Secure OOB Pairing Prep flow, shown in Figure 19-17, enables secure sharing
11 of OOB data between device and vehicle to enable Bluetooth LE Secure OOB pairing.

12 **Message Type:** Supplementary Service message

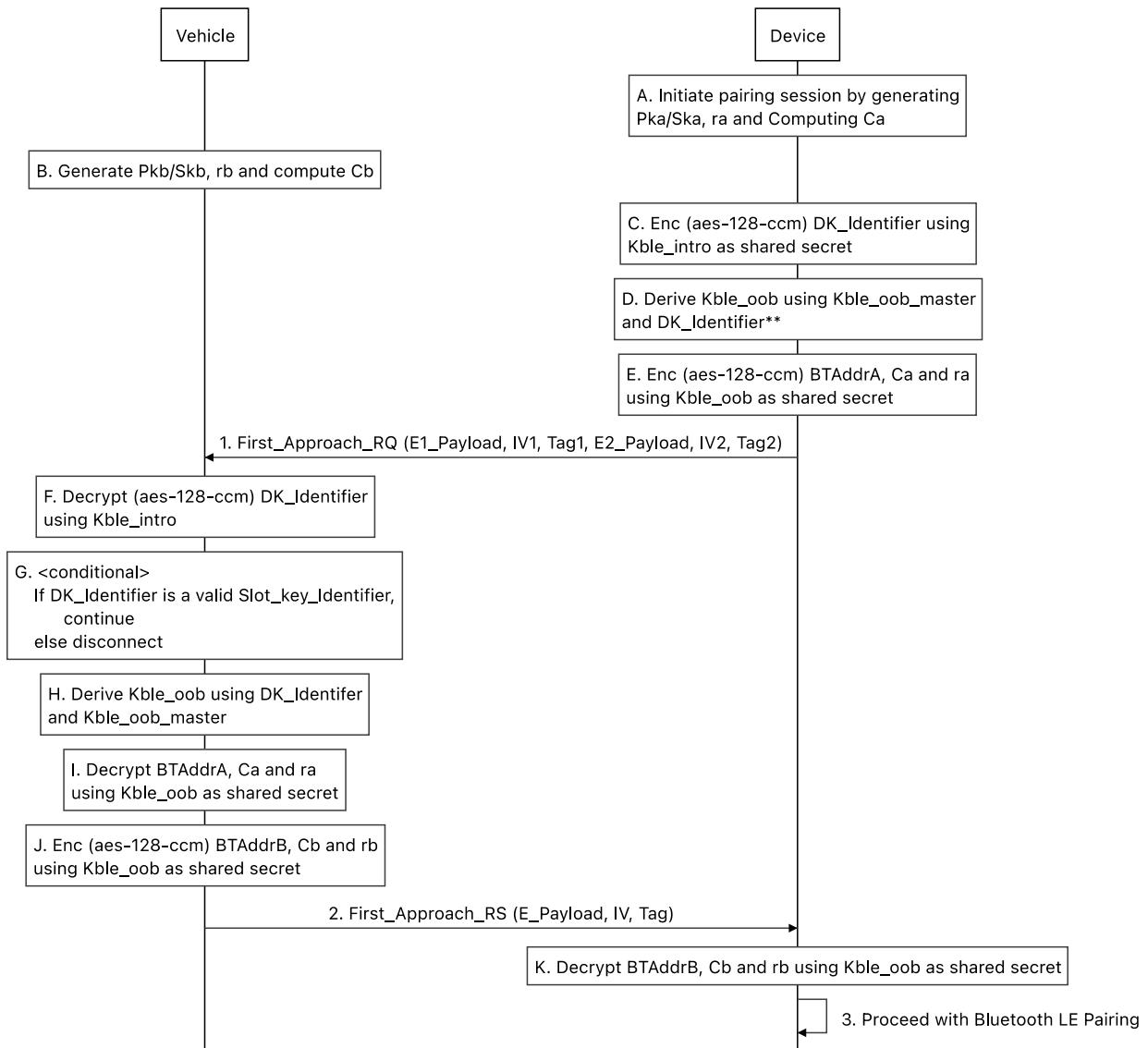
13 **Message:**

14 First_Approach_RQ (see Section [19.3.4.1](#))

15 First_Approach_RS (see Section [19.3.4.2](#))

1

Figure 19-17: Bluetooth LE Secure OOB Pairing Prep.



2

3 ** A friend device does not derive Kble_oob but receives it from the Owner device instead.

4 Elements of the Bluetooth LE Secure OOB Pairing Prep flow:

- 5 • The BTAddrA and BTAddrB are the Device Bluetooth Address and Vehicle Bluetooth Address respectively, used during pairing (See Vol 3, Part H, Section 2.2.7 of [30]).
 - 7 • For box F, vehicle shall use AES-128-CCM with Kble_intro as shared secret and 8-byte IV provided by device to decrypt DK_Identifier
 - 9 • For box G, vehicle shall react depending on Tag DA_h configuration in Table 5-14: Device Configuration Data.
- 11 If Tag DA_h in Table 5-14 is not present, or set to values 00_h or 02_h, vehicle shall validate that decrypted DK_Identifier value belongs to a valid slot.

- 1 If Tag DA_h in Table 5-14 is present and set to values 01_h, 03_h, 04_h, 05_h, vehicle shall not
2 validate that decrypted DK_Identifier value belongs to a valid slot in this step. The
3 vehicle shall validate the slot identifier during the first standard transaction after
4 Bluetooth LE pairing instead.
- 5 • For box H, vehicle shall use HKDF function with decrypted DK_Identifier and
6 Kble_oob_master to derive Kble_oob.
7 ○ Kble_oob = HKDF-SHA256(16, Kble_oob_master, DK_Identifier) //
8 (output_len, input_keying_material, info)
9 ○ Where DK_Identifier is an 8-byte padded slot_identifier. For example, if the
10 slot_identifier is 6-byte, it will have 0x0000 added as prefix to make it 8-byte.
- 11 • For box I, vehicle shall use AES-128-CCM with Kble_oob as shared secret and 8byte IV
12 to decrypt device OOB data required for Bluetooth LE Secure OOB pairing (BTAddrA ||
13 Ca || ra)
- 14 • For box J, vehicle shall use AES-128-CCM with Kble_oob as shared secret and 8-byte IV
15 to encrypt vehicle OOB data required for Bluetooth LE Secure OOB pairing (BTAddrB ||
16 Cb || rb)
- 17 • For box K, device shall use AES-128-CCM with Kble_intro as shared secret and 8-byte
18 IV provided by device to decrypt DK_Identifier
- 19 • Once device and vehicle have successfully performed Bluetooth LE Secure OOB Pairing
20 Prep, device shall initiate Bluetooth LE Pairing procedure by sending pairing request as
21 described in Section [19.2.1.4](#). Post successful pairing, device shall setup Bluetooth LE
22 encryption before moving to next step.
- 23 • The Bluetooth LE Pairing data shall be persisted for device and vehicle at the same point
24 in time the endpoint is persisted on either side. Any failure before that point shall lead to
25 rolling back the Bluetooth LE pairing data to be able to perform a retry.
- 26 • Any Bluetooth LE messages being exchanged over-the-air as part of this flow shall not
27 make use of DK message defined in Section [19.3](#).

28 **URSK Derivation Flow**

29 Once the Bluetooth LE encryption has been setup, if the vehicle supports UWB, the vehicle shall
30 execute the standard transaction flow for deriving URSK. (The URSK Derivation flow is shown
31 in Figure 19-21)

32 To derive URSK over Bluetooth LE, each APDU command and response shall be encoded as
33 shown below.

34 **Message Type:** SE message

35 **Message:**

36 APDU Command Encapsulation: DK_APDU_RQ (see Section [19.3.2.1](#))

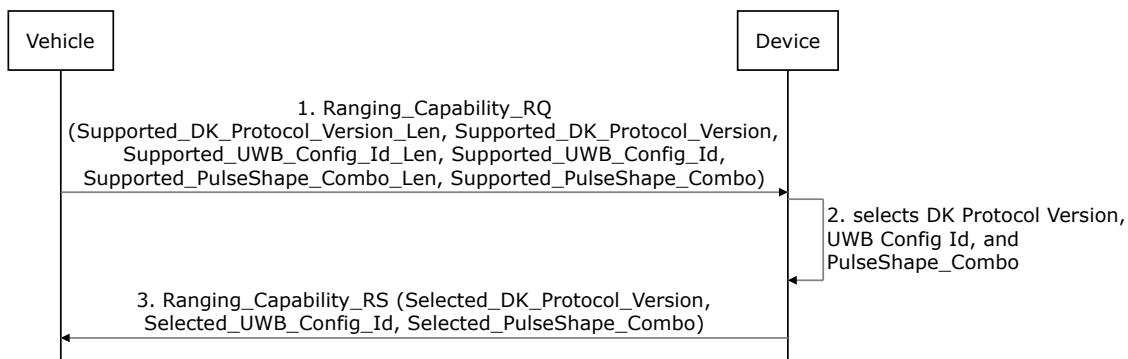
37 APDU Response Encapsulation: DK_APDU_RS (see Section [19.3.2.2](#))

38 **Capability Exchange**

39 Capability Exchange for owner pairing shall be performed if either the vehicle or device has an
40 updated DK Protocol Version, UWB Configuration Identifier, PulseShape Combinations, or a
41 combination of these parameters, which is different than the one the vehicle and device

1 supported during the last Capability Exchange. The Capability Exchange is shown in Figure
2 19-18.
3 The vehicle initiates the Capability Exchange by sending Ranging_Capability_RQ (see Section
4 [19.3.1.1](#)) to the device. Upon receiving the Ranging_Capability_RQ, the device shall respond by
5 sending Ranging_Capability_RS (see Section [19.3.1.2](#)).

6 *Figure 19-18: Capability Exchange.*



7
8 The device triggers the Capability Exchange by sending DK Event Notification with
9 Subevent_Category set to 0x01_h and the corresponding command status code set to 0x02_h. Upon
10 receiving the DK Event Notification, the vehicle shall initiate the Capability Exchange as shown
11 in Figure 19-18.

12 Time Sync

13 If the Device UWB Clock is “Not in sync” while Bluetooth LE connected with the device, the
14 vehicle shall trigger the Bluetooth LE Timesync procedure 1 if all conditions are met (see
15 Section 19.4.2).

16 Secure Ranging Setup Flow

17 Once the Capability Exchange is done, vehicle shall initiate secure ranging setup flow. In this
18 flow, the vehicle and device go through a few exchanges to finalize the ranging parameter values
19 required to set up the secure ranging. As part of this flow, the vehicle shall notify the device on
20 which URSK to use by sending Ranging_Session_RQ with its UWB_Session_Id.

21 To set up secure ranging over Bluetooth LE, each message shall be encoded as below:

22 **Message Type:** UWB Ranging Service message

23 **Message:**

- 24 Ranging Service Message: RS_RQ (see Section [19.3.1.3](#))
- 25 Ranging Service Message: RS_RS (see Section [19.3.1.4](#))
- 26 Ranging Service Message: RSS_RQ (see Section [19.3.1.5](#))
- 27 Ranging Service Message: RSS_RS (see Section [19.3.1.6](#))
- 28 DK Event Notification (see Section [19.3.8](#))

29 Figure 19-23 illustrates the Secure Ranging Setup flow.

1 The vehicle shall perform secure ranging and detect that the device is inside the vehicle before
2 proceeding to Owner Pairing Phase 3. If the device is not detected inside of the vehicle or other
3 ranging failure occurred, the vehicle shall send an appropriate SubEvent code defined in Table
4 19-72 or Table 19-74 and abort the owner pairing attempt.

5 **Owner Pairing Phase 3**

6 To perform Owner Pairing Phase 3 exchange (see Section [6.3.4](#)) over Bluetooth LE, each APDU
7 command and response shall be encoded as below.

8 **Message Type:** SE message

9 **Message:**

10 APDU Command Encapsulation: DK_APDU_RQ (see Section [19.3.2.1](#))

11 APDU Response Encapsulation: DK_APDU_RS (see Section [19.3.2.2](#))

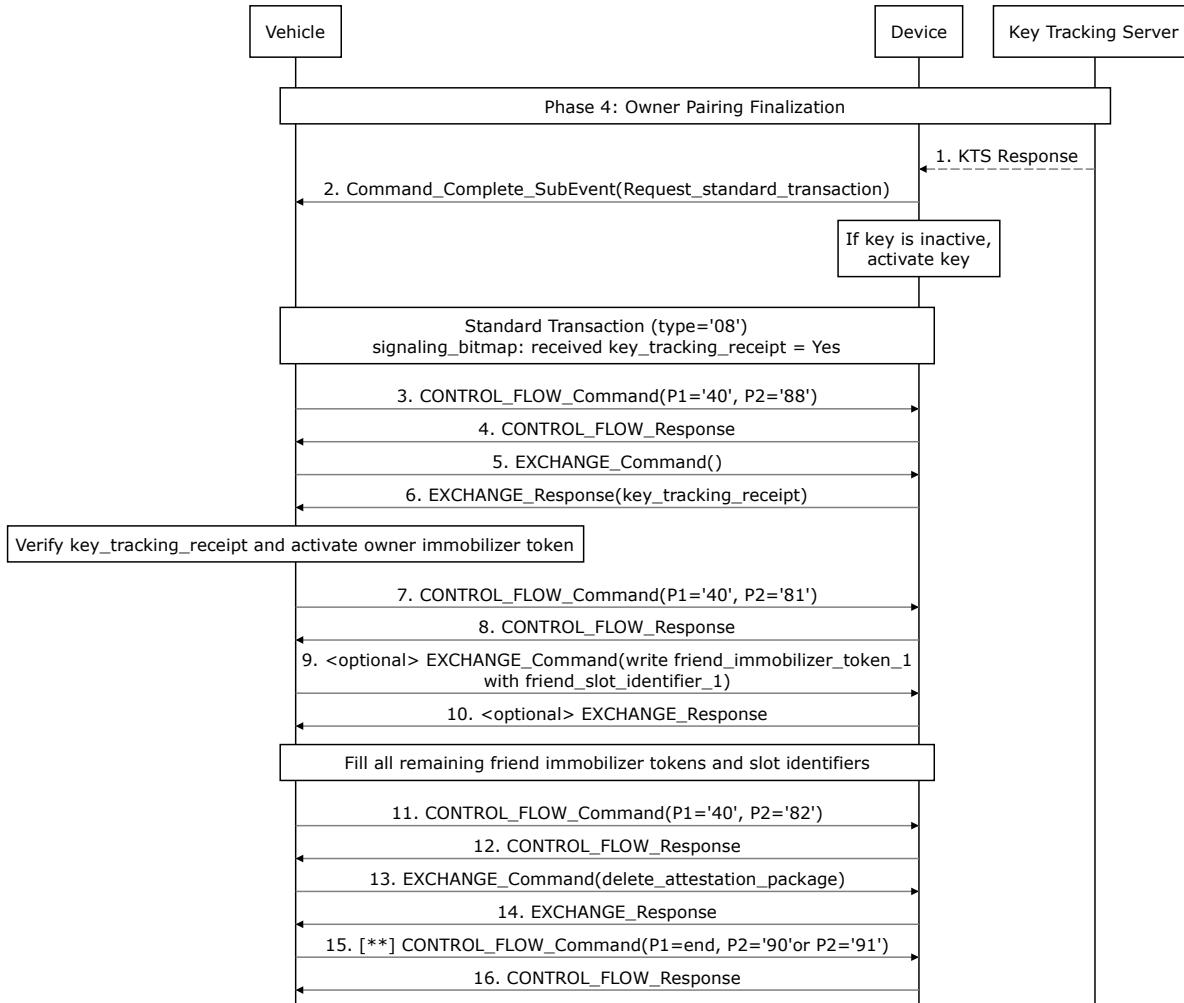
12 **Owner Pairing Phase 4**

13 Owner Pairing Phase 4 exchange (see Section [6.3.5](#)) over Bluetooth LE shall be modified as
14 shown in Figure 19-19. If KTS is supported, the device shall wait for KTS the response before
15 sending the Request_standard_transaction subevent to the vehicle to trigger standard transaction.

16

1

Figure 19-19: Owner Pairing Phase 4 over Bluetooth LE



2

3 Each APDU command and response shall be encoded as below

4 **Message Type:** SE message

5 **Message:**

6 APDU Command Encapsulation: DK_APDU_RQ (see Section [19.3.2.1](#))

7 APDU Response Encapsulation: DK_APDU_RS (see Section [19.3.2.2](#))

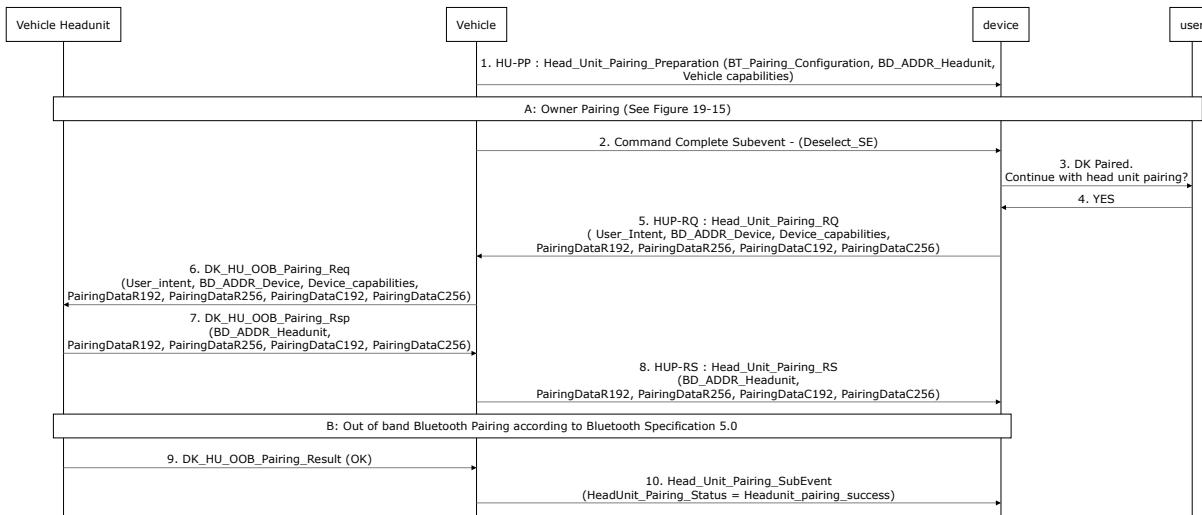
8 19.5.2 Head Unit Pairing Flow

9 This section describes the Head Unit Pairing flow once the DK pairing is completed. This
10 feature, referred as “Head Unit Pairing”, is optional for both the device and vehicle. This feature
11 only applies if the device’s Bluetooth supports BR/EDR (classic), and the vehicle Bluetooth
12 module for DK is different from that of the head unit (i.e. in a different subsystem or a physical
13 implementation). The Head Unit Pairing flow is shown in [Figure 19-20](#).

14 The Head Unit Pairing Preparation (HU-PP) message is used for head unit capability discovery
15 and initiating the Head Unit Pairing.

The vehicle may send HU-PP either before or after owner pairing is completed. Sending the HU-PP prior to owner pairing completion allows the device to display a single message at the Subevent Notification about DK pairing complete and head unit capability. If the Head Unit Pairing is supported by the vehicle, the vehicle shall send the HU-PP within 2 seconds from the Command Complete Subevent (Deselect_SE) at the end of owner pairing Phase 4 i.e., step 5 in [Figure 19-16](#). If the Head Unit Pairing is not supported or for all subsequent head unit (re)pairing the device and vehicle shall follow the Bluetooth pairing and encryption setup as defined in [\[30\]](#).

Figure 19-20: Head Unit Pairing Flow Diagram.



- Note that the order of HU-PP message and owner pairing subsection (refer to Box A in Figure 19-20) is an example and can be interchanged.
- If a device supports Head Unit Pairing, the device shall send HUP-RQ in step 5 (see Section [19.3.7.2](#)). The vehicle shall respond with HUP-RS in step 8 (see Section [19.3.7.3](#)) to complete the Head Unit Pairing. If the device does not support Head Unit Pairing, the flow in Figure 19-20 completes after step 2 and subsequent steps are not required.
- If Headunit pairing (step 10) is not finished within 60 seconds from the moment the message HUP-REQ has been sent, the device may retry the head unit pairing from step 3.
- Note that DK_HU_OOB_Pairing_Req and DK_HU_OOB_Pairing_Result messages in step 6 and 7 are implementation-specific by the vehicle OEM and are only provided as an example.
- For Friend First Approach (Section [19.5.8.2](#)), the message HU-PP shall be sent after the “secure ranging setup flow” is completed.
- The flow diagram in Figure 19-20 shows the case where OOB communication is possible from both directions. The vehicle shall configure the parameter BT_Pairing_Configuration in message 1 to indicate to device if the vehicle head unit supports both directions or a single direction (vehicle head unit to device) for the OOB communication.
- If OOB communication is possible only in the direction from vehicle head unit to device, the parameters PairingDataR192, PairingDataR256, PairingDataC192, and PairingDataC256 in step 6 should be ignored by the vehicle head unit (see volume 2 Part H Section 7.2.2 of [\[30\]](#)).

1 **19.5.3 Passive Entry**

2 After owner pairing has been successfully performed, upon each approach, at minimum secure
3 ranging flow shall be exercised to establish secure ranging before enabling any of the passive
4 entry features/actions such as ‘Welcome Lights’, ‘Unlock’, ‘Lock’, etc. Once the secure ranging
5 has been established and device has been localized, vehicle may decide to initiate any of the
6 above actions, as per its policy or requirements.

7 For passive entry, one of the following conditions shall trigger the vehicle (except if the vehicle
8 is in low power mode, or vehicle is not ready, for example, due to internal conflict) to establish
9 secure ranging with the device:

- 10 1. Device sends Device Ranging Intent SubEvent with at least low_approach_confidence.
11 2. The vehicle determines that there is user intent to use vehicle features that require secure
12 ranging with the device, e.g., user touches door handle

13 The URSK is needed before secure ranging can be established. The vehicle may have a pre-
14 derived URSK or derive a new URSK as needed. The URSK confidentiality and integrity shall
15 be preserved during the whole lifetime of the URSK.

16 **19.5.4 URSK Management**

17 Vehicle may derive and store a new URSK by:

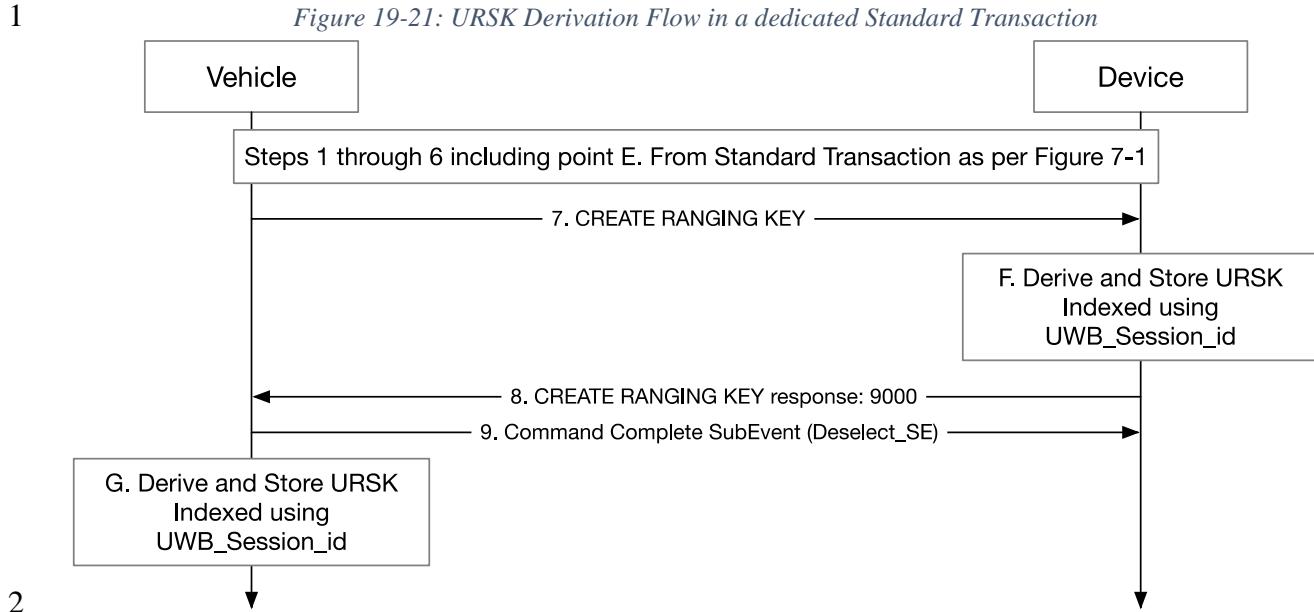
- 18 • Executing a dedicated transaction flow (i. e. with a AUTH0 command indicating a
19 transaction_code value 10_h) as shown in [Figure 19-21](#).
20 OR
- 21 • By adding a CREATE RANGING KEY command to a Standard Transaction flow
22 intended for another purpose like start engine (i.e., with a AUTH0 command indicating a
23 transaction_code value not equal to 10_h) as shown in [Figure 19-30](#); see section [19.5.6.1](#).
24 Each time this flow is executed successfully, a unique URSK is derived and stored
25 indexed using a UWB_Session_Id. This UWB_Session_Id is the least significant 4 bytes
26 of the transaction_identifier, a 16 bytes random number generated by the vehicle for each
27 AUTH0 and sent to the device as part of AUTH0 command along with other parameters.
28 For more details, see Section 15.3.2.9.

29 To derive URSK over Bluetooth LE, each APDU command and response shall be encoded as
30 below.

31 **Message Type:** SE message

32 **Message:**

- 33 APDU Command Encapsulation: DK_APDU_RQ (see Section [19.3.2.1](#))
34 APDU Response Encapsulation: DK_APDU_RS (see Section [19.3.2.2](#))



- 3 To find details on vehicle's processing for box A, E and H, see Section [7](#) and Section [15.3.2.26](#).
4 For box G and H, once the URSK has been successfully derived, it shall be stored in secure
5 storage indexed using UWB_Session_Id.
6 Vehicles capable of secure ranging shall support the derivation and storage of at least one pre-
7 derived URSK per Digital Key. Devices capable of secure ranging shall support the derivation
8 and storage of two pre-derived URSKs per Digital Key. Each of these pre-derived URSKs is
9 indexed using its associated UWB_Session_Id.
10 The vehicle shall use the Secure Ranging Setup Flow (see [Figure 19-23](#)) to activate a pre-derived
11 URSK. There shall be at most one active URSK per Digital Key, and this active URSK is
12 discarded anytime another one is activated.
13 After a pre-derived URSK is activated, the vehicle shall request derivation and storage of another
14 pre-derived URSK. The vehicle shall control when this additional URSK derivation occurs to
15 minimize user impact (for example, after passive entry completed) but is recommended to occur
16 shortly after URSK activation. [Table 19-85](#) shows the URSK storage requirements.

17 *Table 19-85: URSK storage requirements per Digital Key endpoint.*

URSK Types	Definition	Vehicle URSK Storage	Device URSK Storage
Pre-derived	URSK that has not been activated using the Secure Ranging Setup Flow and has only been kept in secure storage. TTL is not defined until key is activated.	At least 1	2
Active	URSK activated using the Secure Ranging Setup Flow and has neither exhausted its STS_Index max incrementation nor has an expired TTL.	1	1

- 18 The URSK shall be discarded if one of the following conditions is met:
19 1. The STS_Index reaches its maximum value of $2^{31}-1$.
20 2. The STS_Index was lost, and it cannot be ensured that a previously used STS_Index
21 won't be used again

- 1 3. The URSK TTL (Time-To-Live) has expired. The vehicle shall enforce an URSK TTL
2 lower or equal to 12 hours (vehicle OEM specific). The URSK TTL starts when the first
3 dURSK is derived. The first dURSK shall be derived immediately before sending or after
4 receiving Ranging Session Setup Response for the device or vehicle, respectively.
- 5 4. A new URSK is activated through Secure Ranging Setup Flow.

6 *19.5.5 Flow Selection - Establish Secure Ranging*

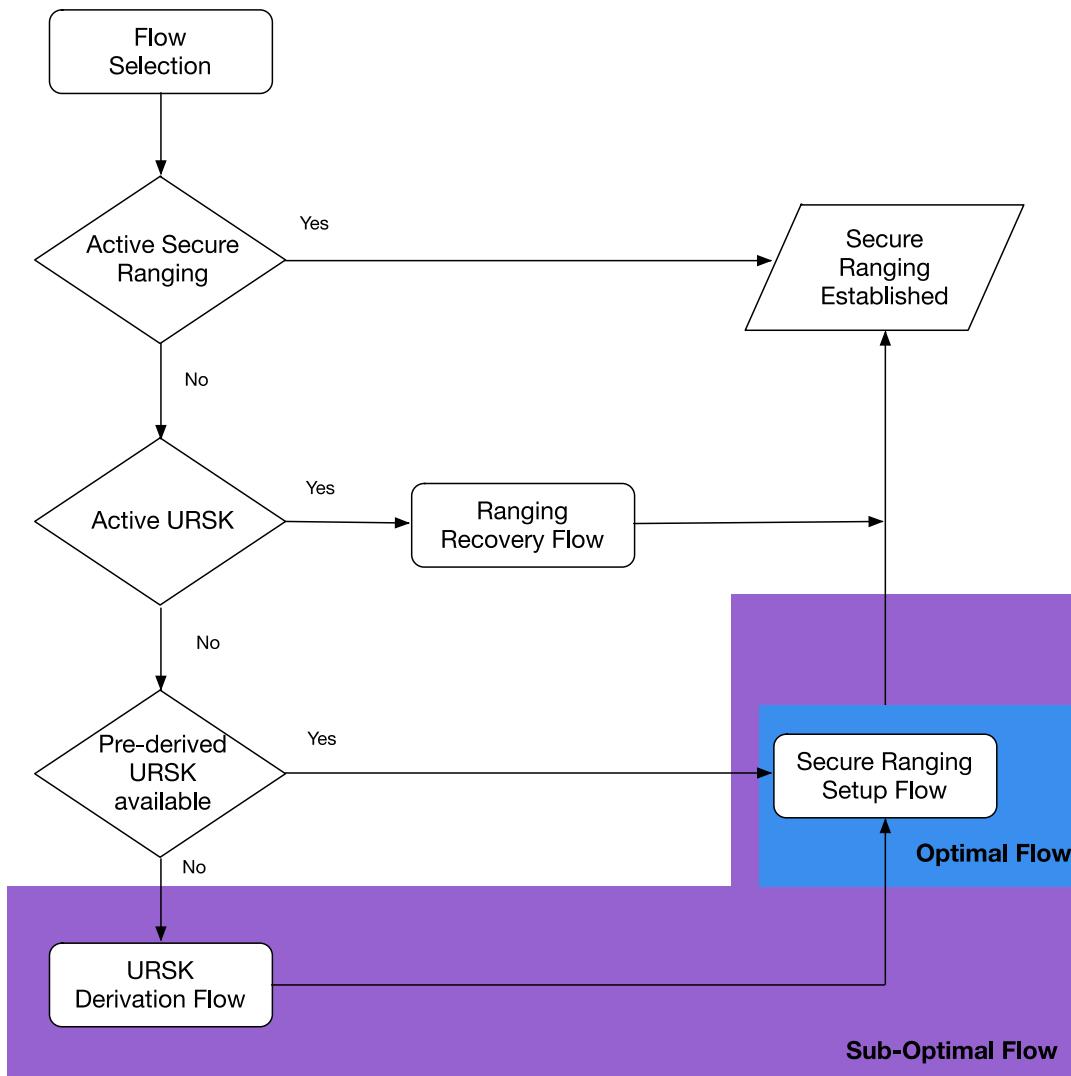
7 The vehicle has three options to establish secure ranging:

- 8 1. Optimal Flow
- 9 2. Sub-Optimal Flow
- 10 3. Ranging Recovery Flow

11 To decide which flow needs to be exercised for establishing secure ranging, vehicle shall go
12 through below flow selection. In the following, a ranging session is considered as an Active
13 Ranging Session when the vehicle and the device are exchanging UWB secure ranging messages
14 using the same URSK. This flow is shown in Figure 19-22.

1

Figure 19-22: Flow Selection for Establishing Secure Ranging



2

- 3 If the vehicle needs to localize the device, the vehicle shall first check which flow should be initiated for establishing secure ranging. For this, the vehicle shall first check if it already has an active ranging session. If it does, vehicle shall use that session to localize the device.
- 4 If there is no active ranging session, the vehicle shall then check whether an active URSK exists.
- 5 If it does, the vehicle shall follow the ranging recovery flow to recover the ranging session (using its UWB_Session_Id) associated with that active URSK.
- 6 If there is no active URSK, the vehicle shall check whether a pre-derived URSK exist. If it does, the vehicle shall follow optimal flow to establish secure ranging.
- 7 If there are no active or pre-derived URSKs, the vehicle shall fallback to sub-optimal flow.
- 8 If there is an active ranging session, and if the URSK TTL at the vehicle is about to expire or
- 9 vehicle decides to use a URSK with shorter TTL for engine start (see Section 19.5.6.1), the
- 10 vehicle may setup a new secure ranging session with a pre-derived URSK. When the pre-derived
- 11 URSK becomes active, the vehicle and the device shall discard the previously active URSK (if

any). For more details on ranging recovery flow, optimal flow and sub-optimal flow, refer to below subsections.

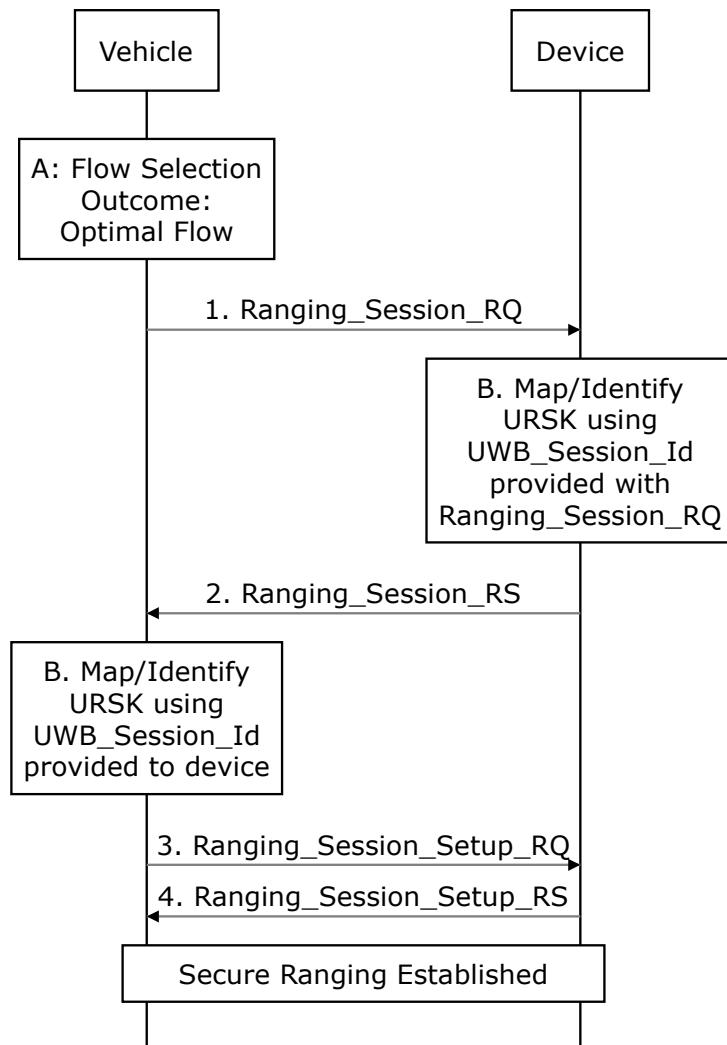
Optimal Flow

Optimal flow shall be exercised by vehicle, if it has a pre-derived URSK available. Optimal flow simply follows Secure Ranging Setup flow using a pre-derived URSK. If the vehicle uses this flow to activate a pre-derived URSK, the vehicle and device shall discard the currently active URSK (if any) for the associated Digital Key before the new URSK is activated.

To setup secure ranging, each message shall be encoded with Message Type: UWB Ranging Service message

The Secure Ranging Setup flow is shown in Figure 19-23.

Figure 19-23: Secure Ranging Setup Flow.

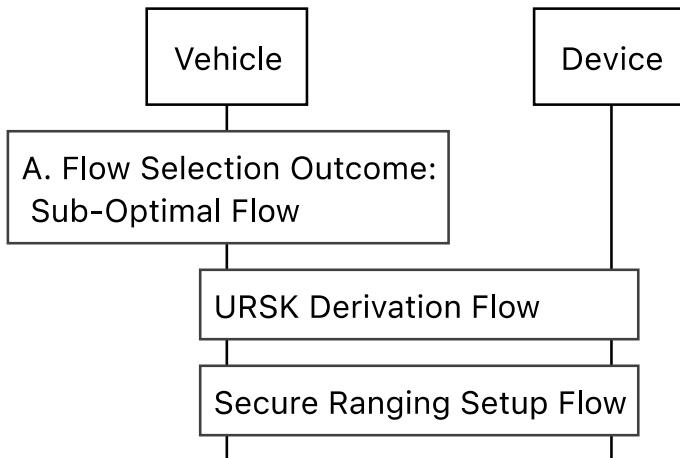


13

1 **Sub-Optimal Flow**

2 Sub-Optimal flow shall be exercised if a vehicle has no pre-derived URSK available. Sub-
3 Optimal flow is a combination of URSK Derivation flow and Secure Ranging Setup flow.
4 Figure 19-24 illustrates the use of Sub-Optimal flow, in the absence of pre-derived URSK.

5 *Figure 19-24: Sub-Optimal Flow.*



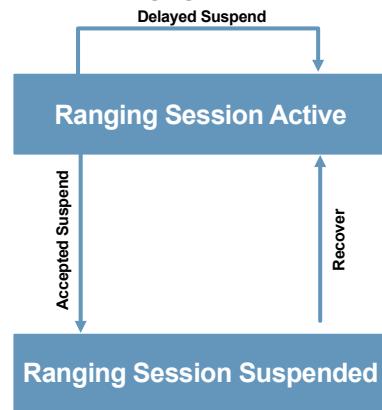
6
7
8 In addition, if the device fails to activate an URSK as part of Optimal flow and no more pre-
9 derived URSKs are available or when vehicle URSKs are out of sync with device URSKs (refer
10 Section [19.8](#)), vehicle shall fall back to Sub-Optimal flow.
11 Once secure ranging has been established by either using Optimal or Sub-Optimal flow, vehicle
12 may localize the device. Once device localized, vehicle may perform any of the passive entry
13 features/actions.

14 **Ranging Recovery Flow**

15 Once a ranging session has been established, it can be active for an extended period of time as
16 long its URSK TTL has not expired or STS_Index has not reached max limit. However, as
17 shown in Figure 19-25, either the vehicle or device may put an active ranging session in a
18 suspended state for power optimization.

1

Figure 19-25: Ranging Session State Machine.



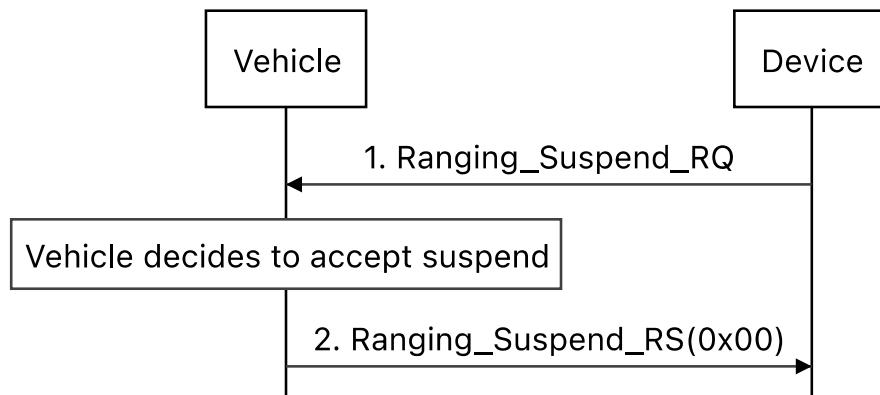
2

3 The decision on when to send a Ranging Suspend Request is up to the sender requester.
4 However, the receiver may choose to delay the suspension request for a limited period of time.
5 For example, device may request to put the ranging session to suspended state, if it does not
6 detect any motion for extended period of time. On the other hand, vehicle may respond back with
7 the request to delay the suspension, if it strongly believes ranging session is either still needed or
8 will be in immediate future.

Figure 19-26 illustrates Ranging Suspend Accepted flow when requested by the device.

10

Figure 19-26: Ranging Suspend Accepted Flow.

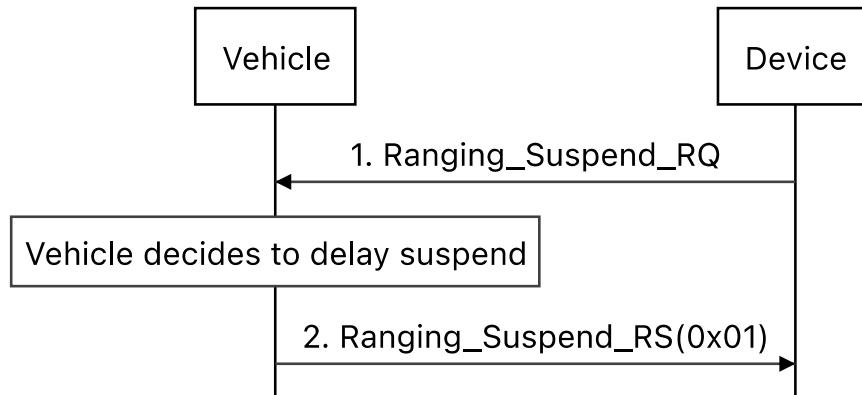


11

12 Figure 19-27 illustrates Ranging Suspend Delayed flow when requested by the device.

1

Figure 19-27: Ranging Suspend Delayed Flow.

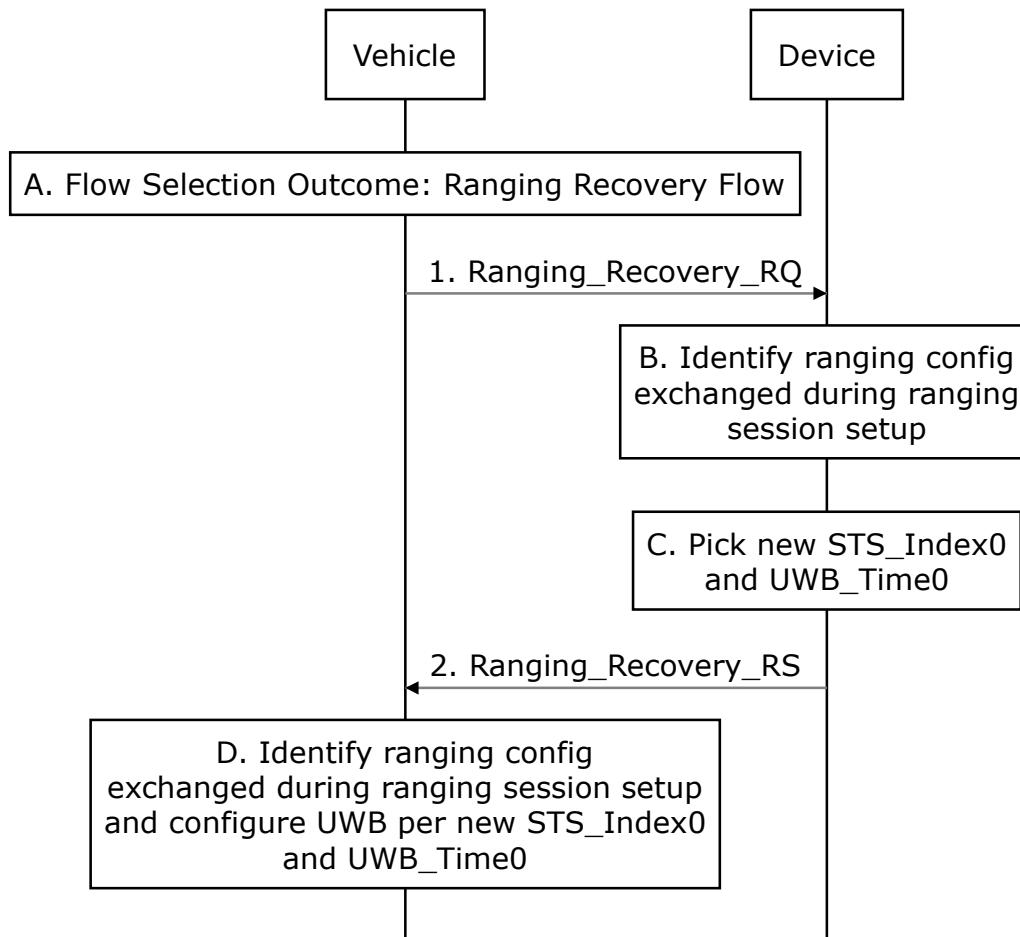


2

- 3 When in suspended mode, vehicle may send a Ranging Recovery Request message (see Section
4 [19.3.1.9](#)) to the device. The device may trigger ranging recovery by sending Device Ranging
5 Intent SubEvent to the vehicle
- 6 The Ranging Recovery flow offers a low latency based secure ranging which requires minimal
7 exchange and no new URSK.
- 8 However, before initiating Ranging Recovery flow, vehicle shall meet the following
9 requirements:
- 10 1. Has an active URSK (TTL has not expired) associated with the suspended ranging session in
11 which the vehicle intends to recover
- 12 2. No change to ranging configurations of the suspended ranging session is needed, such as
13 change of frequency, # of anchors, etc.
- 14 If vehicle needs a secure ranging, the vehicle shall check whether there is any suspended ranging
15 session for the connected device that meets both of the above requirements. If it does, the vehicle
16 shall initiate Ranging Recovery flow. When device receives the Ranging Recovery Request, it
17 shall identify the same set of configurations used to establish the ranging session for the provided
18 UWB_Session_Id. Device shall then pick and send a new UWB_Time0 and STS_Index0.
- 19 Figure 19-28 illustrates Ranging Recovery flow.

1

Figure 19-28: Ranging Recovery Flow.



2

3 For box A, refer to Section [19.5.5](#).

4 For box D, vehicle shall select previously negotiated UWB config parameters for
5 UWB_Session_Id associated with recovering ranging session.

6 **Vehicle Low Power State**

7 When the vehicle is in low power state, it may not start ranging after establishing a Bluetooth LE
8 connection. In such case the vehicle may send a vehicle state SubEvent notification indicating its
9 “Low power mode”. After receiving this notification, the device shall be responsible to start
10 ranging using the Ranging Intent SubEvent.

11 The device may immediately send the Ranging Intent or at a later point with higher confidence,
12 when a vehicle approach has been detected.

13 **19.5.6 Standard Transaction over Bluetooth LE**

14 If vehicle requires a standard transaction (e.g. in order to request an immobilizer token), the
15 standard transaction protocol as shown in Figure 7-1, followed by a Command Complete
16 SubEvent with Command_Status Deselect_SE, shall be performed.

1 Each APDU command and response shall be encoded as below.

2 **Message Type:** SE message

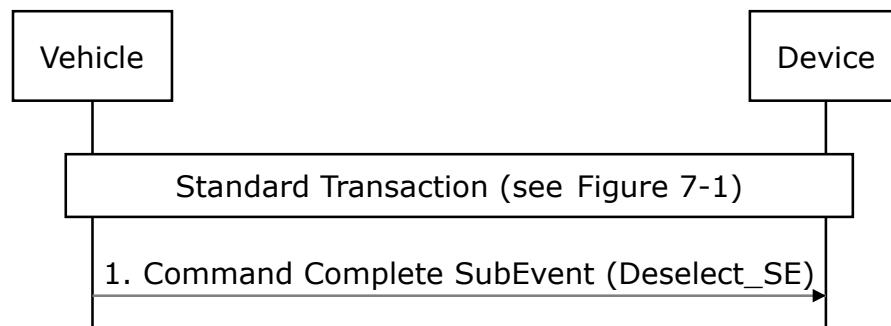
3 **Message:**

4 APDU Command Encapsulation: DK_APDU_RQ (see Section [19.3.2.1](#))

5 APDU Response Encapsulation: DK_APDU_RS (see Section [19.3.2.2](#))

6 Figure 19-29 illustrates the standard transaction flow.

7 *Figure 19-29: Standard transaction over Bluetooth LE.*



8

9 *19.5.6.1 Standard Transaction with URSK derivation via Bluetooth LE*

10 Optionally, a URSK can also be derived within a standard transaction via Bluetooth LE that was
11 mainly intended for other purposes like start engine, i. e., regardless of the AUTH0 Command
12 Transaction Type Coding. The necessary steps for the URSK derivation are performed after an
13 otherwise successful standard transaction with (optional) EXCHANGE commands. This is
14 depicted in [Figure 19-30](#).

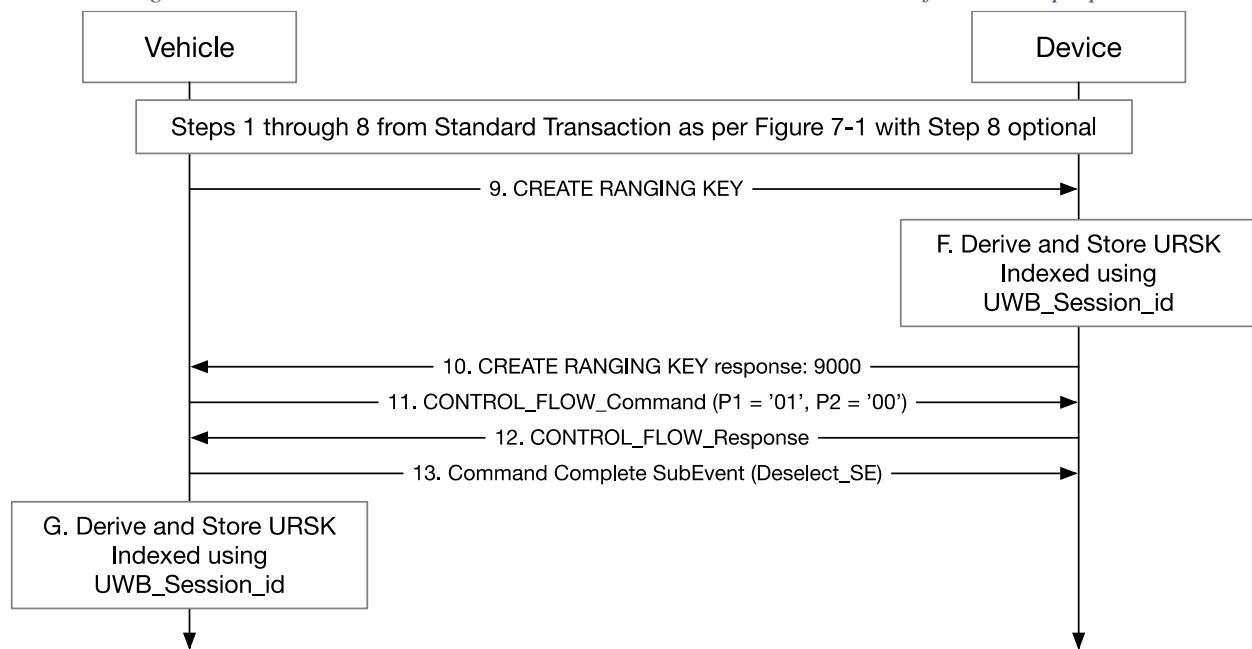
15 *Note:* Steps 1 through 8 in [Figure 19-30](#) are identical to Steps 1 through 8 from Figure 7-1.

16

17

1

Figure 19-30: URSK Derivation Flow in a Standard Transaction intended for another purpose



- 2
- 3 If an error occurs within steps 1 to 8 of the standard transaction, URSK derivation is skipped and the appropriate error is signaled to the device. Vehicle then sends Command complete SubEvent (Deselect SE) to Device as per step 11 from Figure 19-30.
- 4
- 5
- 6 In case an error occurs during URSK derivation, only the action triggered with the preceding standard transaction is (already) accepted by the vehicle. If necessary, step 11: Command
- 7 Complete SubEvent (Deselect SE) shall be sent to the device.
- 8
- 9 If a URSK derivation is still needed, a dedicated standard transaction for URSK derivation as
- 10 described in section 19.5.4 should be performed.

11 19.5.7 Engine Start

- 12 Engine start shall be authorized for a limited time, only after completion of a standard transaction (which may also be used to retrieve an Immobilizer Token) and a UWB ranging.
- 13
- 14 The vehicle OEM may choose to setup a new ranging session using a different URSK with short
- 15 TTL for engine start authorization. The usage of a short TTL (e.g. 30s) for the UWB ranging
- 16 used for engine start increases overall security by reducing the general exposure of the URSK
- 17 used for enabling engine start.

18 19.5.8 Friend - Sharing & First Approach

19 19.5.8.1 Friend Sharing

- 20 Digital Key solution supports sharing feature which allows vehicle owners to offer managed
- 21 digital access of their vehicle with other friends. For a Bluetooth LE/UWB capable vehicle,
- 22 regardless of owner device's 'Wireless Capabilities', sharing ensures that a friend device who
- 23 may have UWB capabilities can enjoy Bluetooth LE/UWB based experience even when owner's
- 24 device may not have Bluetooth LE/UWB capability.

1 To enable passive entry experience for friend devices, vehicle provides its 'Wireless Capabilities'
2 (7F49 template) and both device and vehicle derive Kble_oob_master as part of 'Owner Pairing'
3 flow as described in Section [19.5.1](#).

4 During sharing, owner's device make use of Kble_oob_master and DK_Identifier to derive
5 Kble_oob as described below.

6 Friend Bluetooth LE Pairing: Friend OOB Pairing Data Derivation

- 7 • Kble_oob = HKDF-SHA256(16, Kble_oob_master, DK_Identifier)
8 // (output_len, input_keying_material, info)

9 Where DK_Identifier is an 8-byte padded slot_identifier of the friend. For example, if the
10 slot_identifier is 6 bytes, it will have 0x0000 added as prefixed to make it 8 bytes.

11 Owner device shall send an updated 7F49 template (see Table 19-84) as part of the Import
12 Request. This template is shared by the owner device to the friend device as described in Section
13 [11.8.2.1](#).

14 *19.5.8.2 Friend First Approach*

15 For a friend with NFC/Bluetooth LE/UWB capable device which has successfully received
16 Digital Key access for owner's vehicle (Bluetooth LE/UWB capable), the friend device can
17 enjoy authorized RKE features using Bluetooth LE and passive entry feature without any further
18 manual input or intent.

19 Note that the first friend transaction over Bluetooth LE is unrelated to any vehicle action such as
20 lock or unlock. There is no requirement for user authentication for this transaction type.

21 After a successful Bluetooth LE Pairing Procedure, the vehicle shall initiate a first friend
22 transaction. If the first friend transaction was successful, the vehicle shall end the Standard
23 Transaction as defined in Figure 7-1 by a CONTROL FLOW command indicating First approach
24 successful.

25 The Bluetooth LE Pairing Data shall be persisted:

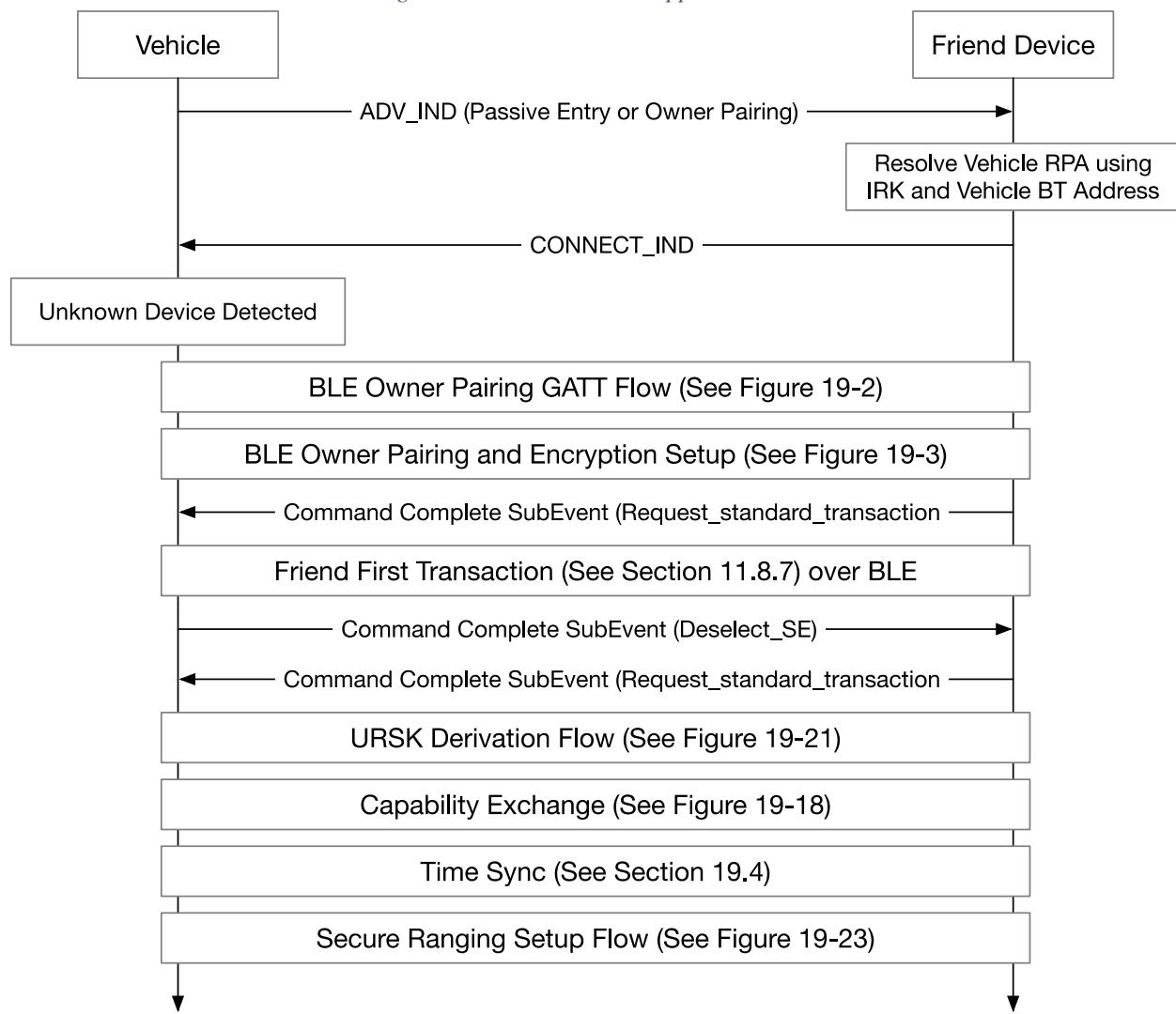
- 26 1. For the vehicle, when the CONTROL FLOW command indicating First approach
27 successful was sent,
- 28 2. For the device, when the CONTROL FLOW command indicating First approach
29 successful was received.

30 Any error prior to this point shall lead to rolling back the Bluetooth LE pairing data. The device
31 shall then retry the first approach flow on reconnecting to the vehicle.

32 Figure 19-31 illustrates Friend First Approach flow.

1

Figure 19-31: Friend First Approach Flow.



2

Bluetooth LE Link Layer Connection Establishment and L2CAP Setup Connection-Oriented Channel Setup

Upon receiving an ADV_IND from the vehicle, a friend device shall first resolve the vehicle RPA using the IRK, and then shall send a CONNECT_IND to establish Bluetooth LE link layer connection. Once the Bluetooth LE link layer connection is established, the friend device and vehicle shall establish L2CAP Connection-Oriented channel.

Bluetooth LE Pairing and Encryption Procedure

Both device and vehicle shall perform Bluetooth LE Secure OOB Pairing Prep flow, except for Kble_oob derivation (see Figure 19-17, except for box D).

Once vehicle has successfully derived Kble_oob, device shall initiate Bluetooth LE Pairing procedure by sending pairing request as described above in Section 19.2.1.4. Post successful pairing, device shall setup Bluetooth LE encryption before moving to Friend First Transaction and URSK Derivation flow as described in Sections 11.8.7 and 19.5.4, respectively.

- 1 Any Bluetooth LE messages being exchanged over-the-air as part of this flow shall not make use
2 of DK message defined in Section [19.3](#).

3 **Capability Exchange**

- 4 For Friend First Approach, the Capability Exchange shall be performed as described in Figure
5 19-18.

6 **Time Sync**

- 7 If the Device UWB Clock is “Not in sync” while Bluetooth LE connected with the device, the
8 vehicle shall trigger the Bluetooth LE Timesync procedure 1 if all conditions are met (see
9 Section 19.4.2)

10 **19.5.9 RKE Function Flow**

11 Remote transactions allow device to initiate on-demand features (e.g. lock, unlock and alarm)
12 from within Bluetooth LE range. The protocol supports single event-triggered actions (e.g.,
13 locking on single tap) and enduring actions (e.g., holding a slider to adjust a window).

14 RKE Functions shall be supported by vehicles and devices with Wireless Capability
15 Combinations WCC2 or WCC3 (see Table 19-78 for list of functions)

16 Before triggering a desired action, the device shall perform user authentication (see Section [9](#)). If
17 the user is successfully authenticated, the device shall request a challenge from the vehicle by
18 sending RKE Request SubEvent. The device shall use the RKE_Challenge received in
19 RKE_AUTH_RQ as arbitrary data to create a device signature using its private key for the
20 requesting vehicle.

21 The device shall provide the user with the option to enable implicit UA, to enable explicit UA, or
22 to disable use of RKE according to one of the allowed configurations defined in Table 19-86.
23 RKE policies may be selected individually for each Digital Key.

24 *Table 19-86: RKE allowed user authentication configuration.*

Configuration	Improved UX Policies	High Security Policies	
	Enable Implicit UA	Enable Explicit UA	Disable RKE
1	Supported	Supported	Supported
2	Supported	Supported	Not Supported
3	Supported	Not Supported	Supported
4	Not Supported	Supported	Supported
5	Not Supported	Supported	Not Supported

- 25 The implicit UA policy is described in Section [9.2](#).
26 The explicit UA policy shall require an additional UA to perform an RKE function (see Section
27 [9.1](#)). This additional UA shall be performed after the intent to perform the RKE function has
28 been registered and before the device signature for the RKE_AUTH_RS is generated for a
29 specific key in the SE.
30 If usage of RKE is enabled, explicit UA should be selected by default.

1 To exchange RKE_Challenge and challenge response over BLE, it shall be encoded as below.

2 **Message Type:** Supplementary Service message

3 **Message Type:**

4 **Message:** RKE_Auth_RQ (see Section [19.3.5.1](#))

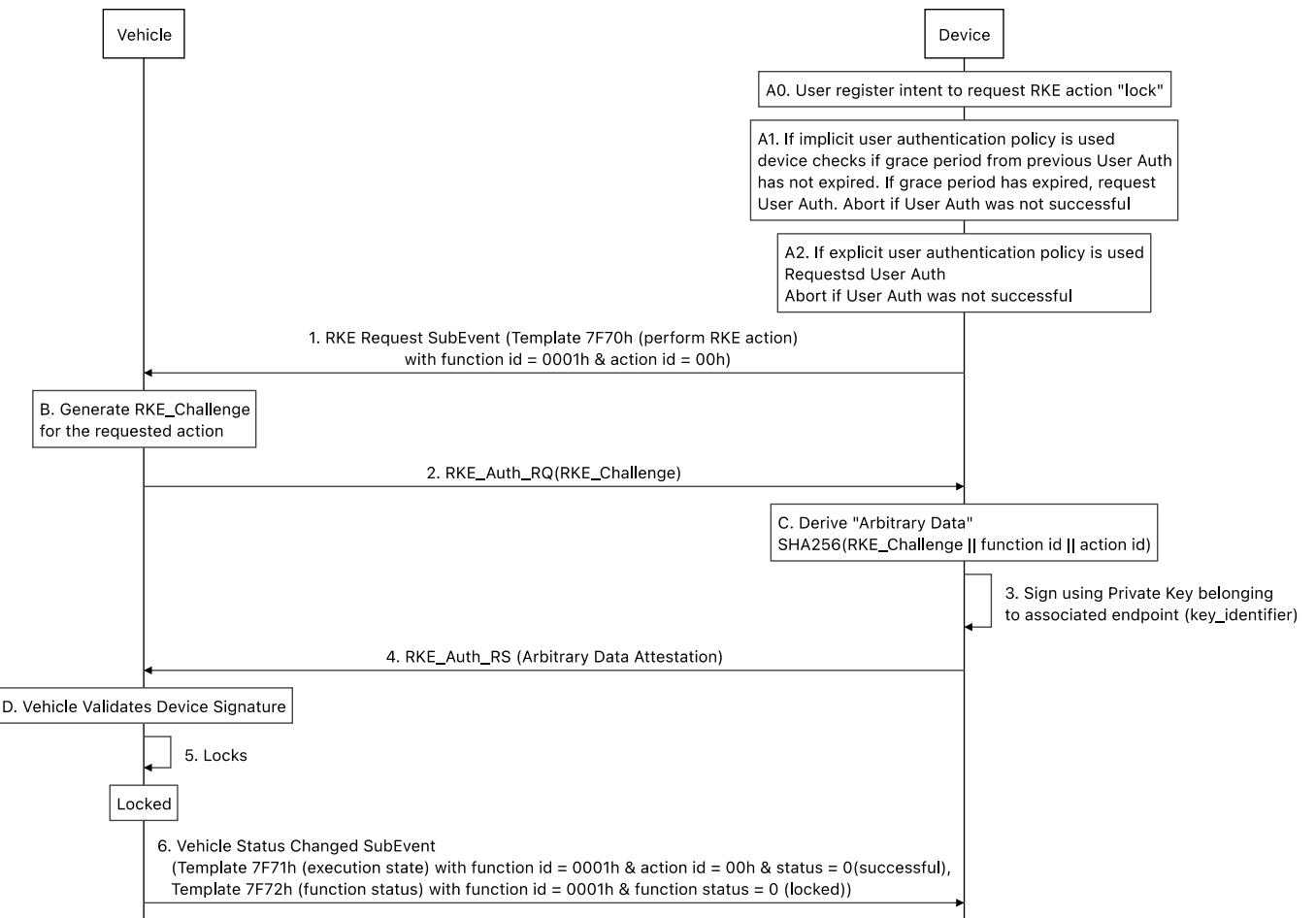
5 **Message:** RKE_Auth_RS (see Section [19.3.5.2](#))

6 DK Event Notification (see Section [19.3.8](#))

7 *19.5.9.1 Event-based RKE Action*

8 Figure 19-32 provides an example flow for the event-based RKE flow with function status
9 exchange.

10 *Figure 19-32: RKE Flow for an Event-based RKE Action.*



11

12 In box A, the user triggers the action and performs user authentication (see Section [9](#)).

13 In box C, the device shall derive arbitrary data by performing SHA256 of the concatenation of
14 the RKE_Challenge and the corresponding requested RKE Request SubEvent function id and
15 action id (see Section [19.3.8.2](#)).

- 1 In box D, vehicle shall derive arbitrary data using the same procedure as the device and shall
- 2 perform signature verification (see Section [15](#)), before the action is executed.
- 3 In step 6, the successful execution together with a Vehicle function status change is
- 4 communicated via the Vehicle Status Changed SubEvent.

5 *19.5.9.2 Enduring RKE Action*

6 Enduring RKE allows an action to be performed as long as the user confirms that the action
7 should be continued (e.g. by holding a slider to close a window). The vehicle may stop the
8 function when an end state was reached or an error occurred.

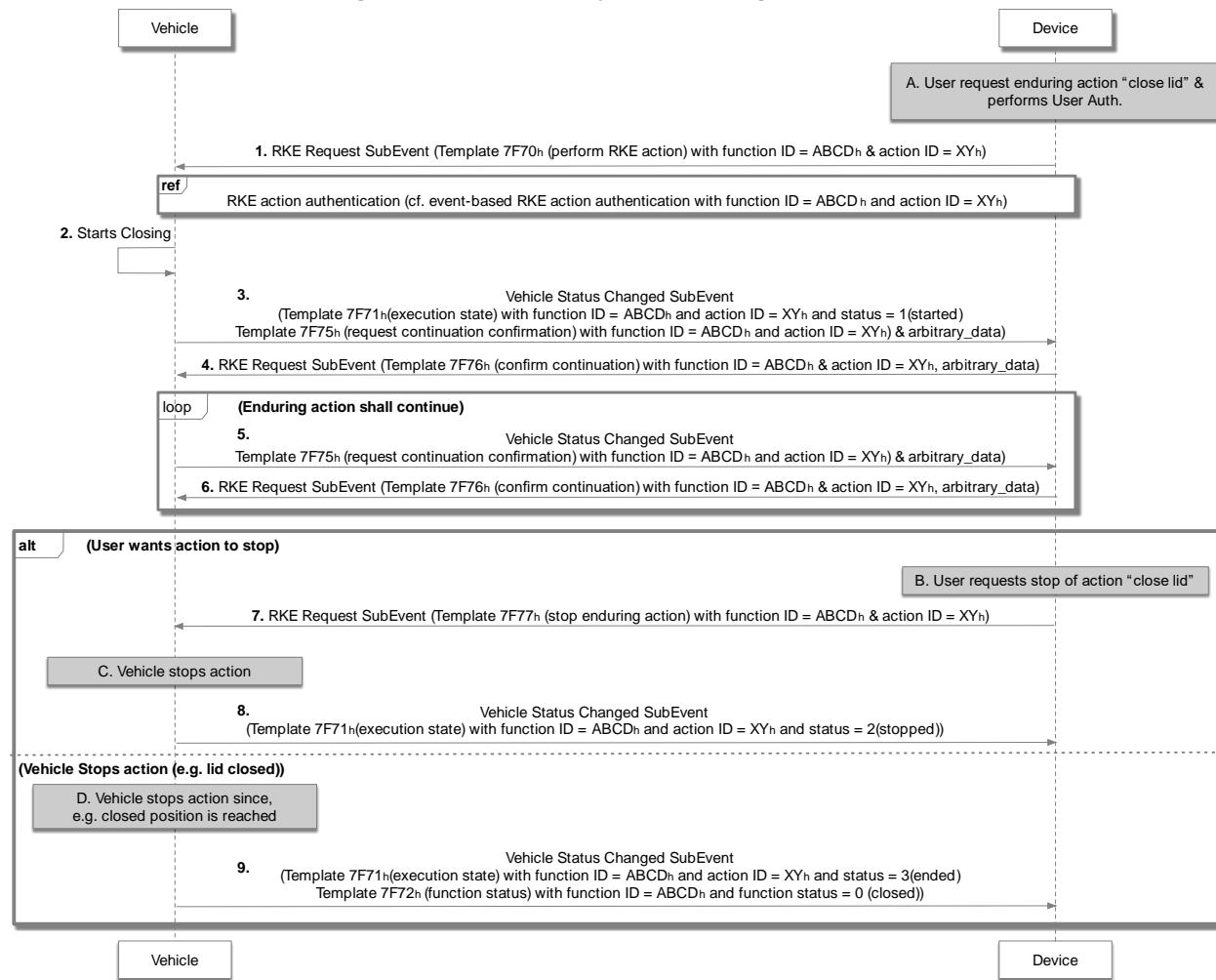
9 Enduring RKE offers two modes:

- 10 1. With Continuous Confirmation: (see loop with “Enduring action shall continue”) is required.
In this case, the continuation confirmation shall only be sent if the user confirms the action
on the user interface.
- 11 2. Without Continuous Confirmation: a certain action takes place until the device stops the
execution (loop with continuation request is not present).

12 Figure 19-33 provides an example of an Enduring RKE action with Function id ABCD and
13 action id XY are used for illustration only.

1

Figure 19-33: RKE Flow for an Enduring RKE action.



2

3 In box A, the user starts the Enduring RKE action.

4 After performing the RKE authentication as defined in Section 19.5.9.1, the Enduring RKE flow starts. The confirmation interval is controlled by the vehicle. The arbitrary data as payload in the Vehicle Status Changed SubEvent with "Request Continuation Confirmation" and, in the RKE Request SubEvent with "Confirm Continuation" can be the basis of a vehicle OEM-specific mechanism to ensure that the app on the device is still reachable or to perform round-trip time measurements. If a "Request Continuation Confirmation" Vehicle Status Changed SubEvent contains arbitrary data, the device shall return it unmodified in the corresponding "Confirm Continuation" RKE Request SubEvent. This should happen within the next connection event (30ms recommended). It is up to the Device OEM implementation to call the callback before or after sending the "Confirm Continuation" RKE Request SubEvent.

14 While the user is actively using an Enduring RKE function with continuous confirmation on the device for the given vehicle, and if the vehicle does not receive a continuation confirmation within 100ms after a continuation confirmation request was sent, it may stop the function. This time measurement is between the over-the-air packet sent by vehicle and the over-the-air packet received by the vehicle.

1
2 In box B, the user wants to stop the Enduring RKE action. This is indicated by the device via the
3 “stop enduring action” payload.

4 In box D, the vehicle initiates the stop of the Enduring RKE action. The device can be notified
5 when an enduring function reaches a certain stop state or when an error occurs.

6 The Vehicle Status Changed SubEvent may contain one or more function status updates along
7 with the Continuation Confirmation Request.

8 *19.5.9.3 Vehicle Function Status Update*

9 Figure 19-34 shows the possible flows for requesting a vehicle function’s status and being
10 informed about status changes. The subscription mechanism allows the device to subscribe to
11 vehicle OEM-specific function ids, the vehicle shall send function status updates when the status
12 of those function ids changes. With each subscription message, the vehicle shall delete the
13 current subscription ranges and take over the new ranges. To unsubscribe from updates, a single
14 function range with “from function id” = 0 and “to function id” = 0 shall be sent.

15 Besides subscription, the status of all supported functions or the status of specific functions can
16 be requested. When all supported functions are requested by the device, the vehicle shall only
17 return the function ids with status that are supported by the vehicle. The function status report
18 can be split in multiple messages depending on the maximum supported L2CAP SDU size.

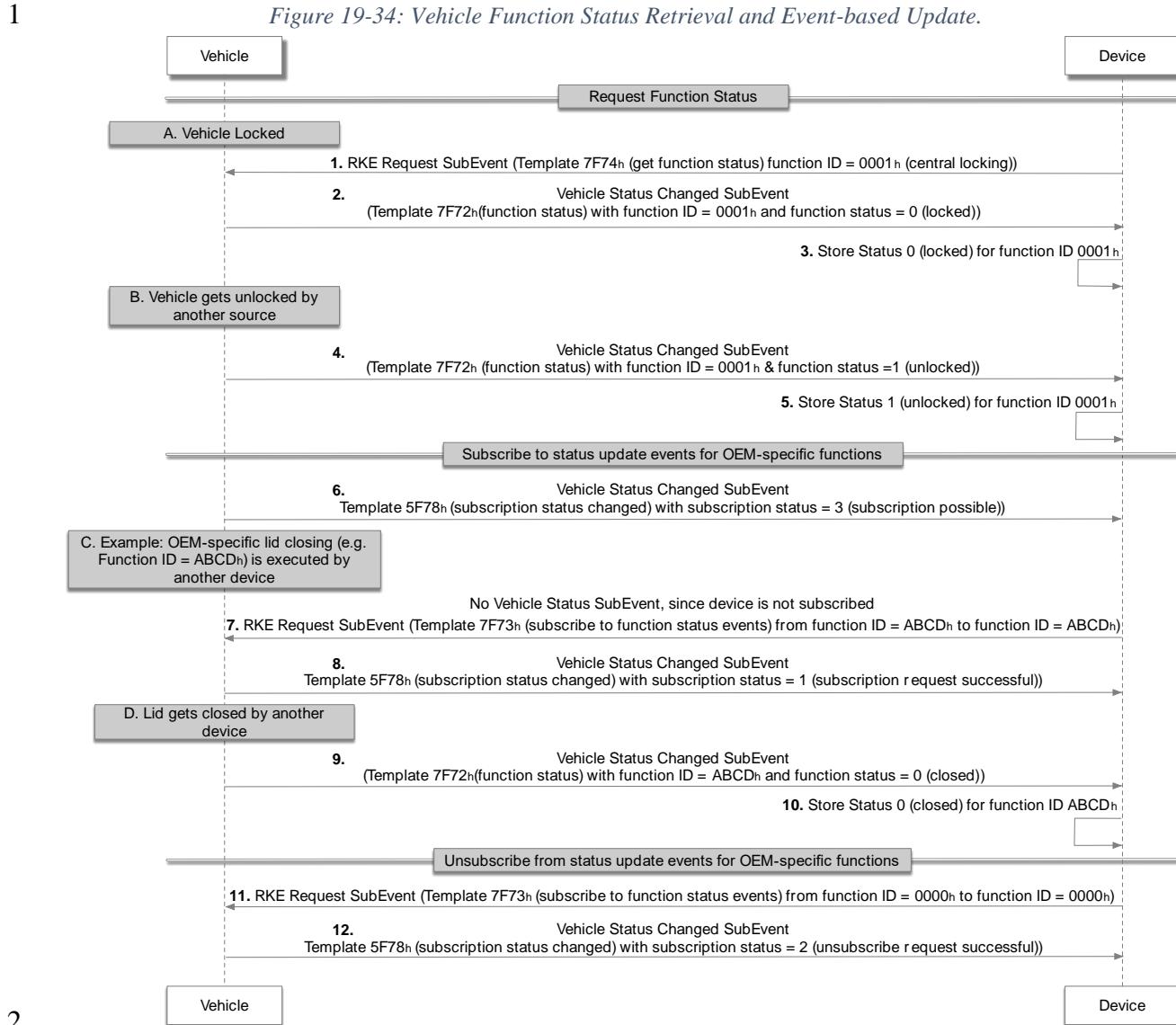
19 For the standardized functions, no subscription is necessary. The vehicle shall send an initial
20 status on connection establishment, after owner pairing, after Friend First Approach and always
21 send status updates to connected devices when the vehicle status has changed. The explicit status
22 request shall only be executed on explicit user intent or for device error recovery. Afterward,
23 changes shall be tracked via the function status updates send out on status change.

24 When entering low power mode, the vehicle shall send Vehicle Status Changed SubEvent
25 indicating it is in low power mode to all connected devices. If the device has received a Vehicle
26 Status Changed SubEvent indicating the vehicle is in a low power mode, the device shall not
27 send a RKE Request SubEvent for status request updates except for device error recovery. After
28 exiting lower power mode, the vehicle shall send a status update for all connected devices.

29 If the vehicle is in low power mode, the status request shall not be sent except for device error
30 recovery. After establishing the Bluetooth LE connection, the device may subscribe to status
31 updates, only when the vehicle indicates first the availability of the subscription mechanism (see
32 message in step 6 of Figure 19-34). The vehicle shall indicate when it stops the automatic status
33 updates for the subscribed ranges (e.g., due to entering low power mode) and when a
34 resubscription is possible again.

35 When a function is not supported by the vehicle, the vehicle shall only send a status update
36 informing about the unavailability (cf. Table 19-80, status F0_h), when the status is explicitly
37 requested by the device (“get function status”).

38 If the Bluetooth LE connection between the device and the vehicle is available and the
39 subscription to a certain function id range is active, the framework shall store the latest received
40 function status values for this range of function ids. The device shall provide an API for eligible
41 vehicle OEM apps for subscribing to a certain function id range, requesting a certain function
42 status value from the vehicle and reading the stored status data.

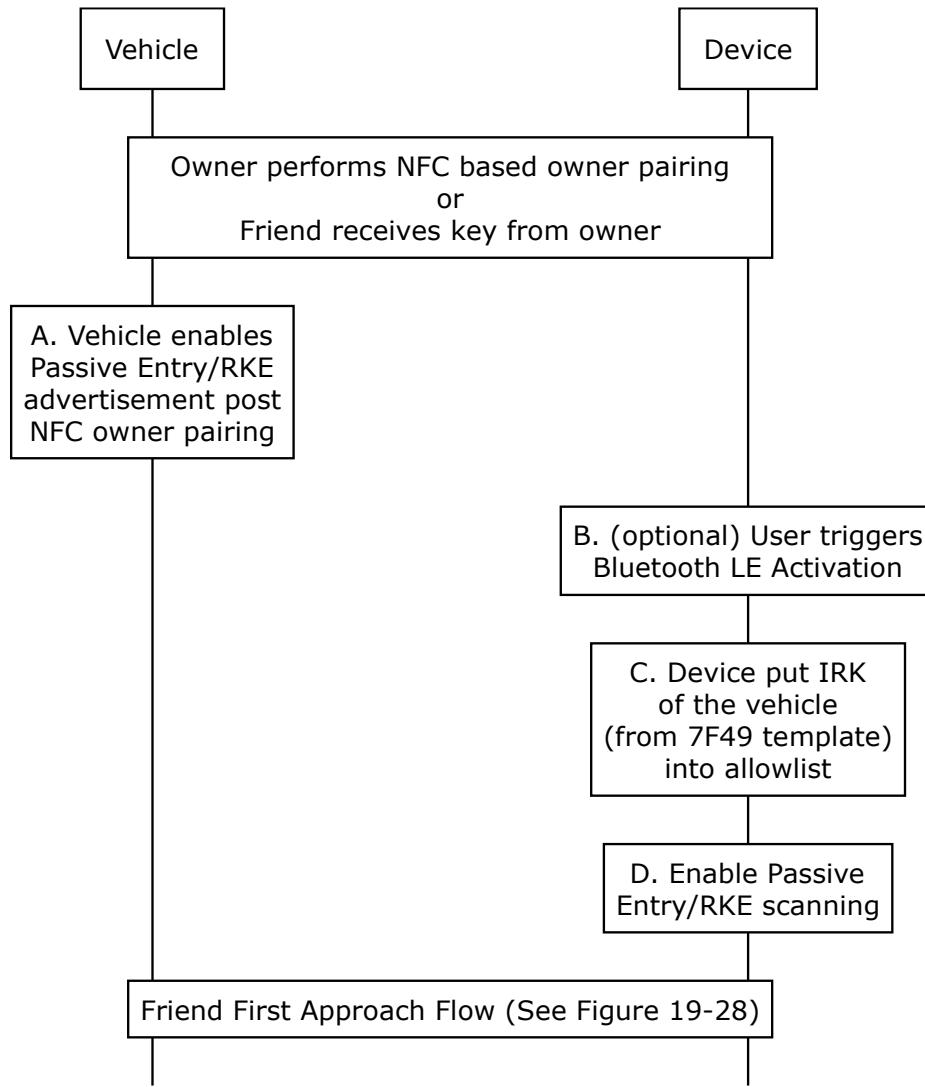


3 19.5.10 Bluetooth LE Activation Flow

- 4 If a user goes through NFC based owner pairing or friend first approach even though both device
5 and vehicle support Bluetooth LE/UWB capability, DK solution allows the user to migrate NFC
6 provisioned key to Bluetooth LE/UWB based passive entry experience without user having to
7 perform owner pairing or key sharing again.
- 8 If a user goes through NFC based owner pairing or friend first approach for a DK system which
9 supports Bluetooth LE without UWB, DK solution allows the user to migrate NFC provisioned
10 key to Bluetooth LE based RKE experience without user having to perform owner pairing or key
11 sharing again
- 12 This NFC to Bluetooth LE migration may be triggered by user anytime even when the vehicle is
13 not within the Bluetooth LE proximity. Upon first approach after user triggered migration,
14 device and vehicle shall establish Bluetooth LE bonding and offer passive entry or RKE
15 experience for that approach.

1 Figure 19-35 illustrates Bluetooth LE Activation.

2 *Figure 19-35: First Approach Activation.*



3

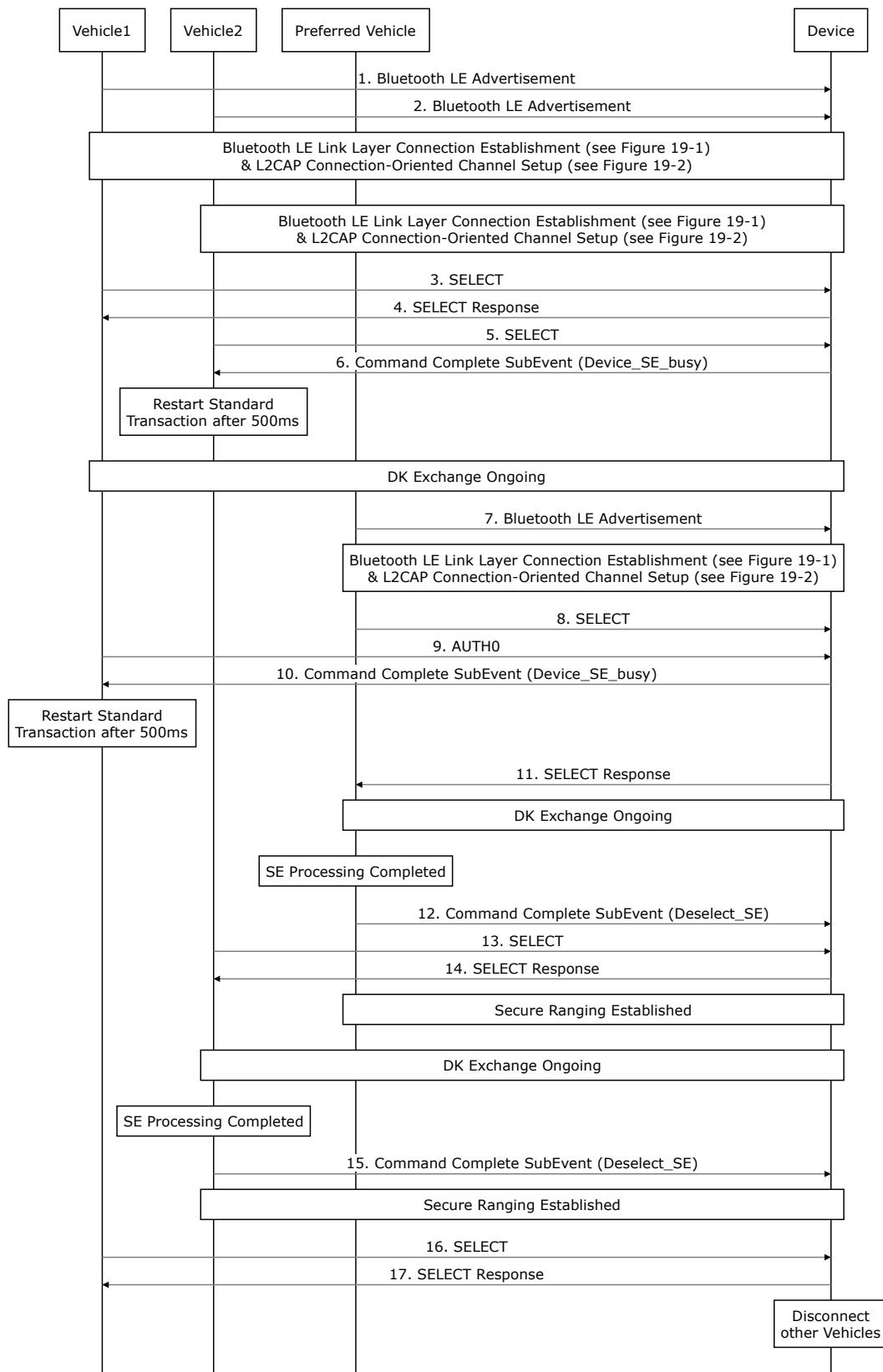
4 19.6 Digital Key - Preference Management

5 19.6.1 Preference Management: Connected Vehicles < Connection Limit

6 A user with multiple Digital Keys can mark any one of the Digital Keys as preferred or default.
7 When such user approaches multiple valid vehicles, the device shall prioritize the SE access for
8 vehicle with the default Digital Key and put the other vehicle's SE request on hold by sending
9 'Device_SE_busy', if they compete for resources with the preferred Digital Key. When vehicle
10 receives 'Device_SE_busy' SubEvent, it shall wait for a defined wait time (e.g. two seconds)
11 before restarting the standard transaction again. Once the device has completed SE processing
12 with the preferred vehicle, it shall continue with the rest of the vehicles. Figure 19-36 provides
13 an example flow for device preference management.

1

Figure 19-36: Device Preference Management.



2

1 **19.6.2 Preference Management: Connected Vehicles >= Connection Limit**

2 When a device is already connected to its max limit Bluetooth LE connections and user
3 approaches (within Bluetooth LE range) additional vehicle for which it has a valid Digital Key
4 with the intent to access it, user may go through one of the following two scenarios:

- 5 1. New Vehicle is Preferred
- 6 2. New Vehicle is Non-Preferred

7 For scenario #1, when the device detects advertisement from a new vehicle which is configured
8 as preferred vehicle for the device, it may drop one of the connected vehicles to prioritize the
9 Bluetooth LE connection for the preferred vehicle.

10 For scenario #2, when the device detects advertisement from a new non-preferred vehicle, it shall
11 drop one of the non-preferred connected vehicles (which connection to drop is device OEM's
12 implementation) to prioritize the Bluetooth LE connection for the new vehicle

13 When there are more vehicles than device's max Bluetooth LE connection limit, user shall be
14 able to exercise passive entry experience with any of the connected vehicle. However, for non-
15 connected, non-preferred vehicle, user may have to fallback to RKE feature.

16 To exercise RKE feature in this scenario, user selects the right vehicle and requests RKE feature.
17 This shall prioritize the non-connected vehicle for Bluetooth LE connection for user to gain
18 access. Once connected and unlocked, vehicle should try to establish secure ranging with newly
19 connected device to move the user to passive entry experience and for device in/out detection for
20 engine start

21 **19.7 SubEvent Handling**

22 **19.7.1 Command Complete SubEvent Code Handling**

23 This section defines how vehicle shall recover from a SubEvent defined in Section [19.3.8](#).

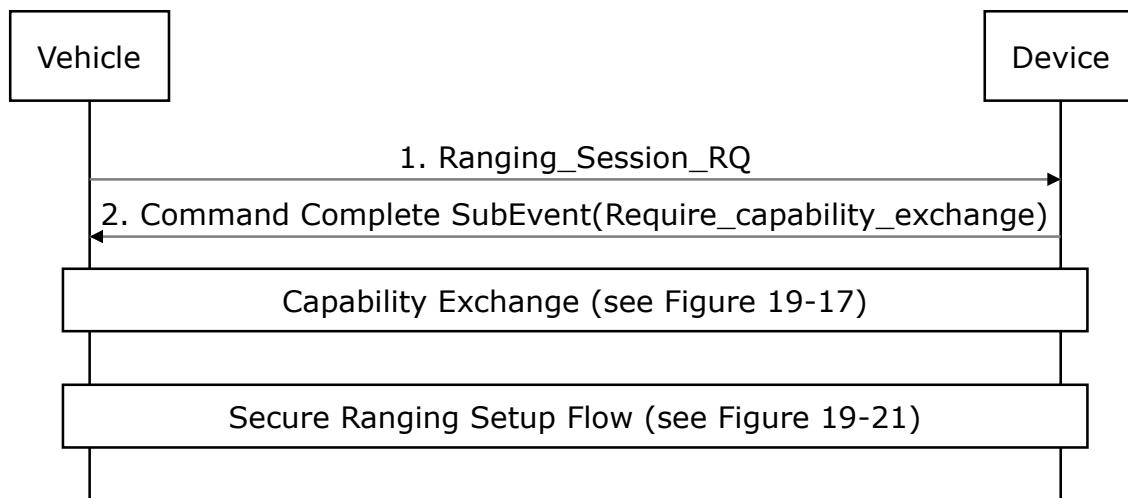
24 **19.7.1.1 Require Capability Exchange**

25 Ranging capabilities are first negotiated between device and vehicle as part of the owner pairing
26 flow (See Section [19.5.1](#)) using Ranging_Capability_RQ and Ranging_Capability_RS.

27 Figure 19-37 provides an example flow for handling Require Capability Exchange SubEvent.

1

Figure 19-37: Required Capability Exchanged.



2

3 19.7.2 *Ranging Session Status Changed SubEvent*

4 19.7.2.1 *URSK Not Found*

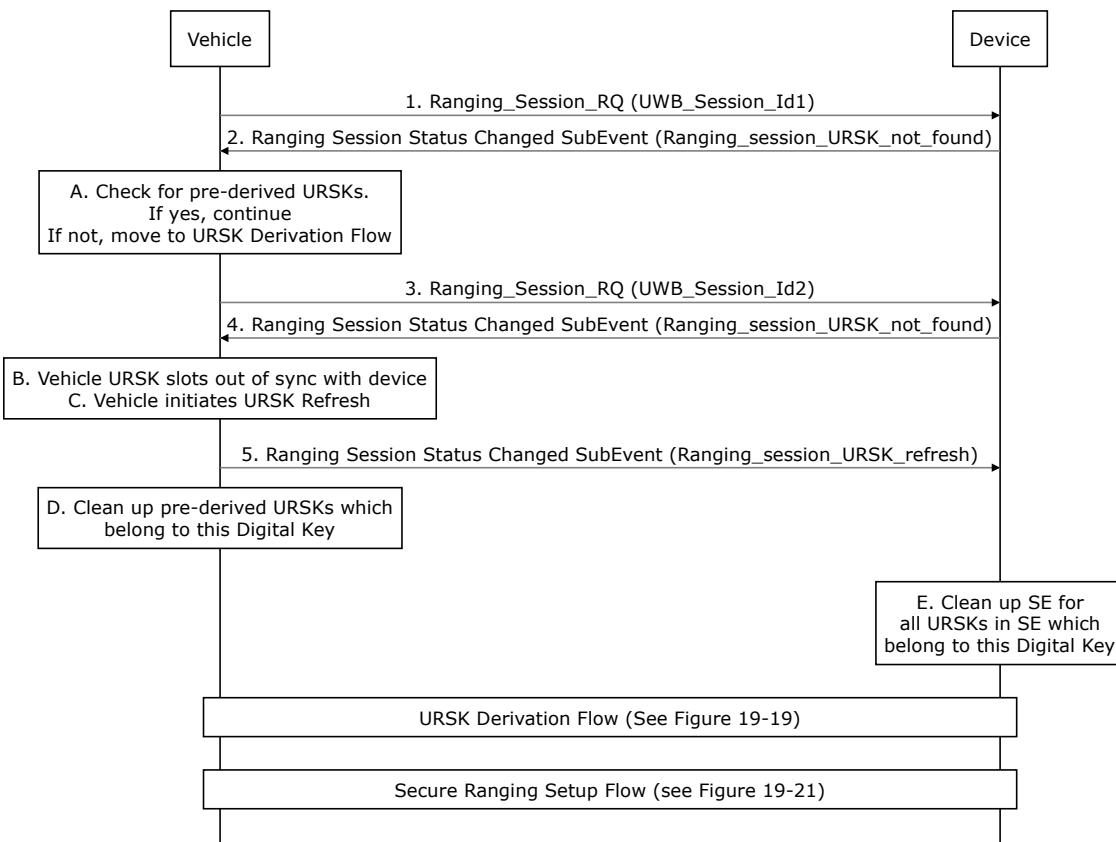
5 When a device fails to retrieve the associated URSK for a requested UWB_Session_Id, it shall
6 respond with a ‘Ranging Session Status Changed SubEvent’ with parameter value
7 “URSK_not_found”.

8 The vehicle should clean up the pre-derived URSKs on the device if it fails to retrieve URSKs
9 repeatedly.

10 Figure 19-38 provides an example flow for URSK Retrieval Failure handling.

1

Figure 19-38: URSK Not Found.

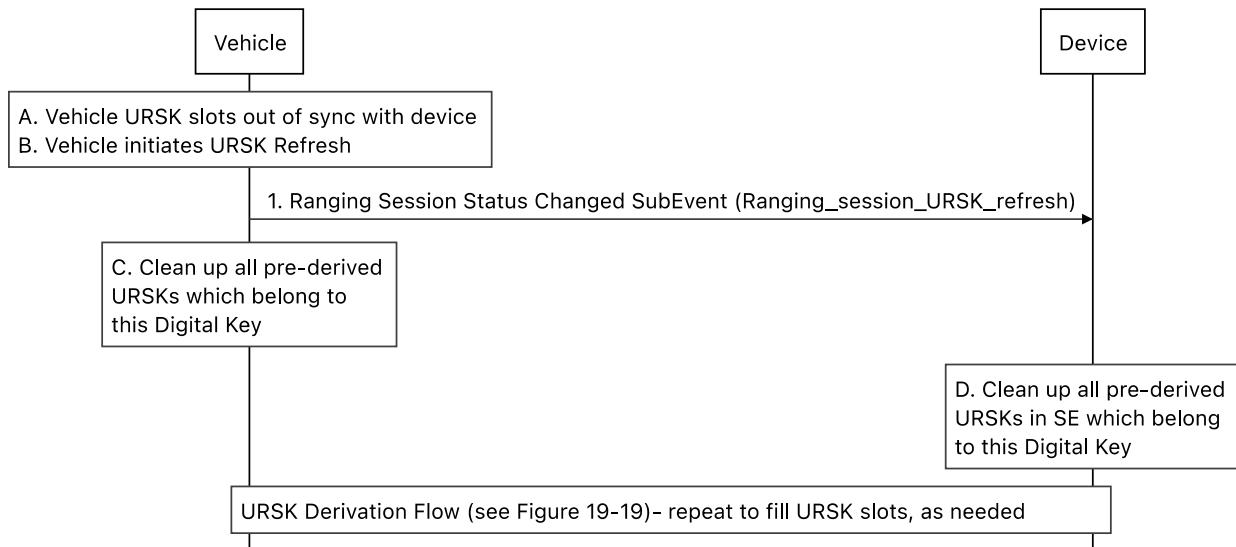


2

- 3 For box A, vehicle shall query for stored URSKs.
- 4 For box B, it is vehicle's decision on when to initiate URSK Refresh flow.
- 5 For box D, vehicle should delete URSKs associated with the digital key.
- 6 **19.7.2.2 URSK Refresh**
- 7 The vehicle may trigger a deletion of all pre-derived URSKs in the device's SE at any time by sending a Ranging Session Status Changed SubEvent Notification with Session_Status "Ranging_session_URSK_refresh". This method shall not affect the URSK used in the active ranging session.
- 11 Figure 19-39 provides an example flow for URSK Refresh flow.

1

Figure 19-39 URSK Refresh Flow



2

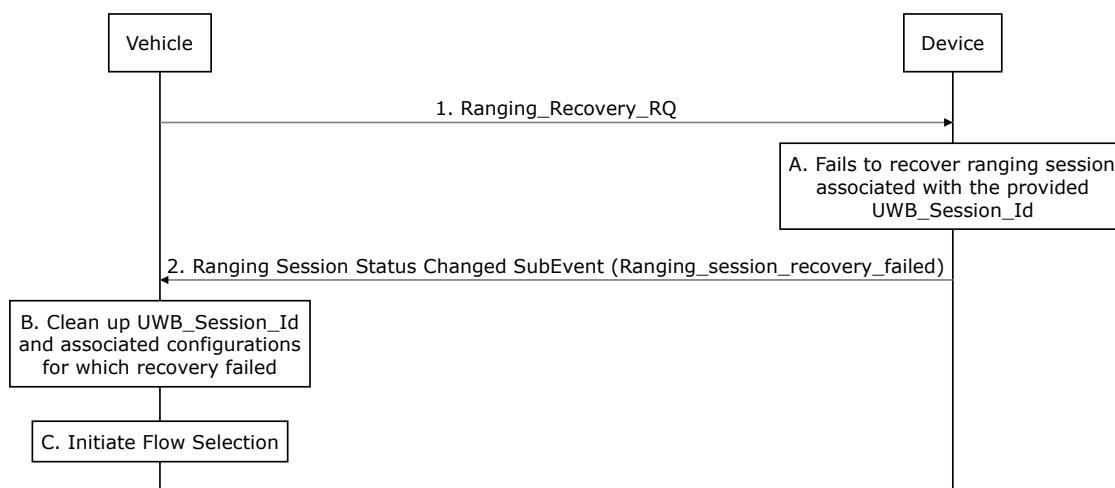
19.7.2.3 Recovery Failed

When the device fails to recover a ranging session, which was requested by vehicle, the device shall respond to the vehicle with Ranging Session Status Changed SubEvent Notification with Session_Status “Ranging_session_recovery_failed”. Upon receiving this notification, the vehicle shall discard the corresponding URSK. A new ranging session should then be established according to the flow selection (see Figure 19-22).

Figure 19-40 provides an example flow for “Recovery Failed” handling.

10

Figure 19-40: Recovery Failed Flow.



11

For box B, process the received message, the vehicle removes the URSK, configurations and any other metadata associated with UWB_Session_Id for which recovery failed.

For box C, refer to Section [19.5.5](#).

19.8 Error Handling

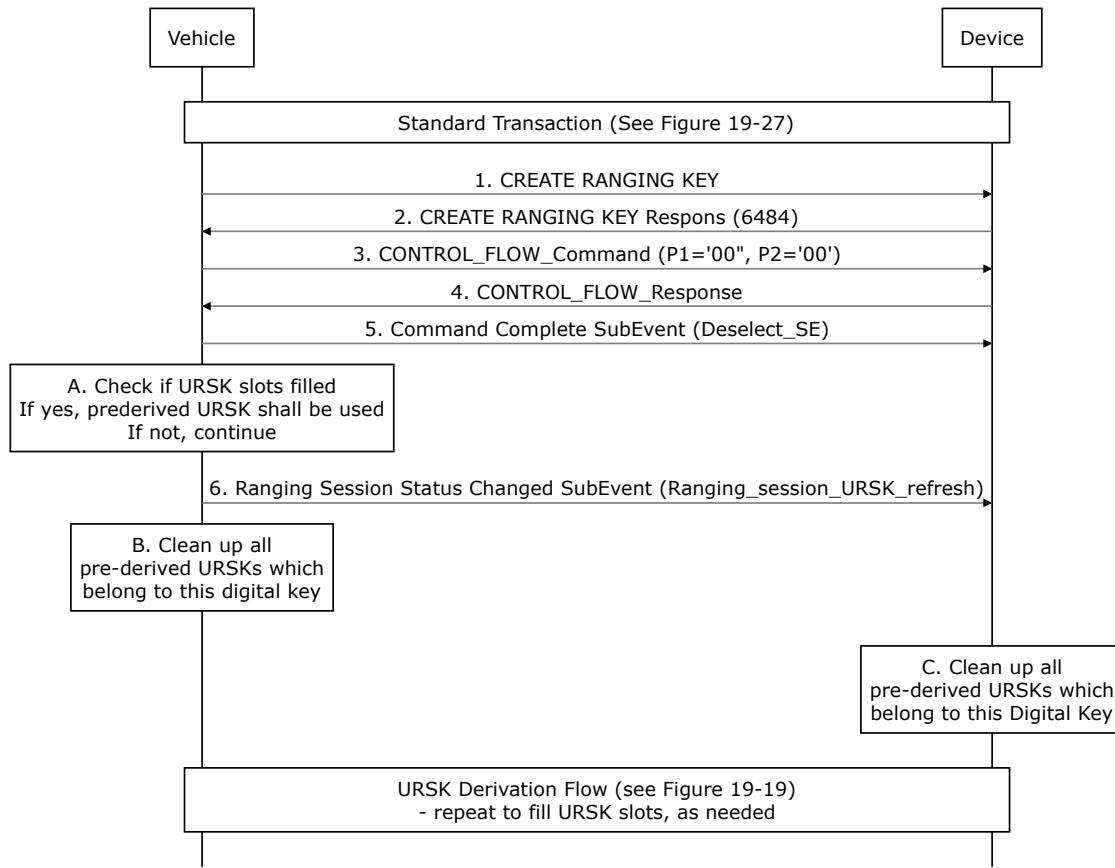
19.8.1 SE Transaction Recovery

19.8.1.1 CREATE RANGING KEY Failure

If device URSK slots for a given Digital Key are filled, it shall fail to store any new key. In this event, the device shall respond to CREATE RANGING KEY command with Status Word 6484_h, which means ‘URSK slots have been exhausted’. Upon receiving status word 6484_h, vehicle shall not request new URSK derivation unless it requests “URSK_refresh” as outlined in Figure 19-41 first.

If vehicle identifies that device and vehicle are out of sync with pre-derived URSKs, vehicle may go through URSK Refresh flow (see Figure 19-39). Figure 19-41 provides an example flow for Create Ranging Key Failure.

Figure 19-41: URSK Refresh due to Status Word 6484_h in CREATE RANGING KEY Response.



13

- 14 For box A, vehicle shall query for pre-derived URSKs.
- 15 For box B, vehicle shall delete all pre-derived URSKs associated to the Digital Key.
- 16 For box C, device shall delete all pre-derived URSKs associated to the Digital Key.

- 1 *19.8.1.2 Others*
- 2 Error handling for rest of the applet commands shall be done based on the Status Words and flows defined in Section [15](#).

1 **20 UWB MAC AND CHANNEL ACCESS [WCC3]**

2 This section defines the MAC layer and channel access protocols for UWB ranging for Digital
3 Key.

4 **20.1 UWB MAC Architecture**

5 **20.1.1 Overview**

6 This subsection describes the concepts that are used in the UWB MAC (see [31] and [33]).

7 **20.1.1.1 Ranging Roles Definition**

8 In the DK UWB ranging service, the ranging device roles are based on which device starts the
9 ranging procedure and is responsible for setting the ranging exchange. The following role
10 definitions apply to UWB layer only:

- 11 1. An entity that starts the UWB ranging packet exchange by sending a first UWB POLL
12 packet is called an “initiator.” In the case of DK this is the device.
- 13 2. An entity that responds to a UWB POLL packet is called a “responder”. In the case of
14 DK, these are the anchors on the vehicle.
- 15 3. An entity that includes a number of responders is called a “responder-device”. In the case
16 of DK, this is the vehicle.
- 17 4. An entity that controls the ranging procedure by sending a Pre-POLL packet is called the
18 controller. In the case of DK, this is the device, i.e. the device is the initiator and the
19 controller

20 Note that the roles definition above may not apply to the Bluetooth LE layer.

21 **20.1.1.2 Logical and Physical Responders**

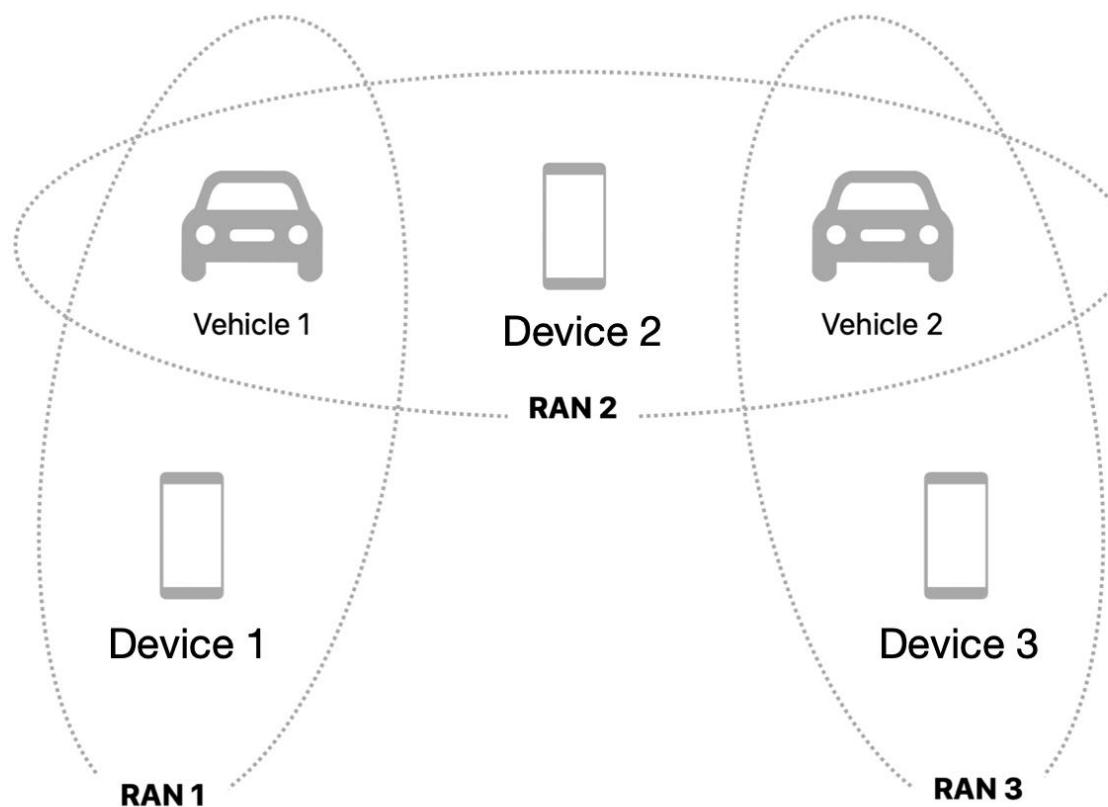
- 22 1. A responder can be a logical responder as well as a physical one.
- 23 2. A physical responder shall have one UWB module and one or more physical antennas.
- 24 3. A logical responder corresponds to one responder role which, for example, could be
25 assigned to a specific UWB module and one specific physical antenna. Hence, a physical
26 responder may include one or more logical responders.
- 27 4. The responder-device shall coordinate which logical responders transmit and in which
28 order such that there is no interference between transmissions from two logical
29 responders belonging to the same responder-device.

30 **20.1.1.3 DK Ranging Area Network (RAN)**

- 31 1. An initiator and a responder-device engaged in a continuous ranging procedure that is
32 characterized by a specific set of parameters is called a ranging session.
- 33 2. An initiator and its (one or more) ranging sessions are called a “ranging area network
34 (RAN).” Each RAN is characterized by a timing reference that is established by the
35 initiator. All responder-devices in the same RAN approximate the initiator timeline (there
36 is no assumption of global synchronization here, see Section [20.3](#)).

- 1 3. A single RAN shall have one initiator only and may have more than one responder-device (e.g. a device ranging with two vehicles). In the example shown in Figure 20-1, this is represented by Device 2, Vehicle 1, and Vehicle 2 which all belong to RAN 2.
- 2 4. Each responder-device may have a different number of logical responders (e.g. one vehicle may have 7 responders and another vehicle in the same RAN may have 5 responders).
- 3 5. A single responder-device may belong to multiple RANs, e.g. a single vehicle ranging with two different devices. In the example shown in Figure 20-1, this is represented by Vehicle 2 which belongs to RAN 2 controlled by Device 2 and RAN 3 controlled by Device 3.

11 Figure 20-1: Digital Key RAN Concept.



- 12 20.1.1.4 *Inter-/Intra- RAN Interference and Resource Management*
13 Each responder on a responder-device can reliably predict its allowed transmission window and there will be no interference between responders on the same responder-device. However, given the above definitions, three possible performance degradation scenarios can be identified:
 - 17 1. Inter-RAN Interference: - 18 • This may be caused by actual over-the-air collisions of packets from different RANs.

- 1 • This is the normal mode of operation and should be expected as there is no assumption of
2 coordination between different RANs. The impact of these collisions may be mitigated
3 by employing the ranging round hopping strategy defined below (see Section [20.4](#)).
4

5 2. Intra-RAN Resource Conflict:

- 6 • A resource conflict may occur when the initiator would have to serve two different
7 ranging exchanges (to two different responder-devices) at the same time.
8 • This scenario may be resolved at the initiator by prioritizing one of the ranging sessions
9 involved. The target ranging round may be used for the ranging session with the highest
10 priority and the initiator shall skip the round for the other ranging sessions. The impact
11 of skipped ranging round appears like a failed ranging round and may be mitigated by
12 employing the ranging round hopping strategy explained in Section [20.4](#).
13

14 3. Inter-RAN Resource Conflict:

- 15 • A resource conflict occurs when a responder would have to serve ranging rounds from
16 two (or more) different RANs at the same time.
17 • This scenario may be resolved by prioritizing one RAN and skipping the round for the
18 other RANs. The prioritization is determined by the involved responder-device.
19 • For the initiators of the skipped RANs, this appears like a failed reception of the
20 responses for the current ranging exchange and the responder-device hops to a different
21 round (see Section [20.4](#)).
22

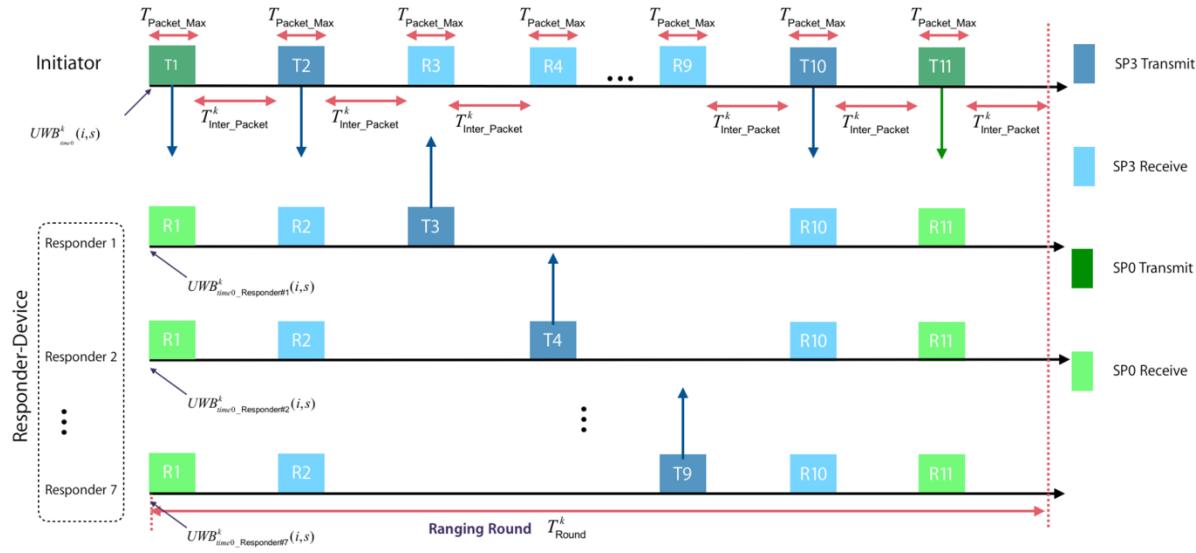
23 The criteria for determining the priority is left to the device and the vehicle manufacturer. In all
24 subsequent sections and text below, unless otherwise stated, a responder is understood to mean a
25 logical responder.

26 **20.2 MAC Time Grid**

27 The DK UWB ranging protocol is a one-to-many (O2M) ranging protocol between the initiator
28 and the responder-device. Figure 20-2 shows an example for the packet exchange in the O2M
29 ranging protocol for an initiator ranging with a responder-device with seven responders. The
30 symbols used in the figure are defined in subsequent text.

1

Figure 20-2: Digital Key One-to-Many Ranging Protocol.



2

3 The DK UWB ranging protocol is a block-based ranging approach as described in [33]. The
4 protocol uses a ranging block structure where each ranging block is T_{Block}^k in time duration (it is
5 explained later in this section how the block timing is established within the RAN). Each ranging
6 block is divided into N_{Round}^k ranging rounds that are long enough to carry out one complete
7 ranging exchange. The length T_{Round}^k of the ranging rounds is session specific.

- First, the following global parameters are defined:

- T_{Chap} : Unit of time for the MAC protocol. All durations (except for T_{Packet_Max} defined below) of the MAC protocol are integer multiples of T_{Chap} .

$$T_{Chap} = \frac{1}{3} \text{ ms} = 400 \text{ RSTU}$$

11 where the RSTU is defined in [33] as $416 / (499.2 \text{ MHz}) \gg 833.33 \text{ ns}$.

- 12
- T_{Block_Min} : Minimum ranging block duration, which is set to 96ms, is an integer multiple of T_{Chap}

$$T(\text{Block_Min}) = 288 \times T_{Chap} = 96\text{ms}$$

- 13
- One RAN consists of one initiator and $k = 1, \dots, K$ responder-devices.
 - Each responder-device is assigned at most one active ranging session in the RAN.
 - The k -th ranging session (which is associated with the k -th responder-device) is identified with a unique $UWB_Session_ID^k$ that is assigned during the negotiation phase.
 - The k -th ranging session is defined relative to a specific time reference UWB_{time0}^k . This time reference is defined by the initiator with respect to its clock base. This time reference defines the beginning of a series of consecutive ranging blocks. Given that

- 1 there is no assumption of global synchronization, Section [20.3](#) describes how that time
2 reference is established at the responders.
- 3 • The length of each ranging block for the k -th session is given by:

4 $T_{\text{Block}}^k = N_{\text{RAN}}^k \times T_{\text{Block_Min}}$

5 -

6 where N_{RAN}^k is the RAN Multiplier. It is a session specific parameter that is used to
7 control the maximum ranging frequency of the corresponding session in a given RAN
8 (see Section [20.5.2](#) and G.1). It is set during the negotiation phase between the initiator
9 and the responder-device. Every ranging session shall have a ranging block duration that
10 is greater than or equal to $T_{\text{Block_Min}}$.

- 11 • The $(i+1)$ -th ranging block of the k -th ranging session starts at:

12 $UWB_{\text{time}0}^k + i \times T_{\text{Block}}^k \quad i = 0,1,2, \dots$

13

- 14 • Each block is divided into a number of slots. The length of each slot T_{Slot}^k shall be greater
15 than or equal to the sum of the maximum UWB packet transmission time and the
16 maximum processing time required by the UWB receiver to process such a packet (these
17 are indicated by $T_{\text{Packet_MAX}}$ and $T_{\text{Inter_Packet}}$ in Figure 20-2)⁶. The duration of the slot in
18 terms of T_{Chap} is expressed as:

19 $T_{\text{Slot}}^k = N_{\text{Chap_per_Slot}}^k \times T_{\text{Chap}}$

20

21 where $N_{\text{Chap_per_Slot}}^k$ is determined during the negotiation phase (see Section [20.5.2](#)).

- 22 • A sequence of $N_{\text{Slot_per_Round}}^k$ consecutive slots constitutes a ranging round of length T_{Round}^k .
23 The number of slots per ranging round shall be greater than or equal to the number of
24 packet exchanges required for one complete ranging exchange between the initiator and
25 the responder-device.

26 $T_{\text{Round}}^k = N_{\text{Slot_per_round}}^k \times T_{\text{Slot}}^k = N_{\text{Slot_per_round}}^k \times N_{\text{Chap_per_Slot}}^k \times T_{\text{Chap}}$

27

- 28 • Each block of the k -th ranging session is divided into N_{Round}^k consecutive ranging rounds,
29 where

30
$$N_{\text{Round}}^k = \frac{T_{\text{Block}}^k}{T_{\text{Round}}^k}$$

⁶ Given that the initiator and the responder-device may be using two different UWB radios with different processing capabilities, the two radios may support different values of slot durations. The duration of the appropriate slot is selected from the set of common values during the negotiation phase between the initiator and the responder-device according to the protocol described in Section [20.5.2](#).

- 1 • The $(s+1)$ -th ranging round in the $(i+1)$ -th ranging block of the k -th ranging session shall
2 start at

3 $UWB_{time0}^k + i \times T_{Block}^k + s \times T_{Round}^k \quad i = 0,1,2,3 \dots \quad s = 0,1,2, \dots, (N_{Round}^k - 1)$

- 5 • Each of the ranging sessions ($k=1,2, \dots, K$) is assigned a pseudo-random hopping
6 sequence

7 $H^k = \{S_0^k, S_1^k, \dots, S_i^k\} \quad \text{with } 0 \leq S_i^k \leq (N_{Round}^k - 1)$

8 where S_i^k denotes the index of the ranging round in ranging block i that starts at

9 $UWB_{time0}^k + i \times T_{Block}^k + S_i^k \times T_{Round}^k \quad i = 0,1,2,3 \dots \text{ and } 0 \leq S_i^k \leq (N_{Round}^k - 1)$

10 S_i^k also referred to as *Round _Idx^k(i)* in later sections of this document to denote the
11 ranging round index of $(i+1)$ -th ranging block selected pursuant to the hopping
12 configuration. The selection of the hopping configuration, i.e. adaptive or continuous, and
13 hopping sequence will be made according to the selection made during the ranging
14 session setup described below. Additionally, Appendix G.2 shows the computation for
15 the hopping sequence H^k . The hopping sequence shall be used, along with the hopping
16 mode and the hopping flag to compute the ranging round index in the current ranging
17 block. See Section 20.4 below for details on the hopping flag and ranging round index
18 determination.

19 For details on the ranging packet exchange initiation process and the hopping
20 configuration selection, see Section 20.5.2.

- 21 • Given the above definitions, the average ranging frequency of the k -th ranging session is

22 $f_{Ranging}^k = \frac{1}{T_{Block}^k} \approx \frac{1}{96ms} @ 10.4166667 \text{ Hz}$

23 The achievable ranging block durations T_{Block}^k for a given N_{RAN}^k is $N_{RAN}^k \times 96ms$.

- 25 • For the $(i+1)$ -th ranging measurement in the k -th ranging session, the ranging round with
26 index *Round _Idx^k(i)* is divided into $N_{Slot_per_Round}^k$ consecutive slots of equal length

27 $T_{Slot}^k = N_{Chap_per_Slot}^k \times T_{Chap}$.

28 The slot with index m shall start at time

29 $UWB_{time}^k + i \times T_{Block}^k + Round_{Idx}^k(i) \times T_{Round}^k + m \times T_{Slot}^k$

30 $i = 0,1,2,3, \dots \quad 0 \leq Round_{Idx}^k(i) \leq (N_{Round}^k - 1) \quad m = 0,1,2, \dots, (N_{Slot_per_Round}^k - 1)$

- 32 • The first $N_{Packets}^k$ slots (out of $N_{Slot_per_Round}^k$) of the ranging round with index
33 *Round _Idx^k(i)* (slots $0,1,2,\dots,N_{Packets}^k - 1$) shall be mapped by the initiator and

1 responder-device to transmit and receive UWB ranging packets (see the example in Table
2 20-2). The remaining $N_{\text{Slot_per_Round}}^k - N_{\text{Packets}}^k$ slots shall be left unmapped.

- 3 The initiator and the responder-device shall send the ranging packets as close as
4 technically possible to the beginning of the mapped slots.
- 5 Neither the initiator nor the responder-device shall start transmission before the
6 beginning of a slot.
- 7 If a resource conflict between two (or more) ranging sessions in the same RAN occurs,
8 the initiator shall resolve the conflict by prioritization.

9 The MAC grid described above is illustrated in Figure 20-3. In this figure and throughout this
10 section, the following notation is used:

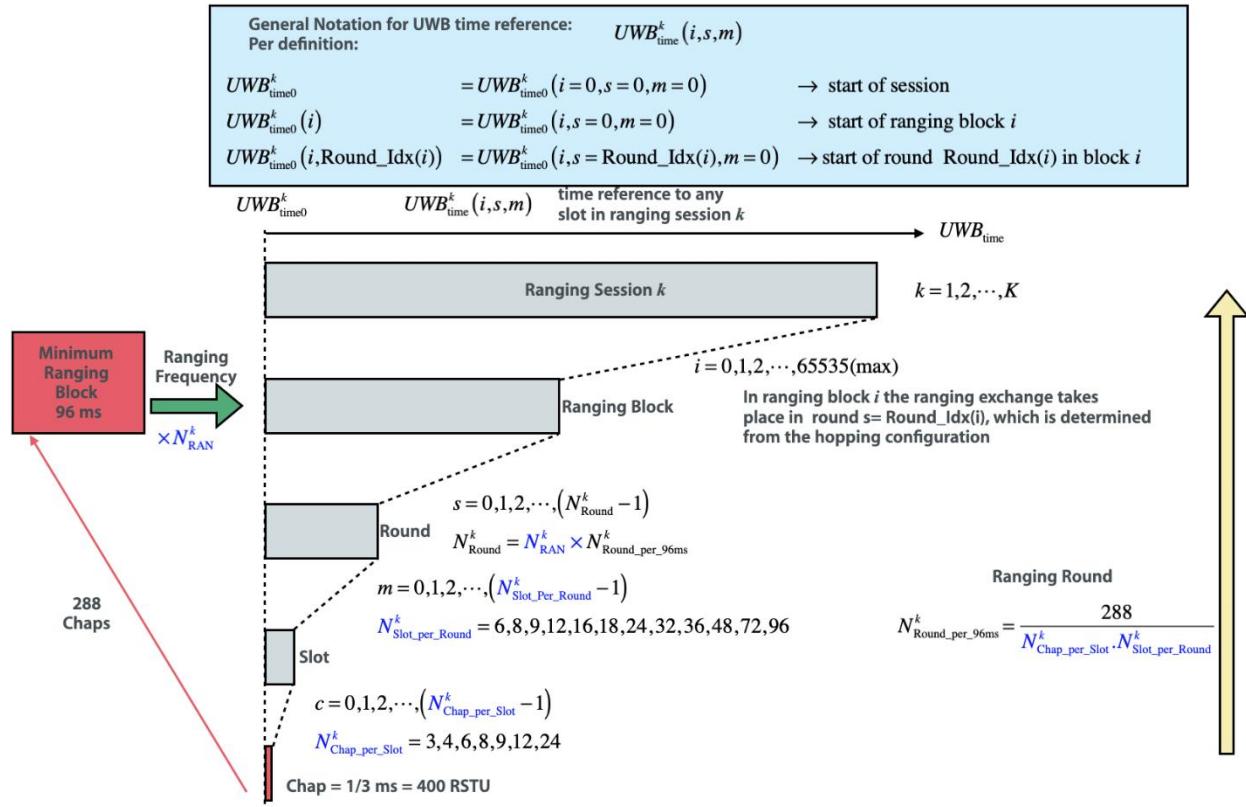
11 $UWB_{\text{time}0}^k(i,s,m)$ is the UWB time reference for the $(m+1)$ -th slot of the $(s+1)$ -th ranging round
12 of the $(i+1)$ -th ranging block in the k -th ranging session.

$$UWB_{\text{time}0}^k = UWB_{\text{time}0}^k(i=0, s=0, m=0) \rightarrow \text{start of session}$$

$$UWB_{\text{time}0}^k(i) = UWB_{\text{time}0}^k(i, s=0, m=0) \rightarrow \text{start of ranging block } i$$

$$UWB_{\text{time}0}^k(i,s) = UWB_{\text{time}0}^k(i, s, m=0) \rightarrow \text{start of ranging round } s \text{ in block } i$$

14 Figure 20-3: Overview of MAC Grid (See Table G- 1).



1

Table 20-1: Example MAC Configuration.

Parameter	Vehicle 1 (k=1)	Vehicle 2 (k=2)
N_{RAN}^k	3	2
$N_{\text{Chap_per_Slot}}^k$	6	3
$N_{\text{Responder}}^k$	2	3
T_{Block}^k	288 ms	192 ms
T_{Round}^k	12 ms	8 ms
N_{Round}^k	24	24
f_{Ranging}^k	3.47 Hz	5.21 Hz

2

3 20.3 MAC Time Grid Synchronization

4 As stated above, each ranging session is defined relative to a specific time reference $UWB_{\text{time}0}^k$,
5 which is defined relative to the initiator's internal clock. The time reference $UWB_{\text{time}0}^k$ establishes
6 the MAC time grid for the k -th ranging session.

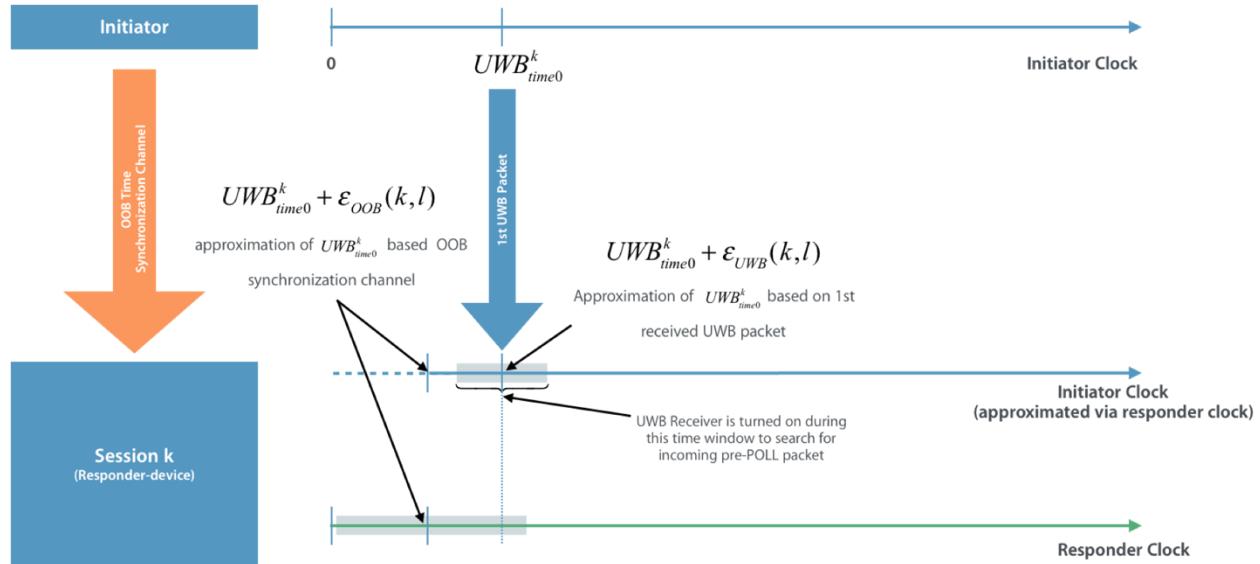
7 Given that there is no assumption of tight clock-level synchronization between the initiator and
8 the responder-device, the MAC time grid shall be estimated, with reasonable accuracy, at each
9 responder (at the responder-device). This allows each responder (at the responder-device) to
10 determine when to expect to receive the Pre-Poll message from the initiator, when to send its
11 response message back to the initiator, and when to expect to receive the Final and Final_Data
12 messages from the initiator.

13 The approximate MAC time grid is established at each responder as follows:

- 14 • During the ranging session negotiation, the initiator shall send the value of $UWB_{\text{time}0}^k$ to
15 the responder-device that is to be associated with the k -th ranging session.
- 16 • Each responder may approximate the MAC time grid of the initiator by:
 - 17 ○ The responder uses a dedicated out-of-band (OOB) clock synchronization (e.g.
18 via the Bluetooth LE control channel). By using an OOB clock synchronization
19 method, the initiator can inform the responder-device when to expect the ranging
20 session to start, i.e. the approximate MAC grid time reference $UWB_{\text{time}0_Responder\#l}^k$
21 for the session for the $(l+1)$ -th responder, $l=0,1,2, \dots, N_{\text{Responder}}^k - 1$. The
22 responders then turn on their UWB receivers and start looking for the first UWB
23 packet.
 - 24 ○ Upon receiving UWB packets, the responders further use the incoming UWB
25 packets from the initiator to continuously refine the estimation of the MAC time
26 grid under the assumption that each incoming packet transmission was not started
27 by the initiator before the beginning of its dedicated slot.
 - 28 ○ The time-of-flight (ToF) calculations shall remain unaffected from the MAC grid
29 timings.

- 1 For the k -th ranging session, the approximate MAC grid and time reference established at the l -th
 2 responder is denoted by $UWB_{time0_Responder\#l}^k$ which is related to the initiator time reference by
 3
$$UWB_{time0_Responder\#l}^k = UWB_{time0}^k + \epsilon(k, l).$$
- 4 The error $\epsilon(k, l)$ depends on whether an incoming UWB packet ($\epsilon(k, l) = \epsilon_{UWB}(k, l)$) or an OOB
 5 clock synchronization protocol ($\epsilon(k, l) = \epsilon_{OOB}(k, l)$) is used to approximate the MAC time grid.
 6 Characterization of this error and techniques to minimize it are out of scope for this specification.
 7 A high-level description of the block timing synchronization process is shown in Figure 20-4.

8 *Figure 20-4: Overview of Block Synchronization Approach.*



10 20.4 Hopping Flag and Round Index Determination

11 As stated above, the initiator in the k -th ranging session in any given RAN shall start the UWB
 12 ranging procedure in the first ranging round (ranging round 0) in the first ranging block (ranging
 13 block 0) by default. This assumes that the responder-device has either achieved block
 14 synchronization by OOB method or, if not, is permanently listening for Pre-Poll messages.

15 At the initiator, and assuming no resource conflict occurs, the $Hop_Flag^k(i+1)$ and
 16 $Round_Idx^k(i+1)$ for the next ranging block (ranging block $i+1$) will depend on the hopping
 17 mode selected during ranging session setup as follows:

- 18 • If the hopping mode is set to “no hopping,” then the initiator shall continue to use the
 19 same ranging round in ranging block i and $Round_Idx^k(i+1)$ is set as:

$$20 \quad Round_Idx^k(i+1) = Round_Idx^k(i) = 0$$

21 and $Hop_Flag^k(i+1)$ is set to 0. Note that in this case, $Hop_Flag^k(i+1)$ is irrelevant
 22 to the receiver and shall be ignored.

- 23 • If the hopping mode is set to “continuous hopping,” the initiator uses the S_{i+1}^k -th ranging
 24 round

$$Round_Idx^k(i+1) = S_{i+1}^k$$

and $Hop_Flag^k(i+1)$ is set to 1. Again, $Hop_Flag^k(i+1)$ is irrelevant to the receiver and shall be ignored.

- If the hopping mode is set to “adaptive hopping,” then at the initiator, the $Hop_Flag^k(i+1)$ and $Round_Idx^k(i+1)$ for the next ranging block (ranging block $i+1$) shall be set as follows:
 - If the initiator determines that the round is clean, i.e. no interference and ranging in the current round is successful, the initiator shall stay in the current round and set as:

$$Hop_Flag^k(i+1) = 0 \quad \text{and} \quad Round_Idx^k(i+1) = Round_Idx^k(i)$$

- If the initiator determines that there is some interference on the current round or if the ranging is not successful, the initiator shall hop to a different round:

$$Hop_Flag^k(i+1) = 1 \quad \text{and} \quad Round_Idx^k(i+1) = S_{i+1}^k$$

The initiator shall send the $Hop_Flag^k(i+1)$ and $Round_Idx^k(i+1)$ for next ranging block to the responder-device as part of the payload packet carrying time stamps Final_Data at the end of the ranging sequence.

At the responder-device, the $Hop_Flag^k(i+1)$ and $Round_Idx^k(i+1)$ for subsequent ranging rounds shall be set as follows:

```

if Final_Data packet is received
{
    Hop_Flagk(i + 1) = Final_Data.Hop_Flag

    if (Hop_Flagk(i + 1) = 0)
    {
        set Round_Idxk(i + 1) = Round_Idxk(i)
    }

    elseif (Hop_Flagk(i + 1) = 1)
    {
        set Round_Idxk(i + 1) = Si + 1k
    }

}

else
{
    set Round_Idxk(i + 1) = Si + 1k.
}

```

1 The initiator may decide to trigger the hopping (i.e. setting Hop_Flag=1) based on the
 2 interference level or packet collision rate that it measures over the current exchange and/or the
 3 number of responses that it receives back from the responder-device. In general, the exact criteria
 4 for triggering hopping at the initiator is out of scope of this specification and is left to each
 5 device manufacturer.

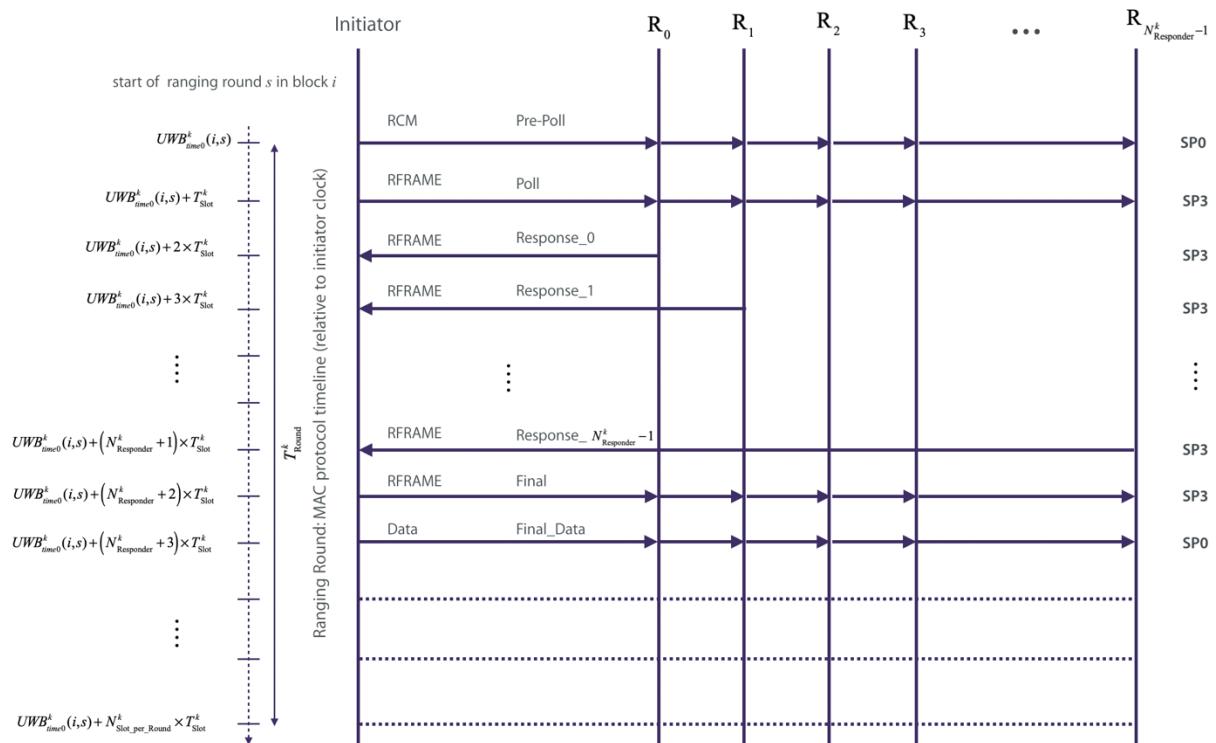
6 *Note:* The only requirement in any such criteria is that if the device does not receive any
 7 responses from the responder-device, the device shall trigger hopping.

8 20.5 MAC Protocol

9 20.5.1 Ranging Exchange Sequence

10 This section details the DK MAC protocol for a double-sided, two-way ranging with a three-
 11 packet exchange between the initiator and each responder of the responder-device. Figure 20-5
 12 shows the UWB message flow for the MAC protocol between an initiator and $N_{\text{Responder}}^k$
 13 responders in the $(s+1)$ -th ranging round in the $(i+1)$ -th ranging block with $N_{\text{Slot_per_Round}}^k$ slots.
 14 While the figure shows the MAC protocol according to the MAC timeline relative to the initiator
 15 clock and time reference, this protocol shall be carried out according to the respective responder-
 16 device's time reference as described in Section 20.3. Table 20-2 shows the mapping of the
 17 $N_{\text{Responder}}^k$ UWB ranging packets to the $N_{\text{Slot_per_Round}}^k$ slots.

18 *Figure 20-5: Example of UWB Message Flow for MAC Protocol.*



1

Table 20-2: Mapping of Slots to UWB Ranging Packets.

Slot Index	Mnemonic	STS Packet Format	Sender
0	Pre-POLL	SP0 : Data Packet	Initiator
1	POLL	SP3 : RFRAFME	Initiator
1+1	Response_0	SP3 : RFRAFME	Responder
1+2	Response_1	SP3 : RFRAFME	Responder
...	...	SP3 : RFRAFME	Responder
$1 + N_{\text{Responder}}^k$	Response_ $N_{\text{Responder}}^k - 1$	SP3 : RFRAFME	Responder
$1 + N_{\text{Responder}}^k + 1$	Final	SP3 : RFRAFME	Initiator
$1 + N_{\text{Responder}}^k + 2 = N_{\text{Packets}}^k$	Final_Data	SP0 : Data Packet	Initiator
$1 + N_{\text{Responder}}^k + 3$	Slot not used	N/A	N/A
...	Slot not used	N/A	N/A
$N_{\text{Slot_per_Round}}^k - 1$	Slot not used	N/A	N/A

2 The MAC protocol sequence under the normal mode of operation (i.e. both initiator and
3 responder-device are engaged in an active ranging session) is as follows:

- 4 1. The first packet sent by the initiator is a data packet called Pre-POLL message at
5 UWB_{time0}^k . Note that, as mentioned above, the ranging starts in ranging round 0 in
6 ranging block 0 by default when a ranging session is started or recovered. This first
7 packet is a SP0 packet. Its payload shall include:
 - 8 a. $UWB_Session_ID^k$: ID of the current ranging session.
 - 9 b. $STS_Index^k(i, Round_Idx^k(i), POLL)$: STS index of the succeeding POLL
message.
 - 10 c. i : ranging session current block index
 - 11 d. $Hop_Flag^k(i)$: Hopping flag as set from the previous ranging exchange. If
12 current ranging round is the first ranging round after start or recovery of the
13 ranging session, then $Hop_Flag^k(i = 0)$ is set to ‘0.’ Note that this field is only
14 analyzed by the receiver if the hopping mode is set to “adaptive hopping.”
 - 15 e. $Round_Idx^k(i)$: Round index for the $(i+1)$ -th ranging block as set in Final_Data
16 packet of the previous ranging exchange (i -th ranging block). If the current
17 ranging round is the first in the started or recovered ranging session, then
18 $Round_Idx^k(i = 0)$ is set to ‘0.’

20 The Pre-POLL message and its parameters are defined in Table 20-3 and Table 20-4. All values
21 of the parameters listed in Table 20-4 shall be encoded in little endian format.

1

Table 20-3: Pre-Poll Request Message and its parameters.

UWB MAC Message	UWB MAC Message ID	Parameters
Pre-POLL	1	UWB_Session_ID, Poll_STS_Index, Ranging_Block, Hop_Flag, Round_Index

2

Table 20-4: The definition of parameters in the Pre-Poll Message.

Parameters	Length (bytes)	Value	Description
UWB_Session_ID	4	0 - (2 ³² -1)	ID of the UWB ranging session
Poll_STS_Index	4	0 - (2 ³¹ -1)	STS index of the succeeding POLL message
Ranging_Block	2	0 - 65535	Session's Index of current ranging block
Hop_Flag	1	0:No Hopping 1:Hopping	Hop flag for current ranging block as set from the ranging exchange in the previous ranging block For no hopping configuration this field is always 0 For continuous hopping configuration this field is always 1 (except for the first ranging block with index 0 after start or recovery of a ranging session).
Round_Index	2	0-65535	The ranging round index for the current ranging block as set from the ranging exchange in the previous ranging block. This field is set to 0 in first ranging block where we always start in ranging round 0.

- 3 2. After a period of T_{Slot}^k measured from $UWB_{\text{time}0}^k$, the initiator shall start sending a POLL
4 message. This shall be an SP3 type packet (see Section 21).
5 3. Each responder l ($l = 0, \dots, N_{\text{Responder}}^k - 1$) shall send its response packet in its dedicated
6 response slot.
7 a. Each message shall be sent as a SP3 type packet. The index to the STS used is
8 related to the POLL message sent in step 2 above (see Section 20.6).
9 b. If any of the responders does not receive the POLL message in the current ranging
10 round from the initiator, that responder shall not transmit during its dedicated
11 response slot.
12 4. If at least one valid response has been received by the initiator:
13 a. After $(N_{\text{Responder}}^k + 2)$ slots, the initiator shall send its Final message. This packet is
14 an SP3 packet and it completes the time-of-flight measurements. The initiator
15 may optionally send its Final message together with the following Final_Data
16 packet, if no valid response has been received. In the case the Final message and
17 Final_Data packet is skipped:
18 • The hopping mode for both the initiator and the responder shall be triggered if
19 the adaptive hopping mode is active.
20 • The Frame Counter value shall not be incremented for the skipped Final Data
21 packet.

- 1 b. After $(N_{\text{Responder}}^k + 3)$ slots, the initiator shall complete the ranging exchange by
 2 sending a Final_Data packet to the responder-device. This packet is an SP0 type
 3 packet and the payload shall include the following:
 4
 - $UWB_Session_ID^k$: id of the current ranging session.
 - i : index of the current ranging block.
 - $Hop_Flag^k(i+1)$: Hopping flag to be used in ranging block $i+1$. Note that this field is
 7 only relevant if the hopping configuration is set to adaptive hopping.
 - $Round_Idx^k(i+1)$: ranging round index of the next ranging exchange in ranging block
 9 $i+1$.
 - $STS_Index^k(i, Round_Idx^k(i), FINAL)$: STS index of the preceding FINAL message.
 - $Final_Time_Stamp^k(i)$: the time stamp for the Final message. This time stamp shall be
 12 calculated as the difference between the RMARKER of the initiator POLL message and
 13 the RMARKER of the initiator Final message
 - $N_{\text{Rx_Responses}}$: Number of received responses in the current ranging exchange.
 15 [$Time_Stamps^k(l)$]: All the ranging measurement time stamps for all responders, up to
 16 $N_{\text{Responder}}^k$, whose valid ranging responses have been received at the initiator. The initiator
 17 may additionally add the time stamp data of responders where no valid response has been
 18 received. The size of the payload depends on the number of added time stamp data. The
 19 time stamp data for each responder shall include the following fields:
 - o Responder_Index l , $l = 0, 1, \dots, N_{\text{Responder}}^k - 1$, corresponding to responder slot l (slot
 21 index $2+l$ in the ranging round).
 - o Ranging_Timestamp_Responder value for $(l+1)$ -th responder. The time stamp
 23 value shall be calculated as the difference between the RMARKER of the initiator
 24 POLL message and the RMARKER of the $(l+1)$ -th responder's response message
 25 (see Section 21.6 for details). In the case of no valid response was received from
 26 the responder but the time stamp data are added for this responder, the time stamp
 27 value shall be set to 0.
 - o Ranging_Timestamp_Uncertainty_Responder parameter.
 - o Ranging_Status_Responder parameter is set to 0x0 (success) in case of a valid
 30 response was received, otherwise this parameter is set to one of the values shown
 31 in Table 20-7 to show the reason for the error.
- 32 The Final_Data message and its parameters are defined in Table 20-5 and Table 20-6.

Table 20-5: Final_Data Message and its parameters.

UWB MAC Message	UWB MAC Message ID	Parameters
Final_Data	2	UWB_Session_ID, Ranging_Block, Hop_Flag, Round_Index, Final_STS_Index,

UWB MAC Message	UWB MAC Message ID	Parameters
		Ranging_Timestamp_FINAL_TX, Number_Ranging_Responders, Responder_Index, Ranging_Timestamp_Responder_l, Ranging_Timestamp_Uncertainty_Responder_l, Ranging_Status_Responder_l, ...

1

Table 20-6: The definition of parameters in the Final_Data Message.

Parameters	Length (bytes)	Value	Description
UWB_Session_ID	4	0-(2^{32} -1)	ID of the UWB ranging session
Ranging_Block	2	0-65535	Index of current ranging Block
Hop_Flag	1	0: No Hopping 1: Hopping	Hop Flag for next ranging Block For no hopping configuration this field is always 0 For continuous hopping configuration this field is always 1
Round_Index	2	0 – 65535	The ranging round in which the ranging cycle will be executed in the next ranging block
Final_STS_Index	4	0 – (2^{31} -1)	STS index of the preceding Final message.
Ranging_Timestamp_FINAL_TX	4	Range = 0 to 0xFFFFFFFF Timestamp = N × 1/128 × 1/499.2e6 sec Time Range = 0 to 67.21 ms	Time difference between POLL and Final messages transmit times at the initiator.
Number_Ranging_Responders	1	0-255	Number of timestamps to follow in this message
Responder_Index	1	0-255	Index of Responder l whose timestamp data is referred to in the Ranging_Timestamp_Responder_l, Ranging_Timestamp_Uncertainty_Responder_l, and Ranging_Status_Responder_l fields
Ranging_Timestamp_Responder_l	4	Range = 0 to 0xFFFFFFFF Timestamp=N × 1/128 × 1/499.2e6 sec Time Range = 0 to 67.21 ms	Time difference between POLL and RESPONSE of Responder l , as received by the initiator.
Ranging_Timestamp_Uncertainty_Responder_l	1	See Section 6.9.1.7 in [33]	Range of values from 1.5 cm–3.6 m at various confidences
Ranging_Status_Responder_l	1	See Table 20-7	The status for the response frame from the responder. See the corresponding description for the ranging status in Table 20-7
...	Repeat parameters Responder_Index to Ranging_Status_Responder_l for each responder whose time stamp data has been added (up to Number_Ranging_Responders times)

All values of the parameters listed in Table 20-6 shall be encoded in little endian format. Additionally, with the above definition of the Final_Data message and with a 127 maximum payload size, the maximum number of responders $N_{\text{Responder}}^k$ that can be involved in a single ranging round is 10 responders. In case $N_{\text{Chap_per_Slot}}^k = 24$, then the maximum number of responders $N_{\text{Chap_per_Slot}}^k$ in a single ranging round is limited to 7 due to the maximum time stamp value for the Final message. A vehicle may have any number of responders Nv (physical and/or logical) and this number may be >10 responders. However, the vehicle shall set the value of the number of responders it wants to involve in each ranging round $N_{\text{Responder}}^k$ to a number that is less than or equal to 10 in the ranging session setup procedure. It is up to the vehicle to determine which set of its responders it will involve in the ranging exchange in each ranging round.

The Ranging_Status_Responder_l parameter uses the values shown in Table 20-7.

Table 20-7: Ranging Status of a Responder.

Ranging Status	Value	Description
Success	0x0	Successful receipt and transmission of a packet
Transaction overflow	0x1	RESPONSE SP3 frame from this responder cannot be processed by the device
Transaction expired	0x2	No RESPONSE SP3 frame was received from this responder
Incorrect frame	0x03	The RESPONSE SP3 frame received from this responder was not correct
Reserved	0x04-0xFF	

20.5.2 Ranging Session Setup

The following procedure describes how the MAC parameter negotiation shall be carried out between the initiator and the responder-device. First, the responder-device and the initiator shall perform a ranging capability exchange step over Bluetooth LE link, where the responder-device sends a RC-RQ message to the initiator and the initiator replies with a RC-RS message, as defined in Sections [19.3.1.1](#) and [19.3.1.2](#), respectively. Once this exchange is successful, the following negotiation steps shall be carried out over the Bluetooth LE control channel to setup the ranging session:

- Vehicle initiates the ranging session setup by sending the Ranging Session Request message defined in Section [19.3.1.3](#) with the following content:
 - Selected_Protocol_Version*^k: selected protocol version from the ranging capability exchange.
 - Selected_UWB_Config_ID*^k: selected UWB configuration from the ranging capability exchange.
 - UWB_Session_ID*^k: id of the current ranging session.
 - Selected_PulseShape_Combo*: device selected single pulse shape combination from list of common supported ones.
 - Channel_BitMask*: $\{\text{CH_IDX}\}_{\text{Responder}}$ list of available UWB RF channels.

- 1 2. Device responds by sending the Ranging Session Response message (see Section
 2 [19.3.1.4](#)) with the following contents:
- 3 • *RAN_Multiplier*: RAN multiplier N_{RAN}^k sets the minimum ranging block duration for
 4 the k -th ranging session that can be supported by the initiator (i.e. sets the maximum
 5 ranging frequency that can be supported by the initiator for this ranging session)

$$T_{\text{Block_RAN}}^k = N_{\text{RAN}}^k \quad T_{\text{Block_Min}} = N_{\text{RAN}}^k \cdot 96 \text{ ms}$$

- 6
- $$f_{\text{Ranging_RAN}}^k = \frac{1}{T_{\text{Block_RAN}}^k} = \frac{10.4166667}{N_{\text{RAN}}^k} \text{ Hz}$$
- 7 • *Slot_BitMask*: $\{N_{\text{Chap_per_Slot}}\}_{\text{Initiator}}$ list of slot durations supported by the initiator for
 8 the session expressed as specified in Section [19.3.1.4](#) and Table G-1. Note that
 9 initiator may prefer to use slot durations that are not necessarily the minimum (by
 10 excluding the minimum $N_{\text{Chap_per_Slot}}$ from the list) to accommodate other constraints
 11 it may have, such as co-existence.
- 12 • *SYNC_Code_Index_BitMask*: $\{\text{SYC_IDX}\}_{\text{Initiator}}$ list of available UWB preamble
 13 sequences that the responder-device may choose from.
- 14 • *Selected_UWB_Channel CH_IDX*: selected UWB RF channel.
- 15 • *Hopping_Config_Mask*: $\{H_{\text{Config_Seq}}\}_{\text{Initiator}}$ bitmask indicating the hopping
 16 configuration (no hopping, continuous, adaptive) and hopping sequences supported
 17 by the device.

- 18 3. Responder-device calculates/selects:

- 19 • *Session_RAN_Multiplier $N_{\text{RAN_S}}^k$* : the responder-device selects an N_{RAN}^k value that is
 20 greater than or equal to the value sent by the initiator to achieve a desired ranging
 21 interval of T_{Block}^k that is an integer multiple of $T_{\text{Block_Min}}$:

$$N_{\text{RAN_S}}^k = \frac{T_{\text{Block}}^k}{T_{\text{Block_Min}}} \quad \left(N_{\text{RAN_S}}^k \geq N_{\text{RAN}}^k \right) \quad \text{and} \quad f_{\text{Ranging}}^k = \frac{1}{T_{\text{Block}}^k} = \frac{10.4166667}{N_{\text{RAN_S}}^k} \text{ Hz}$$

- 22 • *Number_Chaps_per_Slot $N_{\text{Chap_per_Slot}}^k$* : shortest slot duration that is common between
 23 slot durations supported by initiator $\{N_{\text{Chap_per_Slot}}\}_{\text{Initiator}}$ and slot durations supported
 24 by the responder-device $\{N_{\text{Chap_per_Slot}}\}_{\text{Responder}}$:

$$N_{\text{Chap_per_Slot}}^k = \min \left(\{N_{\text{Chap_per_Slot}}\}_{\text{Initiator}} \square \{N_{\text{Chap_per_Slot}}\}_{\text{Responder}} \right)$$

- $N_{\text{Slot_per_Round}}^k$: responder-device selects the number of slots that is greater than or equal to $(N_{\text{Responder}}^k + 4)$ out of all possible values of slots corresponding to $N_{\text{Chap_per_Slot}}^k$ (see Table G-1 for details).
 - $\{\text{SYNC_Code_Index}\}_{\text{Responder}}$: list of UWB preamble sequences that the responder-device selected to use. This set is a subset of the list sent by the initiator.
 - $Hopping_Config_Mask$: $\{H_Config_Seq\}_{\text{Responder}}$ bitmask indicating the hopping configuration and hopping sequences selected by responder-device.
4. Responder-device uses the “Ranging Session Setup Request (RSS-RQ)” message as defined in Section [19.3.1.5](#) to send the following to the initiator:

10 a. $N_{\text{RAN_S}}^k, N_{\text{Chap_per_Slot}}^k, N_{\text{Responder}}^k, N_{\text{Slot_per_Round}}^k, \{\text{SYNC_IDX}\}_{\text{Responder}},$
 11 $\{H_Config_Seq\}_{\text{Responder}}$

- 12 5. Initiator selects the number of rounds per block for the ranging session N_{Round}^k as

$$13 N_{\text{Round}}^k = \frac{288 \times N_{\text{RAN_S}}^k}{N_{\text{Chap_per_Slot}}^k \times N_{\text{Slot_per_Round}}^k}$$

15 Appendix G.1 shows a list of all valid/possible numbers of rounds for different combinations of
 16 $(N_{\text{Chap_per_Slot}}^k, N_{\text{Slot_per_Round}}^k)$ for each 96 ms of block duration.

- 17 6. Initiator selects the preamble SYNC code index SYNC_Code_Index^k . See Section
 18 [21.4.1](#) for the definition of BPRF SYNC.
- 19 7. Initiator uses the “Ranging Session Setup - Response (RSS-RS)” message as defined in
 20 Section 19.3.1.6 to send the following back to the responder-device:
- 21 a. $STS_Index0^k, UWB_{\text{time}0}^k, HOP_Key_RW^k$
 22 b. SYNC_Code_Index^k

23 Figure 20-6 shows an example of the negotiation handshake used to set the MAC parameters.
 24 The following applies to MAC parameters negotiation:

1. $N_{\text{Chap_per_Slot}}^k = 8$ is mandatory and shall be supported by all devices.
 - This avoids a deadlock situation due to a possibly disjoint configuration parameters capabilities between the initiator and the responder-device during negotiation.
2. Responder-device will accept any N_{RAN}^k chosen by the initiator, even if it does not allow it to realize the desired ranging frequency.
 - The initiator shall set N_{RAN}^k as small as possible to meet the quality of service (QoS) constraints to meet the functional requirement.

1	<p><i>Figure 20-6: Example of MAC Parameter Negotiation and Setting.</i></p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="12"></th> <th style="text-align: right;">Tchap</th> <th style="text-align: right;">0.333 msec</th> </tr> </thead> <tbody> <tr> <td colspan="12">Step 1: RS-RQ ("Ranging Session Request") - Vehicle initiates ranging session setup</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Selected_Protocol_Version</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Selected_UWB_Config_Id</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">UWB_Session_Id</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Step 2: RS-RS ("Ranging Session Response") - Smartphone sets supported minimum block duration and Chaps per Slot</td> <td colspan="2"></td> </tr> <tr> <td colspan="12"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_RAN</th> <th>2</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th>T_Block_RAN</th> <th>192.0 msec</th> </tr> </thead> <tbody> <tr> <td>N_Chap_per_Slot</td> <td>3</td> <td>4</td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>24</td> <td></td> <td></td> <td></td> <td>f_Ranging_RAN</td> <td>5.208 Hz</td> </tr> <tr> <td>{N_Chap_per_Slot}_SP</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> <td>FALSE</td> <td>TRUE</td> <td>TRUE</td> <td colspan="3">select supported values</td> <td></td> <td></td> </tr> <tr> <td>SYNC_Code_Index_BitMask</td> <td colspan="7">></td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Channel_BitMask</td> <td colspan="7"></td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Hopping_Config_Mask</td> <td colspan="7"></td> <td colspan="3"></td> <td></td> <td></td> </tr> </tbody> </table> </td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Step 3/4: RSS-RQ ("Ranging Session Setup Request") - Vehicle selects N_RAN_S, calculates N_Chap_per_Slot, N_Slot_per_Round</td> <td colspan="2"></td> </tr> <tr> <td colspan="12"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_RAN_S</th> <th>3</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th>T_Block</th> <th>288.0 msec</th> </tr> </thead> <tbody> <tr> <td>Vehicle determines N_Chap_per_Slot</td> <td></td> <td colspan="7"></td> <td colspan="3"></td> <td>f_Ranging</td> <td>3.472 Hz</td> </tr> <tr> <td>N_Chap_per_Slot</td> <td>3</td> <td>4</td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>24</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>{N_Chap_per_Slot}_CAR</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td colspan="3">select supported values</td> <td></td> <td></td> </tr> <tr> <td>common between SP and CAR</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> <td>FALSE</td> <td>TRUE</td> <td>TRUE</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>common N_Chap_per_Slot</td> <td colspan="7"></td> <td>8</td> <td>12</td> <td>24</td> <td></td> <td></td> </tr> <tr> <td>N_Chap_per_Slot</td> <td colspan="7">8</td> <td colspan="3"></td> <td>T_Slot</td> <td>2.667 msec</td> </tr> <tr> <td colspan="12">Vehicle defines number of Responder Nodes</td> <td colspan="2"></td> </tr> <tr> <td colspan="12"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Responder</th> <th>8</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th></th> <th></th> </tr> </thead> </table> </td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Vehicle selects number of slots per round</td> <td colspan="2"></td> </tr> <tr> <td>Required number of slots</td> <td>12</td> <td colspan="7">or more</td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Available from Table A.2-1</td> <td></td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>16</td> <td>18</td> <td>24</td> <td>32</td> <td>36</td> <td>48</td> <td>72</td> <td>96</td> </tr> <tr> <td>... valid combination with N_Chap_per_Slot</td> <td></td> <td>6</td> <td></td> <td>9</td> <td>12</td> <td></td> <td>18</td> <td></td> <td>36</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>... that also fulfills required number of slots</td> <td></td> <td></td> <td></td> <td>12</td> <td></td> <td>18</td> <td></td> <td>36</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>N_Slot_per_Round</td> <td colspan="7">12</td> <td colspan="3"></td> <td>T_Round</td> <td>32.0 msec</td> </tr> <tr> <td colspan="12">/{SYC_IDX}_CAR</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">CH_IDX</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">{H_Config_Seq}_CAR</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Step 5: Initiator (Smartphone) calculates the number of rounds per block</td> <td colspan="2"></td> </tr> <tr> <td colspan="12"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Round</th> <th>9</th> <th colspan="7"></th> <th>Rounds per block</th> <th>9</th> </tr> </thead> </table> </td> <td colspan="2"></td> </tr> <tr> <td colspan="12"></td> <td>check</td> <td>T_Block</td> <td>288.0 msec</td> </tr> </tbody> </table> </td> <td style="vertical-align: top; width: 10%;">2</td></tr> </tbody></table>													Tchap	0.333 msec	Step 1: RS-RQ ("Ranging Session Request") - Vehicle initiates ranging session setup														Selected_Protocol_Version														Selected_UWB_Config_Id														UWB_Session_Id														Step 2: RS-RS ("Ranging Session Response") - Smartphone sets supported minimum block duration and Chaps per Slot														<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_RAN</th> <th>2</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th>T_Block_RAN</th> <th>192.0 msec</th> </tr> </thead> <tbody> <tr> <td>N_Chap_per_Slot</td> <td>3</td> <td>4</td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>24</td> <td></td> <td></td> <td></td> <td>f_Ranging_RAN</td> <td>5.208 Hz</td> </tr> <tr> <td>{N_Chap_per_Slot}_SP</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> <td>FALSE</td> <td>TRUE</td> <td>TRUE</td> <td colspan="3">select supported values</td> <td></td> <td></td> </tr> <tr> <td>SYNC_Code_Index_BitMask</td> <td colspan="7">></td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Channel_BitMask</td> <td colspan="7"></td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Hopping_Config_Mask</td> <td colspan="7"></td> <td colspan="3"></td> <td></td> <td></td> </tr> </tbody> </table>												N_RAN	2	Input value										T_Block_RAN	192.0 msec	N_Chap_per_Slot	3	4	6	8	9	12	24				f_Ranging_RAN	5.208 Hz	{N_Chap_per_Slot}_SP	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	select supported values					SYNC_Code_Index_BitMask	>												Channel_BitMask													Hopping_Config_Mask															Step 3/4: RSS-RQ ("Ranging Session Setup Request") - Vehicle selects N_RAN_S, calculates N_Chap_per_Slot, N_Slot_per_Round														<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_RAN_S</th> <th>3</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th>T_Block</th> <th>288.0 msec</th> </tr> </thead> <tbody> <tr> <td>Vehicle determines N_Chap_per_Slot</td> <td></td> <td colspan="7"></td> <td colspan="3"></td> <td>f_Ranging</td> <td>3.472 Hz</td> </tr> <tr> <td>N_Chap_per_Slot</td> <td>3</td> <td>4</td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>24</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>{N_Chap_per_Slot}_CAR</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td colspan="3">select supported values</td> <td></td> <td></td> </tr> <tr> <td>common between SP and CAR</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> <td>FALSE</td> <td>TRUE</td> <td>TRUE</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>common N_Chap_per_Slot</td> <td colspan="7"></td> <td>8</td> <td>12</td> <td>24</td> <td></td> <td></td> </tr> <tr> <td>N_Chap_per_Slot</td> <td colspan="7">8</td> <td colspan="3"></td> <td>T_Slot</td> <td>2.667 msec</td> </tr> <tr> <td colspan="12">Vehicle defines number of Responder Nodes</td> <td colspan="2"></td> </tr> <tr> <td colspan="12"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Responder</th> <th>8</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th></th> <th></th> </tr> </thead> </table> </td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Vehicle selects number of slots per round</td> <td colspan="2"></td> </tr> <tr> <td>Required number of slots</td> <td>12</td> <td colspan="7">or more</td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Available from Table A.2-1</td> <td></td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>16</td> <td>18</td> <td>24</td> <td>32</td> <td>36</td> <td>48</td> <td>72</td> <td>96</td> </tr> <tr> <td>... valid combination with N_Chap_per_Slot</td> <td></td> <td>6</td> <td></td> <td>9</td> <td>12</td> <td></td> <td>18</td> <td></td> <td>36</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>... that also fulfills required number of slots</td> <td></td> <td></td> <td></td> <td>12</td> <td></td> <td>18</td> <td></td> <td>36</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>N_Slot_per_Round</td> <td colspan="7">12</td> <td colspan="3"></td> <td>T_Round</td> <td>32.0 msec</td> </tr> <tr> <td colspan="12">/{SYC_IDX}_CAR</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">CH_IDX</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">{H_Config_Seq}_CAR</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Step 5: Initiator (Smartphone) calculates the number of rounds per block</td> <td colspan="2"></td> </tr> <tr> <td colspan="12"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Round</th> <th>9</th> <th colspan="7"></th> <th>Rounds per block</th> <th>9</th> </tr> </thead> </table> </td> <td colspan="2"></td> </tr> <tr> <td colspan="12"></td> <td>check</td> <td>T_Block</td> <td>288.0 msec</td> </tr> </tbody> </table>												N_RAN_S	3	Input value										T_Block	288.0 msec	Vehicle determines N_Chap_per_Slot												f_Ranging	3.472 Hz	N_Chap_per_Slot	3	4	6	8	9	12	24						{N_Chap_per_Slot}_CAR	TRUE	select supported values					common between SP and CAR	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE						common N_Chap_per_Slot								8	12	24			N_Chap_per_Slot	8										T_Slot	2.667 msec	Vehicle defines number of Responder Nodes														<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Responder</th> <th>8</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th></th> <th></th> </tr> </thead> </table>												N_Responder	8	Input value														Vehicle selects number of slots per round														Required number of slots	12	or more												Available from Table A.2-1		6	8	9	12	16	18	24	32	36	48	72	96	... valid combination with N_Chap_per_Slot		6		9	12		18		36					... that also fulfills required number of slots				12		18		36						N_Slot_per_Round	12										T_Round	32.0 msec	/{SYC_IDX}_CAR														CH_IDX														{H_Config_Seq}_CAR														Step 5: Initiator (Smartphone) calculates the number of rounds per block														<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Round</th> <th>9</th> <th colspan="7"></th> <th>Rounds per block</th> <th>9</th> </tr> </thead> </table>												N_Round	9								Rounds per block	9															check	T_Block	288.0 msec	2						
												Tchap	0.333 msec																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
Step 1: RS-RQ ("Ranging Session Request") - Vehicle initiates ranging session setup																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
Selected_Protocol_Version																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
Selected_UWB_Config_Id																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
UWB_Session_Id																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
Step 2: RS-RS ("Ranging Session Response") - Smartphone sets supported minimum block duration and Chaps per Slot																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_RAN</th> <th>2</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th>T_Block_RAN</th> <th>192.0 msec</th> </tr> </thead> <tbody> <tr> <td>N_Chap_per_Slot</td> <td>3</td> <td>4</td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>24</td> <td></td> <td></td> <td></td> <td>f_Ranging_RAN</td> <td>5.208 Hz</td> </tr> <tr> <td>{N_Chap_per_Slot}_SP</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> <td>FALSE</td> <td>TRUE</td> <td>TRUE</td> <td colspan="3">select supported values</td> <td></td> <td></td> </tr> <tr> <td>SYNC_Code_Index_BitMask</td> <td colspan="7">></td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Channel_BitMask</td> <td colspan="7"></td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Hopping_Config_Mask</td> <td colspan="7"></td> <td colspan="3"></td> <td></td> <td></td> </tr> </tbody> </table>												N_RAN	2	Input value										T_Block_RAN	192.0 msec	N_Chap_per_Slot	3	4	6	8	9	12	24				f_Ranging_RAN	5.208 Hz	{N_Chap_per_Slot}_SP	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	select supported values					SYNC_Code_Index_BitMask	>												Channel_BitMask													Hopping_Config_Mask																																																																																																																																																																																																																																																																																																																																																																																																																																																									
N_RAN	2	Input value										T_Block_RAN	192.0 msec																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
N_Chap_per_Slot	3	4	6	8	9	12	24				f_Ranging_RAN	5.208 Hz																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
{N_Chap_per_Slot}_SP	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	select supported values																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
SYNC_Code_Index_BitMask	>																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																						
Channel_BitMask																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
Hopping_Config_Mask																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
Step 3/4: RSS-RQ ("Ranging Session Setup Request") - Vehicle selects N_RAN_S, calculates N_Chap_per_Slot, N_Slot_per_Round																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_RAN_S</th> <th>3</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th>T_Block</th> <th>288.0 msec</th> </tr> </thead> <tbody> <tr> <td>Vehicle determines N_Chap_per_Slot</td> <td></td> <td colspan="7"></td> <td colspan="3"></td> <td>f_Ranging</td> <td>3.472 Hz</td> </tr> <tr> <td>N_Chap_per_Slot</td> <td>3</td> <td>4</td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>24</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>{N_Chap_per_Slot}_CAR</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td>TRUE</td> <td colspan="3">select supported values</td> <td></td> <td></td> </tr> <tr> <td>common between SP and CAR</td> <td>FALSE</td> <td>FALSE</td> <td>FALSE</td> <td>TRUE</td> <td>FALSE</td> <td>TRUE</td> <td>TRUE</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>common N_Chap_per_Slot</td> <td colspan="7"></td> <td>8</td> <td>12</td> <td>24</td> <td></td> <td></td> </tr> <tr> <td>N_Chap_per_Slot</td> <td colspan="7">8</td> <td colspan="3"></td> <td>T_Slot</td> <td>2.667 msec</td> </tr> <tr> <td colspan="12">Vehicle defines number of Responder Nodes</td> <td colspan="2"></td> </tr> <tr> <td colspan="12"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Responder</th> <th>8</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th></th> <th></th> </tr> </thead> </table> </td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Vehicle selects number of slots per round</td> <td colspan="2"></td> </tr> <tr> <td>Required number of slots</td> <td>12</td> <td colspan="7">or more</td> <td colspan="3"></td> <td></td> <td></td> </tr> <tr> <td>Available from Table A.2-1</td> <td></td> <td>6</td> <td>8</td> <td>9</td> <td>12</td> <td>16</td> <td>18</td> <td>24</td> <td>32</td> <td>36</td> <td>48</td> <td>72</td> <td>96</td> </tr> <tr> <td>... valid combination with N_Chap_per_Slot</td> <td></td> <td>6</td> <td></td> <td>9</td> <td>12</td> <td></td> <td>18</td> <td></td> <td>36</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>... that also fulfills required number of slots</td> <td></td> <td></td> <td></td> <td>12</td> <td></td> <td>18</td> <td></td> <td>36</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>N_Slot_per_Round</td> <td colspan="7">12</td> <td colspan="3"></td> <td>T_Round</td> <td>32.0 msec</td> </tr> <tr> <td colspan="12">/{SYC_IDX}_CAR</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">CH_IDX</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">{H_Config_Seq}_CAR</td> <td colspan="2"></td> </tr> <tr> <td colspan="12">Step 5: Initiator (Smartphone) calculates the number of rounds per block</td> <td colspan="2"></td> </tr> <tr> <td colspan="12"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Round</th> <th>9</th> <th colspan="7"></th> <th>Rounds per block</th> <th>9</th> </tr> </thead> </table> </td> <td colspan="2"></td> </tr> <tr> <td colspan="12"></td> <td>check</td> <td>T_Block</td> <td>288.0 msec</td> </tr> </tbody> </table>												N_RAN_S	3	Input value										T_Block	288.0 msec	Vehicle determines N_Chap_per_Slot												f_Ranging	3.472 Hz	N_Chap_per_Slot	3	4	6	8	9	12	24						{N_Chap_per_Slot}_CAR	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	select supported values					common between SP and CAR	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE						common N_Chap_per_Slot								8	12	24			N_Chap_per_Slot	8										T_Slot	2.667 msec	Vehicle defines number of Responder Nodes														<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Responder</th> <th>8</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th></th> <th></th> </tr> </thead> </table>												N_Responder	8	Input value														Vehicle selects number of slots per round														Required number of slots	12	or more												Available from Table A.2-1		6	8	9	12	16	18	24	32	36	48	72	96	... valid combination with N_Chap_per_Slot		6		9	12		18		36					... that also fulfills required number of slots				12		18		36						N_Slot_per_Round	12										T_Round	32.0 msec	/{SYC_IDX}_CAR														CH_IDX														{H_Config_Seq}_CAR														Step 5: Initiator (Smartphone) calculates the number of rounds per block														<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Round</th> <th>9</th> <th colspan="7"></th> <th>Rounds per block</th> <th>9</th> </tr> </thead> </table>												N_Round	9								Rounds per block	9															check	T_Block	288.0 msec	2																																																																																																																																																																																																	
N_RAN_S	3	Input value										T_Block	288.0 msec																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
Vehicle determines N_Chap_per_Slot												f_Ranging	3.472 Hz																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
N_Chap_per_Slot	3	4	6	8	9	12	24																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
{N_Chap_per_Slot}_CAR	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	select supported values																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
common between SP and CAR	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																
common N_Chap_per_Slot								8	12	24																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
N_Chap_per_Slot	8										T_Slot	2.667 msec																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
Vehicle defines number of Responder Nodes																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Responder</th> <th>8</th> <th colspan="7">Input value</th> <th colspan="3"></th> <th></th> <th></th> </tr> </thead> </table>												N_Responder	8	Input value																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
N_Responder	8	Input value																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
Vehicle selects number of slots per round																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
Required number of slots	12	or more																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
Available from Table A.2-1		6	8	9	12	16	18	24	32	36	48	72	96																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
... valid combination with N_Chap_per_Slot		6		9	12		18		36																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																														
... that also fulfills required number of slots				12		18		36																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
N_Slot_per_Round	12										T_Round	32.0 msec																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											
/{SYC_IDX}_CAR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
CH_IDX																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
{H_Config_Seq}_CAR																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
Step 5: Initiator (Smartphone) calculates the number of rounds per block																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>N_Round</th> <th>9</th> <th colspan="7"></th> <th>Rounds per block</th> <th>9</th> </tr> </thead> </table>												N_Round	9								Rounds per block	9																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
N_Round	9								Rounds per block	9																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
												check	T_Block	288.0 msec																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									

3 20.5.3 MAC Control Channel (Over Bluetooth LE)

4 A MAC control channel shall be implemented with following requirements:

- 5 1. Control channel shall be available for negotiation or re-negotiation of UWB parameters (e.g. ranging session setup as in Section [20.5.2](#)).
- 6 2. The control channel shall be session specific and shall be available before the start of, during the entire lifetime of, and after the termination of the ranging session.
- 7 3. A RAN will have as many UWB control channels as it has responder-devices.
- 8 4. The control channel shall be carried over the Bluetooth LE connection.
- 9 5. Control traffic with respect to the k -th ranging session shall be referenced with the *UWB_Session_ID^k*.
- 10 6. Timings during an active ranging session shall be referenced via the appropriate MAC time grid indices (e.g. block index, round index, slot index, or mapped STS indices) of this ranging session.
- 11 7. The control channel shall be triggered by a channel ID in the Bluetooth LE L2CAP channel. Initiator or responder-device may trigger the control channel and can request a change of UWB parameters.

20.5.4 UWB MAC Configuration

All SP0 type packets sent according to the above MAC protocol shall include a MAC header (MHR), MAC payload and MAC Footer (MFR) fields. This section describes how the contents of the SP0 packets are constructed.

Figure 20-7: SP0 Packet Content.

MHR (23 Octets)	MAC Payload Frame Payload (variable) + MIC (8 octects)	MFR (2 Octets)
-----------------	---	----------------

20.5.4.1 MHR Field

The contents of the MHR field according to Section 7.2 of [31] are defined in Table 20-8. All values of the parameters listed in Table 20-8 shall be encoded in little endian format.

Table 20-8: The Contents of the MHR Field.

MAC parameter	Length (bytes)	Description
Frame Control	2	Describes how the MAC frame is configured
Sequence Number	0	Sequence number is not used. Sequence Number Suppression bit field in the Frame Control shall be set to 1
Destination PAN ID	0	Destination PAN ID not present
Destination Address	2	Destination Short Address
Source PAN ID	0	Source PAN ID not present
Source Address	0	Source Address not present
Auxiliary Security Header	10	Describes security configuration for data payload encryption and message identification authentication (MIC)
Vendor Specific Header IE	7	Used to signal the UWB message ID and the UWB message length
Header Termination IE	2	Header Termination IE of Type HT2. See 7.4.1 of [31]

The Frame Control field depends on the specific MAC header configuration. See [31] for more information about the individual parameters and their definitions. The Frame Control field is defined in Table 20-9.

Table 20-9: The Contents of the Frame Control Field.

Frame Control	Length (bits)	Value	Description
Frame Type	3	0b001 (data)	Data frame being transmitted
Security Enabled	1	0b1	Every packet has security enabled
Frame Pending	1	0b0	Frame Pending Flag is always set to 0.
AR	1	0b0	This flag indicates whether this frame requires acknowledgment (0b1) or not (0b0). Always set to 0 as neither Pre-Poll or Final Data packets require an acknowledgment back from the vehicle
PAN ID Compression	1	0b1	Source Addressing Mode set to 0, Destination Addressing Mode set to 2 (short address), and Destination PAN ID not present this corresponds to the 4th entry of Table 7-2 in [31].

Frame Control	Length (bits)	Value	Description
Reserved	1	0b0	
Sequence Number Suppression	1	0b1	Sequence number is disabled
IE Present	1	0b1	Header IE Used.
Destination Addressing Mode	2	0b10	Destination Short Address
Frame Version	2	0b10	802.15.4 frames for data and acknowledgement.
Source Addressing Mode	2	0b0	Source PAN ID and Source Address field are not present

1 The Auxiliary Security Header defines the security configuration. The Auxiliary Security Header
2 is defined in Table 20-10. Note that the Auxiliary header is not relevant to SP3 packets (Poll and
3 Final messages) since SP3 packets do not get security processing due to the fact they do not
4 carry MAC data frames (payload, MHR, and MFR). Therefore, The Auxiliary Security header is
5 applicable to SP0 packets only (Pre-Poll and Final_Data). All values of the parameters listed in
6 Table 20-10 shall be encoded in little endian format.

7 *Table 20-10: The Contents of the Auxiliary Security Header.*

Auxiliary Security Header	Length (bytes)	Value	Description
Security Control Field	See below		
Security Level	3 bits	0b110 (ENC-MIC-64)	Security level for authentication of MAC header and data payload
Key Identifier Mode	2 bits	0b10, Key is determined explicitly from the 4-byte Key Source field and Key Identifier field	Key identification mode
Frame Counter Suppression	1 bit	0b0, enable frame counter	Frame counter is suppressed or not
ASN in Nonce	1 bit	0b0, ASN in not in nonce	Is ASN in nonce or not
Reserved	1 bit	0b0	
Frame Counter	4 bytes	0x00000000-0xFFFFFFFF	Counter that increments for each transmitted packet from a source
Key Identifier Field	See below		
Key Source	4 bytes	0x00000000-0xFFFFFFFF	Key Source derived from UAD
Key Index	1 byte	0xAA	

- 8 The content of the Header IE defined the Vendor Specific Header IE which shall be used to
9 signal the UWB message ID and the UWB message length carried by the corresponding UWB
10 packet.
11 The Vendor Specific Header IE is defined in Table 20-11. All parameter values listed in Table
12 20-11 shall be encoded in little endian format.

1

Table 20-11: The Contents of the Vendor Specific Header IE.

Vendor Specific Header IE	Length (bytes)	Value	Description
Length	7 bits	5 (bit 0 - bit 6)	Length of the IE content in bytes
IE ID	8 bits	0 (bit 7 - bit 14)	Vendor Specific Header IE
Type	1 bit	0 (bit 15)	Header IE
Vendor OUI	3 bytes	0x04DF69	OUI of Car Connectivity Consortium per IEEE registration authority (See http://standards-oui.ieee.org/oui.txt).
Vendor Specific Information	2 Bytes	Byte 1: 0x00 - 0xFF Byte 2: 0x00 - 0xFF	Byte 1: UWB MAC message ID of message sent in this UWB packet 0x01: Pre-Poll UWB MAC message ID 0x02: Final_Data UWB MAC message ID 0x03-0xFF: Reserved for future use Byte 2: UWB MAC message payload length in bytes sent in this UWB packet excluding the MHR, MFR, and MIC fields.

2 20.5.4.2 MHR Source and Destination Address

3 The Source Addressing Mode shall be set to 0 in the Frame Control Field of the MHR. This
4 indicates that the Source Address and the Source PAN ID shall not be present in the MHR. The
5 Destination Addressing Mode shall be set to 2 and the PAN ID Compression shall all be set to 1
6 in the Frame Control field of the MHR. This indicates that the Destination Short address shall be
7 present, and the Destination PAN ID shall not be present in the MHR. The Destination Short
8 Address along with Source Long Address and Key Source fields shall be generated according to
9 address rotation procedure -as described in Section [22](#).

10 20.5.4.3 MAC Payload

11 The MAC payload field is of variable length and includes the MAC frame payload plus the 64-bit
12 message integrity check (MIC) field

13 20.5.4.4 MFR Field

14 MAC footer shall be according to Section 7.2.10 of [31], which states that the MAC footer shall
15 contain a Frame Check Sequence (FCS) field. For the BPRF mode in the HRP UWB PHY, the
16 FCS contains a 16-bit ITU-T CRC. The FCS is calculated over the MHR and MAC payload part
17 of the frame; these parts together are referred to as the calculation fields. This 2-byte FCS is
18 calculated using the following standard generator polynomial of degree 16:

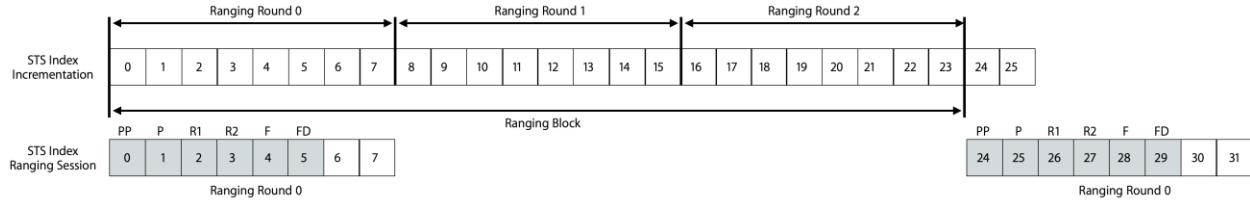
$$G_{16}(x) = x^{16} + x^{12} + x^5 + 1$$

20 20.6 STS Index Incrementation

21 The STS index parameter (which needs to change for every slot in ranging block regardless of
22 whether that slot is used or not) plays an important role for the secure ranging operation. This
23 section defines the STS index incrementation and the mapping of the STS index onto the packets

1 of a ranging session within the MAC layer framework. The basic principle of the STS
2 incrementation is shown in Figure 20-8.

3 *Figure 20-8: Basic Principles of STS Incrementation.*



4
5 (The example uses $N_{\text{Responder}}^k = 2$, $N_{\text{Slot_per_Round}}^k = 8$, $N_{\text{Round}}^k = 3$. Note that $N_{\text{Round}}^k = 3$ is not a
6 supported number of rounds in the current design but is used in this example for illustration
7 purpose only.)

8 20.6.1 Rules for STS Index Incrementation

9 The following rules shall be used for incrementing the STS index:

10 1. The STS index is managed separately for each ranging session of a given RAN.

11 2. A k -th ranging session starts at time UWB_{time0}^k .

12 3. The STS index of the k -th ranging session is referenced to the start of this ranging
13 session:

14 STS_Index0^k is mapped to ranging round 0 of ranging block 0. That is, for $i = 0$ and
15 $Round_Idx^k(i = 0) = 0$

$$16 \quad STS_Index^k \left(i = 0, Round_Idx^k(i = 0) = 0 \right) = STS_Index0^k$$

17 4. Within a ranging round, the STS index increments in every slot even if that slot is
18 unmapped (i.e. not used).

19 5. The incrementation scheme is applied for every ranging round (even if it is not used).
20 This results in:

21 a. The STS index is incremented by $N_{\text{Slot_per_Round}}^k$ in every ranging round.

22 b. The STS index is incremented by $N_{\text{Slot_per_Round}}^k \cdot N_{\text{Round}}^k$ in every ranging block.

23 6. The STS index is incremented for all slots in the round. However, data packets (Pre-
24 POLL and Final_Data) reference the STS index in their payload as follows:

25 a. Pre-POLL message shall carry the reference to the STS index that is used in the
26 following POLL message.

27 b. Final_Data message shall refer to the STS index of the preceding FINAL
28 message.

29 20.6.2 STS Incrementation and Packet Mapping within a Ranging Round

30 Within any ranging round s in ranging block i , the STS index increments for every slot (relative
31 to the first STS index in the round):

- 1 • Pre_POLL $STS_Index^k(i,s)$
- 2 • POLL $STS_Index^k(i,s) + 1$
- 3 • Response_0 $STS_Index^k(i,s) + 2$
- 4 • Response_1 $STS_Index^k(i,s) + 3$
- 5 • ...
- 6 • Response_{{N_{\text{Responder}}^k} - 1} $STS_Index^k(i,s) + N_{\text{Responder}}^k + 1$
- 7 • Final $STS_Index^k(i,s) + N_{\text{Responder}}^k + 2$
- 8 • Final_Data $STS_Index^k(i,s) + N_{\text{Responder}}^k + 3$

9 20.6.3 Calculation of STS Index

10 The following formulas are used to calculate the STS indices for ranging round $Round_Idx^k(i)$
11 in ranging block i :

$$\begin{aligned}
 STS_Index^k(i, Round_Idx^k(i), \text{Pre_POLL}) &= STS_Index0^k + (i \times N_{\text{Round}}^k + Round_Idx^k(i)) \times N_{\text{Slot_per_Round}}^k \\
 STS_Index^k(i, Round_Idx^k(i), \text{POLL}) &= STS_Index0^k + (i \times N_{\text{Round}}^k + Round_Idx^k(i)) \times N_{\text{Slot_per_Round}}^k + 1 \\
 STS_Index^k(i, Round_Idx^k(i), R_0) &= STS_Index0^k + (i \times N_{\text{Round}}^k + Round_Idx^k(i)) \times N_{\text{Slot_per_Round}}^k + 2 \\
 STS_Index^k(i, Round_Idx^k(i), R_1) &= STS_Index0^k + (i \times N_{\text{Round}}^k + Round_Idx^k(i)) \times N_{\text{Slot_per_Round}}^k + 3 \\
 &\vdots \\
 STS_Index^k(i, Round_Idx^k(i), R_{N_{\text{Responder}}^k - 1}) &= STS_Index0^k + (i \times N_{\text{Round}}^k + Round_Idx^k(i)) \times N_{\text{Slot_per_Round}}^k + N_{\text{Responder}}^k + 1 \\
 STS_Index^k(i, Round_Idx^k(i), \text{Final}) &= STS_Index0^k + (i \times N_{\text{Round}}^k + Round_Idx^k(i)) \times N_{\text{Slot_per_Round}}^k + N_{\text{Responder}}^k + 2 \\
 STS_Index^k(i, Round_Idx^k(i), \text{Final_Data}) &= STS_Index0^k + (i \times N_{\text{Round}}^k + Round_Idx^k(i)) \times N_{\text{Slot_per_Round}}^k + N_{\text{Responder}}^k + 3
 \end{aligned}$$

1 **21 UWB PHY [WCC3]**

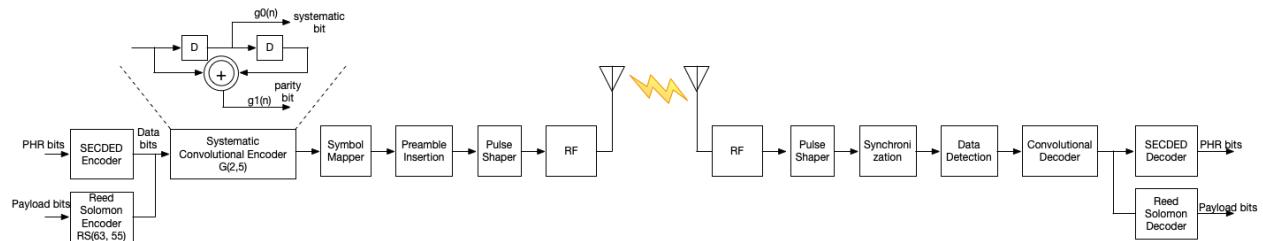
2 **21.1 Introduction**

3 The UWB PHY uses a waveform based on an impulse radio signal using band-limited pulses.
4 UWB PHY is primarily used for ranging but may also be used for data communication. UWB
5 PHY described in this specification is based on HRP UWB PHY in [33].
6 This section describes interoperable enhanced ranging devices (ERDEVs) featuring enhanced
7 immunity to attack. The main enhancement for secure time of flight is the inclusion of a
8 Scrambled Timestamp Sequence (STS) in the basic PPDU format. Note that the STS is included
9 in the basic PPDU format in the BPRF mode as specified in [33].
10 The IEEE standard defines a very flexible UWB PHY. The flexibility in IEEE standard is offered
11 by adaptation of parameters like preamble for synchronization (SYNC) lengths, preamble codes,
12 Start of Frame Delimiter (SFD) lengths/codes, pulse repetition frequencies (PRFs), and data
13 rates. This specification, however, does not require to implement all the parameters and modes
14 that are specified in [33]. Please refer to appropriate sections on mandatory and optional PHY
15 modes for secure ranging, ERDEVs on both sides of the link shall pre-negotiate the specific
16 parameters that will be used for a secure ranging session. The negotiations of secure ranging
17 parameters can be done at higher layers or using Bluetooth LE.

18 **21.2 UWB PHY Block Diagram**

19 Figure 21-1 shows the block diagram for IEEE 802.15.4z HRP UWB PHY.

20 *Figure 21-1: Block diagram for IEEE 802.15.4z HRP UWB PHY.*



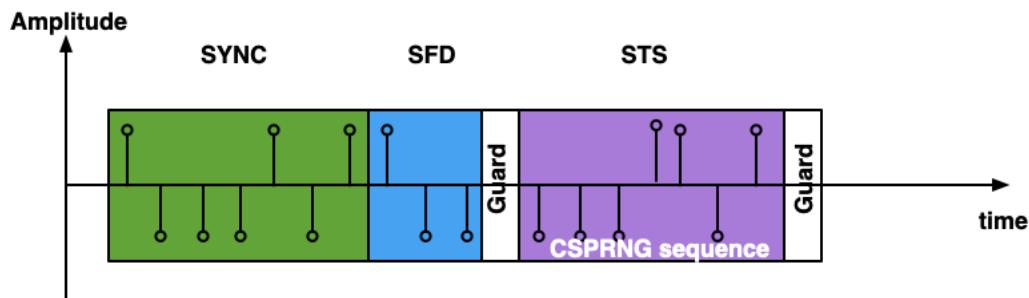
22 **21.3 UWB Packet Format**

23 Figure 21-2 and Figure 21-3 illustrate the two, packet types for UWB configurations: STS packet
24 type 3 (SP3) and STS packet type 0 (SP0).

25 For SP3, the Deterministic Random Bit Generator (DRBG) STS engine shall generate 4096 bits
26 to create the STS waveform and there is no PHR or data field. Note that packet configurations to
27 be used for UWB ranging (and also for UWB data communication) shall be known a-priori. The
28 packet configurations to be used between two nodes shall be exchanged over Bluetooth LE link.
29 However, the length of the data field in SP0 is flexible.

1

Figure 21-2: SP3



2

3

Figure 21-3: SP0



4

- 5 SP3 is intended for use cases where the participants in the secure ranging exchange are known to
6 each other such that information about the source and/or the destination are implicit in the
7 knowledge of what STS is used for transmission and reception between the connected devices,
8 respectively.
- 9 A nominal mean PRF of at least 62.4 MHz shall be used for all packets during a ranging session.
10 Nominal mean PRFs lower than 62.4 MHz are not required for ERDEVs.

11 21.4 UWB Frame Elements

- 12 Only IEEE 802.15.4z HRP BPRF UWB SP0 and SP3 packets are used in this specification. SP0
13 packet consists of preamble (SYNC), SFD, PHY header (PHR), and payload (PSDU or Data
14 field). Supported configurations are referenced as UWB_Config as shown in Table 21-1. Each of
15 the fields is described in detail in the subsections below.

1

Table 21-1: Supported UWB Configurations.

DK protocol			M/O	STS Packet Type 3 (SP3) IEEE 802.15.4z parameters				STS Packet Type 0 (SP0) IEEE 802.15.4z parameters							
UWB configuration	UWB Channel (IEEE 802.15.4)	SYNC preamble code (IEEE 802.15.4)		Mandatory/ Optional	corresponding IEEE BPRF Mandatory Set	phyHrpUwbPsr	SFD (phyHrpUwbSfd- Selector)	STS (segment x symbols)	corresponding IEEE802.15.4z BPRF Set	phyHrpUwbPsr	SFD (phyHrpUwbSfd- Selector)	STS	phyHrpUwbPhr- DataRate	PHR bit rate	PSDU bit rate
0	5, 9	9,10,11,12 Optional: 13-16, 21-24	M	N/A	64	ternary (0)	1× 64	BPRF #1	64	ternary (0)	no	DRBM-LP	0.85 Mbps	6.81 Mbps	
1	5, 9	9,10,11,12 Optional: 13-16, 21-24	Device: M Vehicle: O	BPRF #4	64	binary (2)	1× 64	BPRF #2	64	binary (2)	no	DRBM-LP	0.85 Mbps	6.81 Mbps	

- 2 21.4.1 BPRF SYNC
- 3 SYNC is used to aid receiver algorithms related to AGC settling, timing acquisition, coarse and fine frequency recovery, packet and frame synchronization. [31] defines four mandatory
- 4 preamble lengths for SYNC: a default preamble, a short preamble, a medium preamble, and a
- 5 long preamble. The length of default preamble is 64 symbols, short preamble is 16 symbols,
- 6 medium preamble is 1024 symbols, and long preamble is 4096 symbols. The PHY as specified in
- 7 this specification shall support 64 symbols short preamble.
- 8 ERDEV in this specification, shall support ternary code with length 127, even though, ternary
- 9 code with length 127 is optional in [31]. Each preamble code is a sequence of code symbols
- 10 drawn from a ternary alphabet {−1,0,1} and is selected for use because of their perfect periodic
- 11 autocorrelation properties.
- 12 In this specification, the ERDEV shall operate on channels 5 and 9, and hence the support for the
- 13 four length-127 ternary code sequences shown in Table 21-2 are mandatory, though they are
- 14 optional in [31].

16 Table 21-2: The Mandatory Length-127 Ternary Code Sequences.

Code index	Ternary Code Sequence	UWB channels
9	+00+000-0-00--+0+0+00-++-+0+0000+-000+00-00--0-+0+0-0-++0++000+- 0+00-0++-0++00+00+0+0-0++-+000000+00000-+0000-0-000--	5,9
10	++00+0-00+00+000000-000-00-000-0++-0-0-+00000-+00++0-0+00--+00++- +0-0+0000-0-0-0-++-0+00+0+000-+0++000---++0000++0--	5,9
11	--0000+00--00000-0+0+0-0+00+00+0-00-++00+0000-+0+0-0000+++++-0++- 0+-0++-0-000+0-00+0----000-000000-+00+-0++000++-0++-0-0	5,9
12	-+0++000000-0+0-0-++-+00-+0++0+0+0+000-00-00-+00++-+0000-+0-++-0- 0++++0-00-0++0+0+00++-00+0000-000-0--0000-0000-0+00000-+	5,9

- 17 The Code index 9, 10, 11, 12 shall be supported.

1 Support for code indexes 13-16 and 21-24 ternary code sequences as specified in [31] is optional
2 in this specification.

3

4 **21.4.2 BPRF SFD**

5 SFD is added to the UWB frame to establish frame timing at the receiver. Only length 8 SFD is
6 used in this specification:

7 The short ternary SFD shall be $[0 +1 0 -1 +1 0 0 -1]$ spread by the preamble symbol S_i , where
8 the leftmost bit shall be transmitted first in time. This is as specified in [31]. To improve
9 performance, the binary sequence: $[-1 -1 -1 +1 -1 -1 +1 -1]$ should be supported per [33].

10 **21.4.3 Scrambled Timestamp Sequence (STS)**

11 UWB PHY supports Secure Ranging mechanisms to prevent different forms of attacks carried
12 out by a “rogue” device injecting signal energies with the objective of making ranging receivers
13 misread the distance between associated ranging peers. Scrambled Timestamp Sequence (STS)
14 provides an AES-generated ranging waveform to protect against range measurement spoofing.
15 DRBG based STS is resilient to various types of distance manipulation attacks like Brute force
16 attack, Cicada attack, preamble injection attack and preamble EDLC. Details on how STS
17 sequence is generated is specified in [33]. The number of bits used for STS is specified in Table
18 21-1. STS bits are mapped to pulses as follows:

- 19 1. Bit 1 is mapped to a negative pulse
20 2. Bit 0 is mapped to a positive pulse

21 In the BPRF (64 MHz PRF) mode, there is one pulse every 8 chips (spreading = 8)

22 **21.4.4 BPRF PHR**

23 A PHR, as shown in Figure 21-4, shall be added after the SHR preamble. The PHR consists of
24 19 bits and conveys information necessary for a successful decoding of the packet to the
25 receiver. The PHR contains information about the data rate used to transmit the PSDU, the
26 duration of the current frame’s preamble, and the length of the frame payload. Additionally, six
27 parity check bits are used to further protect the PHR against propagation channel errors.

28 *Figure 21-4: PHR bit assignment.*

Bits: 0–1	2–8	9	10	11–12	13–18
Data Rate	Frame Length	Ranging	Reserved	Preamble Duration	SECDED

29
30 It is mandatory for RDEVs and ERDEVs to support transmission and reception of PRF64 PHRs
31 at 850 Kbit/s data rate. The BPRF PHR shall use Single Error Correct Dual Error Detect
32 (SECDED) encoding followed by systematic convolutional encoder with generator polynomial
33 $G(2,5)$ as shown in Figure 21-1. The SECDED field is a set of six parity check bits that is used to
34 protect the PHR from errors caused by noise and channel impairments. The SECDED bits are a
35 simple Hamming block code that enables the correction of a single error and the detection of two

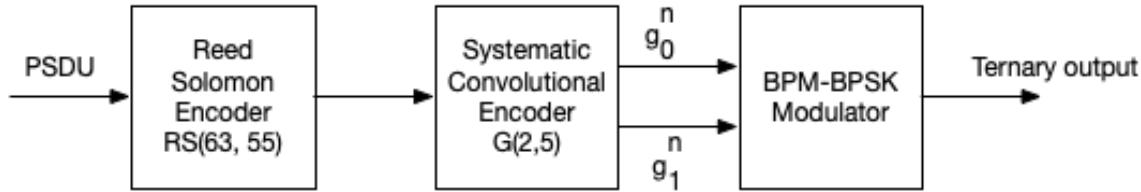
1 errors at the receiver. The SECDED bit values depend on PHR bits 0–12 and are computed as
2 follows:

$$\begin{aligned} b_{18} &= \text{XOR}(b_1, b_0, b_8, b_6, b_4, b_3, b_{10}, b_{11}) \\ b_{17} &= \text{XOR}(b_0, b_6, b_5, b_3, b_2, b_9, b_{10}, b_{12}) \\ b_{16} &= \text{XOR}(b_1, b_8, b_7, b_3, b_2, b_9, b_{10}) \\ b_{15} &= \text{XOR}(b_8, b_7, b_6, b_5, b_4, b_9, b_{10}) \\ b_{14} &= \text{XOR}(b_{12}, b_{11}) \\ b_{13} &= \text{XOR}(b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}, b_{12}, b_{14}, b_{15}, b_{16}, b_{17}, b_{18}) \end{aligned}$$

9 21.4.5 Standard Compliant Data Field

10 The Data field is transmitted using the 6.8 Mbps data rate at a mean PRF of 62.4 MHz and
11 encoded as shown in Figure 21-5.

12 *Figure 21-5: Data Field Encoding for Standard PHY.*



13 The data field shall be formed as follows:

- 14 • Encode the PSDU using systematic Reed-Solomon block code, which adds 48 parity bits
- 15 • Encode the output of the Reed-Solomon block code using a systematic convolutional
- 16 • encoder. If the Viterbi rate for the modulation is defined as one, then the convolutional
- 17 • encoder is bypassed
- 18 • Spread and modulate the encoded block using BPM-BPSK modulation

20 21.5 Pulse Shape Combinations

21 In addition to the pulse shape requirements in Section 15.4.4 and 15.4.5 of [33], a compliant
22 transmitter shall choose between a traditional symmetrical Root-Raised-Cosine Pulse (see
23 Section 21.5.1) and a Precursor-Free Pulse (see Section 21.5.2).

24 21.5.1 Symmetrical Root-Raised-Cosine Pulse – PulseShape 0x0

25 For the symmetrical Root-Raised-Cosine Pulse, referred to as PulseShape 0x0, the RF envelope
26 (or equivalent baseband pulse amplitude) is expressed by [31]:

$$r(t) = \frac{4\beta}{\pi\sqrt{T_p}} \frac{\cos((1+\beta)\pi t/T_p) + \frac{\sin((1-\beta)\pi t/T_p)}{4\beta(t/T_p)}}{1 - (4\beta t/T_p)^2}$$

27 where:

28 β = roll-off factor with a value of 0.45 ± 0.05

29 T_p = Pulse duration, which is inverse of the chip frequency (1/499.2MHz)

1 Finite impulse response (FIR) samples of PulseShape 0x0 using β of 0.45 with 32x oversampling
 2 per chip and total span of 8 chips are included in Appendix H as informative reference.

3 21.5.2 Precursor-Free Pulse – PulseShape 0x1

4 In general, “precursor-free” refers to the pulse’s property to monotonically raise its amplitude to
 5 the first peak that also represents its main lobe. PulseShape 0x1 represents a group of pulse
 6 shapes that meet the time-domain mask of Figure-89 in [33].

7 21.5.2.1 Precursor-Free Pulse – PulseShape 0x2

8 PulseShape 0x2 is a special case within PulseShape 0x1, by applying minimum-phase
 9 conversion of PulseShape 0x0 using method described in [31]. A Python implementation of the
 10 linear-to-minimum phase conversion is included in Appendix G, and FIR samples for a span of 8
 11 chips at oversampling of 32x and β of 0.45 for PulseShape 0x2 are again included in Appendix
 12 H, both provided as informative references. It is recommended to use PulseShape 0x2 whenever
 13 possible if transmitter chooses to implement precursor-free pulse shape.

14 Table 21-3 summarizes the different pulse shapes that allowed in the specification. However, it is
 15 recommended to use PulseShape 0x2.

16 *Table 21-3: Summary of pulse shapes.*

PulseShape	Description	bitfield
Symmetrical Root-Raised-Cosine Pulse	Symmetrical Root-Raised-Cosine Pulse, referred to as PulseShape 0x0. The RF envelope or equivalent baseband pulse amplitude is expressed by [31].	0x0
Precursor-Free Pulse	Group of pulse shapes that meet the time-domain mask of Fig. 89 in [33].	0x1
	A special case within PulseShape 0x1, by applying minimum-phase conversion of PulseShape 0x0 using method described in [31].	0x2

17 21.5.3 PulseShape Combinations

18 Compliant receivers shall be able to receive and suitably process both symmetrical and
 19 precursor-free pulses. It is up to the implementer whether the same receiver configuration is used
 20 for PulseShape 0x2 and PulseShape 0x1, or whether reception of PulseShape 0x2 is explicitly
 21 optimized via a separate receiver configuration.

22 Possible transmitter PulseShape combinations (PulseShape_Combo) for the initiator and
 23 responder are shown in Table 21-4. The responder’s and initiator’s receive configurations shall
 24 be set according to the PulseShape selections made by the initiator’s and responder’s
 25 transmitters, respectively.

26 *Table 21-4: Overview of PulseShape_Combo values and the associated transmit pulse shapes.*

PulseShape_Combo	Transmit PulseShape	
	Initiator	Responder
0x00	0x0	0x0
0x01	0x0	0x1
0x02	0x0	0x2

PulseShape_Combo	Transmit PulseShape	
	Initiator	Responder
0x10	0x1	0x0
0x11	0x1	0x1
0x12	0x1	0x2
0x20	0x2	0x0
0x21	0x2	0x1
0x22	0x2	0x2

1 21.6 Ranging Marker

2 The RMARKER as defined in [33] shall be supported for taking timestamp measurements.

3 21.7 UWB RF

4 21.7.1 *Operating frequency bands and Channel assignments*

5 The UWB technology as specified in [31] supports three independent bands of operation and a
6 wide frequency range from hundreds of MHz to several GHz:

- 7 • The sub-gigahertz band, which consists of a single channel and occupies the spectrum
8 from 249.6 MHz to 749.6 MHz.
- 9 • The low band, which consists of four channels and occupies the spectrum from 3.1 GHz
10 to 4.8 GHz.
- 11 • The high band, which consists of 11 channels and occupies the spectrum from 6.0 GHz to
12 10.6 GHz.

13 In this specification, the ERDEV shall operate in the high band using channel 5 and channel 9.
14 Channel 5 uses center frequency of 6489.6 MHz and channel 9 uses center frequency of 7987.2
15 MHz. ERDEV that only operates in specific geographic regions where regulatory restrictions
16 prevent the use of specific band(s) may be exempted from this requirement.

17 21.7.2 *Frequency Source Requirements*

18 RF center frequency and the symbol clock frequency shall all be derived from the same reference
19 clock (clk_ref), which is typically implemented as a crystal oscillator (XO). Clock and sampling
20 rates for all transmitter and receiver activities in the PHY and the RF Carrier Frequency shall be
21 tied to the same common clk_ref.

22 **clk_ref tolerance**

23 For the device, clk_ref tolerance should be regulated within ± 10 ppm over operation conditions.
24 For the vehicle, clk_ref tolerance should be regulated within ± 15 ppm over operation conditions.

25

1 **22 UWB SECURITY [WCC3]**

2 This section addresses the security requirements inherent (applicable and enforced) to the UWB-
3 related functionality of the Digital Key feature. The device OEM shall use NFC as a fallback
4 mechanism in case correct operation of the UWB and the BLE module is disturbed; Information
5 about the fallback mechanism should be provided by the device OEM to the user.

6 **22.1 Cryptography**

7 This section details the various cryptographic functions identified in the various security
8 functions covered earlier in the document.

9 **Build STS based on cryptographic DRBG**

10 Section 20 describes:

- 11 • MAC protocol (Ranging session and MAC control Channel)
- 12 • STS Index Incrementation (rules, incrementation and update)

13 **22.1.1 Deriving the STS Sequence from the DRBG**

14 The derivation of the STS follows the sequence described in Section 15.2.9.1 and Figure 15-7b
15 of [33]. For the purpose of STS generation and in the context of this specification, the following
16 parameters as defined in [33] are mapped to the corresponding parameters as defined in this
17 specification as follows:

18 phyHrpUwbStsVCounter = SaltedHash[0:32]
19 phyHrpUwbStsVUpper96 = SaltedHash([64:64])|| ((SaltedHash[32:32]+STS_Index) & 0xffffffff)
20 phyHrpUwbStsKey = dURSK[0:128]

21 The 32-bit counter is initialized with phyHrpUwbStsVCounter for each STS generation, that
22 means for each slot. The 32-bit counter is incremented by one for each 128-bit chunk of the STS,
23 that means 31 increments for a 4096-bit long STS.

24 SaltedHash[0:128], STSIndex[0:32], and Counter[0:32] (the same one as the IEEE 32-bit
25 counter, initialized at 0) are used to derive phyHrpUwbStsVUpper96 and
26 phyHrpUwbStsVCounter.

27 **22.1.2 SP0 Frames Encryption**

28 SP0 shall be encrypted using the AES-CCM* algorithm as described in Section 9.3 of [31].
29 Unlike the NIST specification for CCM which requires encrypted data length > 0, the IEEE
30 802.15 CCM* implementation allows authentication-only CCM (as used in [31] with Security
31 Level set to 1, 2, or 3).

32 **STS_index encryption (SP0 Pre-POLL)**

- 33 • *Algorithm*: AES-CCM* as described in Section 9.3 of [31]
- 34 • *Mode*: Encrypted and authenticated [31](Security Level 6 - ENC-MIC64)

- $K(Key) = \text{mUPSK1}[0:128]$
 - $N(Nonce) = \text{Source Long Address} \mid \text{Frame Counter} \mid \text{Nonce Security Level}$ as described in Section 9.3.2.1 of [31]
 - Incrementing Frame Counter, STS_Index in payload

Note: The key mUPSK1 is static for the ranging session.

Timestamps encryption (SP0 Final_Data timestamp frame)

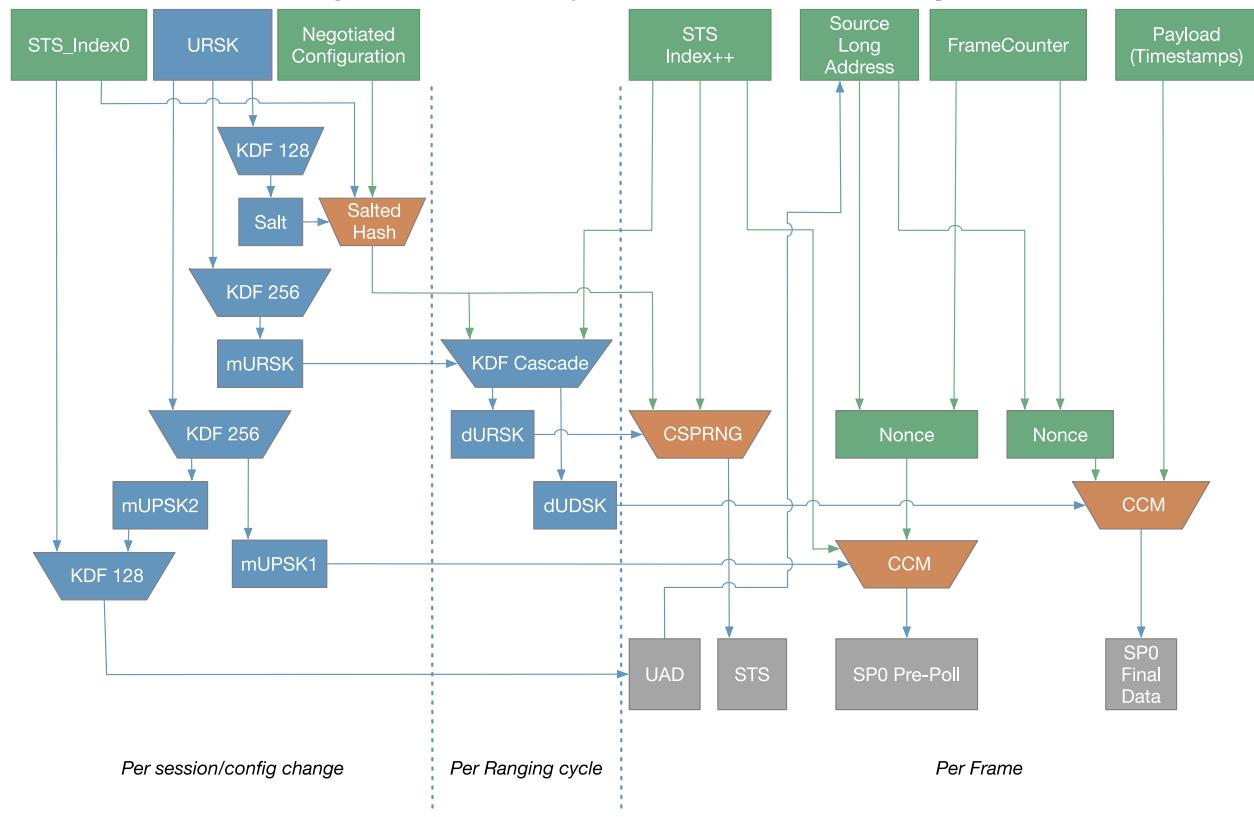
- Algorithm: AES-CCM* per as described in Section 9.3 of [31]
 - Mode: Encrypted and authenticated [31](Security Level 6 - ENC-MIC64)
 - K (Key) = dUDSK, where dUDSK is defined later in this subsection
 - N (Nonce) = Source Address | Frame Counter | Nonce Security Level as described in Section 9.3.2.1 of [31]
 - Incrementing Frame Counter

Note: The probability of IV/Key repetition is insignificant since the key dUDSK shall be updated for every ranging block.

22.1.3 Key Derivation Functions (KDFs)

The cryptographic foundations of the protocol shall be established with the following cascaded KDFs as shown in Figure 22-1. Information about specific key derivations is in the subsequent sections.

Figure 22-1: Overview of the KDF used and its relationship.



1 22.1.3.1 *mUPSKs Key Derivation*

2 **Purpose:** Protection of STS_Index

3 **Occurrence:** Once per session

4 **Cryptography:**

5 mUPSK1[0:128] = CMAC(URSK, 0x00000001 || **0x5550534B** || 0x00 ||
6 0x000000 || 0x00000180)

7 mUPSK2[128:128] = CMAC(URSK, 0x00000002 || **0x5550534B** || 0x00 ||
8 0x000000 || 0x00000180)

9 mUPSK2[0:128] = CMAC(URSK, 0x00000003 || **0x5550534B** || 0x00 ||
10 0x000000 || 0x00000180)

11 CMAC= AES256-CMAC as PRF, r = 32, h=128

12 **Reference:** See Section 5.1 of [4] KDF in Counter Mode

13 **Note:** ASCII (**0x5550534B**) = “UPSK”

14 mUPSK1 is used in order to encrypt the SP0 Pre-POLL.

15 mUPSK2 is used to derive the KeySource and UWB addresses.

16 22.1.3.2 *SaltedHash Key Derivation*

17 **Purpose:** Preserve integrity of ranging configuration

18 **Occurrence:** Once per ranging configuration negotiation

19 **Cryptography:**

20 Salt[0:128] = CMAC(URSK, 0x00000001 || **0x53414C54** || 0x00 || 0x000000 ||
21 0x00000080)

22 CMAC= AES256-CMAC as PRF, r = 32, h=128

23 **Reference:** See Section 5.1 of [4] KDF in Counter Mode

24 PaddedSalt[0:256] = 0x000..0000 || Salt

25 RangingConfiguration = Selected_Protocol_Version || Selected_UWB_Config_Id
26 || UWB_Session_Id || STS_Index0 || Number_Responder_Nodes ||
27 Session_RAN_Multiplier || Number_Slot_per_Round || Number_Chaps_per_Slot
28 || Selected_PulseShape_Combo

29 SaltedHash = CMAC(PaddedSalt, RangingConfiguration)

30 CMAC = AES256-CMAC as PRF, r = 32, h=128

31 **Note:** ASCII (**0x53414C54**) = “SALT”

32 RangingConfiguration parameters are fully described in Section [20.5.2](#).

33 22.1.3.3 *mURSK Key Derivation*

34 **Purpose:** Ranging Keys Seed derivation

35 **Occurrence:** Once per session

1 **Cryptography:**

2 mURSK[128:128]
3 = CMAC(URSK, 0x00000001 || **0x5552534B** || 0x00 || 0x000000 || 0x00000100)
4 mURSK[0:128]
5 = CMAC(URSK, 0x00000002 || **0x5552534B** || 0x00 || 0x000000 || 0x00000100)

6 CMAC = AES256-CMAC as PRF, r = 32, h=128

7 **Reference:** Section 5.1 of [4] KDF in Counter Mode

8 **Note:** ASCII (**0x5552534B**) = “URSK”

9 *22.1.3.4 dURSK/dUDSK Keys Derivation (KDF_cascade definition)*

10 **Purpose:** Ranging Keys derivation

11 **Occurrence:** KDF executed once per ranging cycle (before “POLL”)

12 **Cryptography:**

13 URSK_KT[128:128] = CMAC(mURSK, 0x00000001 || **0x5552534B5F4B54** ||
14 0x00 ||
15 7'b0 || STS_Index[31] || .. 7'b0 || STS_Index[16] || 7'b0 || STS_Index[15] || ..
16 7'b0 || STS_Index[1] || 7'b0 || STS_Index[0] || 0x00000100)
17 where 7'b0 represents seven binary zeros and STS_Index =
18 STS_Index^k(I, Round_Idx^k(i), Pre_POLL)

19 URSK_KT[0:128] = CMAC(mURSK, 0x00000002 || **0x5552534B5F4B54** ||
20 0x00 ||
21 7'b0 || STS_Index[31] || .. 7'b0 || STS_Index[16] || 7'b0 || STS_Index[15] || ..
22 7'b0 || STS_Index[1] || 7'b0 || STS_Index[0] || 0x00000100)

23 dURSK[0:128]

24 = CMAC(URSK_KT, 0x00000001 || **0x5552534B** || 0x00 || SaltedHash ||
25 0x000000 || 0x00000080)

26 dUDSK[0:128]

27 = CMAC(URSK_KT, 0x00000001 || **0x5544534B** || 0x00 || SaltedHash ||
28 0x000000 || 0x00000080)

29 CMAC = AES256-CMAC as PRF, r = 32, h=128

30 **Reference:** Section 5.1 of [4] KDF in Counter Mode

31 **Note:** ASCII (**0x5552534B5F4B54**) = “URSK_KT”

32 ASCII (**0x5552534B**) = “URSK”

33 ASCII (**0x5544534B**) = “UDSK”

1 **22.1.4 UWB Tracking Prevention Between Ranging Recoveries**

2 In order to protect against tracking of the same UWB session at session setup or upon recovery
3 (i.e., upon receiving the RR-RS message), the following actions shall be performed:

- 4 • The following values shall be derived:
5 ○ KeySource (4 bytes)
6 ○ Destination Short Address (2 bytes)
7 ○ Source Long Address (8 bytes)
8 • The Frame Counter shall be reset to 0.

9 **22.1.4.1 UWB Addresses KDF**

10 **Purpose:** Deriving UWB Addresses (UAD)

11 **Occurrence:** At ranging setup or at ranging recovery

12 **Cryptography:**

13
$$\text{UAD}[0:128] = \text{CMAC}(\text{mUPSK2}, 0x00000001 \parallel \text{0x554144} \parallel 0x00 \parallel$$

14
$$\text{STSIndex0}[0:32] \parallel 0x00000080)$$

15 Some addresses shall be reserved for broadcast use:

- 16 • For the KeySourceLow, KeySourceHigh and the destination short address: 0xFFFF and
17 0xFFFE are reserved values.
18 • For the source long address: 0xFFFFFFFFFFFFFF is a reserved value.

19 The following method shall be used to flip the upper bit if the calculated UAD matches the
20 reserved value:

```
21          function SetUpperBitToZeroIfReserved(bytes [X...0]) {  
22            if (bytesAreReserved(bytes)) {  
23              bytes[X...X-7] = bytes[X...X-7] & 0x7f  
24            }  
25            return bytes  
26        }
```

27 *Note:* The indexing notation still refers to bit-indexes.

28 The SetUpperBitToZeroIfReserved remaps the following entries and leaves the others
29 unmodified:

Reserved Address	Remapped Address
0xFFFF	0x7FFF
0xFFFE	0x7FFE
0xFFFFFFFFFFFFFF	0x7FFFFFFFFFFFFFFF

30
$$\text{KeySourceLow}[0:16] = \text{SetUpperBitToZeroIfReserved}(\text{UAD}[112:16])$$

31
$$\text{KeySourceHigh}[0:16] = \text{SetUpperBitToZeroIfReserved}(\text{UAD}[96:16])$$

1 DestinationShortAddress[0:16] = SetUpperBitToZeroIfReserved(UAD[80:16])
2 SourceLongAddress[0:64] = SetUpperBitToZeroIfReserved(UAD[16:64])

3 CMAC = AES256-CMAC as PRF, r = 32, h=128

4 **Reference:** Section 5.1 of [4] KDF in Counter Mode

5 **Note:** ASCII (0x554144) = “UAD”

6 The latest STSIndex0 passed is the one negotiated during session setup or recovery.

7 *22.1.4.2 Key Source for Auxiliary Security Header*

8 The 4-byte Key Source value shall be obtained as follows:

9 KeySource [0:32] = KeySourceHigh || KeySourceLow

10

11 **22.2 UWB Ranging**

12 The following applies to the UWB ranging session:

13 *22.2.1 STS Index Management*

14 The STS Index Incrementation shall comply with the following requirements (see Section 20.6).

- 15 1. The initial value of STS_Index selected at the start of a given ranging session (referred
16 here as initial STS_Index0). It shall be a random value in the [0..2^30] range, therefore
17 the STS_Index may be incremented at least 2^30 times. The initial STS_Index0 is
18 included in Ranging Session Setup Response (see Section 19.3.1.6).
- 19 2. The STS_Index shall be monotonically increasing. Within a given ranging session
20 identified by UWB_Session_ID, the STS_Index shall never go back be decremented or
21 reused between any two different frames.
- 22 3. The maximum value of STS_index in a given ranging session shall be 2^31-1. When this
23 maximum value is reached, the ranging session shall be ended and a new session with a
24 new URSK shall be used.
- 25 4. The first STS_Index of a recovered ranging session (referred here as recovery
26 STS_Index0) is included in Ranging Recovery Response (See Section 19.3.1.10) or
27 Configurable Ranging Recovery Response (see Section 19.3.1.12). The recovery
28 STS_Index0 shall be greater than the last used STS_Index prior to Ranging Suspension
29 which is within the same ranging session.
- 30 5. The STS_Index value included in the Poll_STS_Index parameter of the first Pre-POLL
31 message (received by the vehicle) shall be larger than the initial STS_Index0 or recovery
32 STS_Index0 for the case of a new ranging session or a recovery of a suspended ranging
33 session, respectively.

34 **Re-synchronization**

35 The re-synchronization of the anchors happens on the vehicle side.

36 This mechanism involves the mUPSK1, the STS_index, and a nonce (calculated from data in the
37 MAC header): the device performs the encryption of the frame (SP0) and the vehicle decrypts.

1 **22.3 UWB Module**

2 The pre-derived URSKs are generated by the DK applet. Then, when it becomes active, the
3 URSK or sub keys derived from this URSK (depending on the device architecture) are
4 transferred to the UWB Module.

5 The UWB Module is notably in charge of the processing of some cryptography operation like
6 PPDU ciphering or derivation of the STS. An implementation shall provide hardware and
7 software protection against logical and certain physical attacks for the full lifecycle of all assets
8 related to the UWB secure ranging.

9 The transport channel between a SE and the UWB module shall protect confidentiality and
10 integrity. The SE shall provide a secure binding with the UWB module that shall be resistant to
11 manipulation after production. It shall be prevented that any code running on the application
12 processor has access to or can inject data on this channel. The SE shall have the ability to
13 distinguish communication between the UWB chip interface, and other interfaces.

14 The potential associated with an effective attack on the UWB module and its interface with the
15 SE should be expressed in terms of the total effort required to mount a successful attack as
16 defined by the identified threats. The detailed semantics of such attack potential is to be
17 determined based on definition and related documentation by the DKCTG (e.g. upon conducting
18 a threat analysis exercise).

19 Appropriate security measures shall be taken to prevent an attacker with expert expertise and
20 bespoke equipment from leveraging a vulnerability to mount a successful exploit on any of the
21 CCC DK Release 3 solution components, that would defeat the security of the use case (passive
22 keyless entry or start engine), given sensitive knowledge of the implemented solution and that a
23 few months time is available for searching vulnerabilities.

24

1 23 REFERENCES

2 The documents listed in this section are included in requirements made in the body of this
3 specification. Knowledge of their contents is required for the understanding and implementation
4 of this specification. If a listing includes a date or a version identifier, only that specific version
5 of the document is required. If the listing includes neither a date nor a version identifier, the
6 latest version of the document is required. External references referred herein should apply to the
7 latest version provided they are backward compatible with the version at the time of writing and
8 are not violating the implementation specified in this specification.
9

- 10 [1] ISO/IEC 7816-4: Fourth Edition May 2020
11 [2] ISO 8601-1:2019: Date and time – Representations for information interchange – Part 1:
12 Basic rules
13 [3] RFC 5280 - PKIX Certificate and CRL Profile - May 2008
14 [4] NIST SP 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions
15 - May 2005 (Updated August 2022 as NIST SP 800-108r1)
16 [5] NIST SP 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC
17 Mode for Authentication – May 2005 (Updated October 2016)
18 [6] BSI TR-03111 - Elliptic Curve Cryptography - Version 2.0
19 [7] GPC_SPE_014 - GlobalPlatform - Secure Channel Protocol 03 - Amendment D - Version
20 1.1.1 or later version
21 [8] FIPS PUB 186-4 - Digital Signature Standard - July 2013
22 [9] NIST SP 800-38A - Block Cipher Modes of Operation: Methods and Techniques - 2001
23 [10] Network Working Group Internet-Draft: SPAKE2+, an Augmented PAKE, draft-bar-
24 cfrg-spake2plus-00, Mar 9, 2020
25 [11] Internet Engineering Task Force (IETF): The Scrypt Password-Based Key Derivation
26 Function (RFC 7914), August 2016
27 [12] Internet Engineering Task Force (IETF): Randomness Requirements for Security (RFC
28 4086), June 2005
29 [13] RFC 5869 - HMAC-based Extract-and-Expand Key Derivation Function - May 2010
30 [14] RFC 5480 - ECC SubjectPublicKeyInfo Format - October 2009
31 [15] GlobalPlatform Card Technology – Confidential Card Content Management, Card
32 Specification v2.3 Amendment A Version 1.2 (or higher)
33 [16] NFC Analog Technical Specification 2.1 (<https://nfc-forum.org/our-work/specifications-and-application-documents/>) or later version.
34
35 [17] NFC Digital Protocol Technical Specification 2.1 (<https://nfc-forum.org/our-work/specifications-and-application-documents/>) or later version.
36
37 [18] NFC Activity Technical Specification 2.0 (<https://nfc-forum.org/our-work/specifications-and-application-documents/>) or later version.
38

- 1 [19] [NFC Controller Interface Technical Specification 2.1](https://nfc-forum.org/our-work/specifications-and-application-documents/) (<https://nfc-forum.org/our-work/specifications-and-application-documents/>) or later version.
- 2 [20] GlobalPlatform Card Technology Card Specification v2.2.1 (or higher)
- 3 [21] GlobalPlatform Card – Contactless Extension Version 1.0 (or higher)
- 4 [22] SEC 1: Elliptical Curve Cryptography Version 2.0, <https://www.secg.org/sec1-v2.pdf>
- 5 [23] ISO/IEC 10118-3 Hashfunctions - Part 3: Dedicated hash-functions
- 6 [24] FIPS PUB 180-4: Specifications for the Secure Hash Standard – 2015
- 7 [25] ETSI TS 102 705 Release 9.2.0 (or higher) Smart Cards; UICC Application
9 Programming Interface for Java Card™ for Contactless Applications,
10 <https://www.etsi.org/standards>
- 11 [26] ETSI TS 102 622 Release 9.4.0 (or higher) Smart Cards; Contactless Front End, HCI
12 (Host Controller Interface), <https://www.etsi.org/standards>
- 13 [27] RFC 3986 – Uniform Resource Identifier (URI): Generic Syntax – January 2005
- 14 [28] GPC_SPE_093 - GlobalPlatform - Secure Channel Protocol '11' - Amendment F -
15 Version 1.1
- 16 [29] Car Connectivity Consortium "Android Digital Key Framework API", Version 1.0, Aug
17 2020
- 18 [30] The Bluetooth Core specification, version 5.0, December 2016
- 19 [31] IEEE Standard for Low-Rate Wireless Networks, 802.15.4-2020, July 2020
- 20 [32] CCC Digital Key Release 2 Specification version 1.1
- 21 [33] IEEE 802.15-4z-2020 - IEEE Standard for Low-Rate Wireless Networks, Amendment 1:
22 Enhanced Ultra Wideband (UWB) Physical Layers (PHYs) and Associated Ranging
23 Techniques, June 2020
- 24 [34] N. Damera-Venkata, B. L. Evans, S. R. McCaslin, “Design of optimal minimal phase
25 filter using discrete Hilbert transform,” IEEE Trans. Signal Proc., vol. 48, pp. 1491-1495,
26 May 2000
- 27 [35] CCC Vehicle OEM Document, D1.1
- 28 [36] Description 15.4z HRP UWB PHY Test Vectors”, IEEE 802.15 document number 15-20-
29 0003-02-004z, 2020.
- 30 [37] RFC 2330 - Framework for IP Performance Metrics,
31 <https://tools.ietf.org/html/rfc2330#section-10>
- 32 [38] Internet Engineering Task Force (IETF): Secure Credential Transfer - draft-secure-
33 credential-transfer-03, <https://datatracker.ietf.org/doc/draft-secure-credential-transfer/>
- 34 [39] NIST Special Publication 800-63B, Digital Identity Guidelines, June 2017 including
35 updates of 03-02-2020
- 36 [40] ITU Telecommunication Standardization Sector - Recommendation ITU-T X.690
37 International Standard 8825-1

1

2 Appendix A.

3 A.1. Standard Transaction Cryptography Flow

4 Listing A-1: Standard Transaction Cryptography Flow

```
1 Vehicle key pair:  
2 vehicle public key X: F47EB42A771052580C086EFDAAA3084AA3FF7A67CE23393A0373C63487DF1A63  
3     vehicle public key Y: 7D1FB34B2D2E7D5C8F92097A0619B5C5CC6C5850AF74C019EBBEC4273358AA94  
4 vehicle private key: 86B9E3843D949890FD50E49C8542DB575BAC41D344F17588DDAFE4535521CE55  
5 Endpoint long term key pair:  
6     endpoint public key X: 07B857B9B7F1147E20F4DBE6723CE5F46EF8670CBA20F56297F515C8265E4E42  
7     endpoint public key Y: 5F1FC9B5DAFB62DAAFB5DC9A6F8B2EDC1CDD43E20A614EF2F8703FA1459721C  
8     endpoint private key: 13849B3560961053D985DA5E0FA2AF05EAA99DD73EC50CC4A87029A24D7C331B  
9  
10    >>00A4040005AAAAAAAAAA00  
11  
12 endpoint: received SELECT command  
13  
14    <<5C0201009000  
15  
16 Generate ephemeral keys:  
17     ephemeral public key X: F98CCA31651AD2E63266144B2450FD6081D8FEA8CEB826E1FB10E8034E932446  
18     ephemeral public key Y: CAD19D201062DD1C7CB0BB293BF16A4BEFB2ED500977E7197E01F26906E39B5F  
19     ephemeral private key: B0FBA5FB966FDD3BE4096FA65307AB0A7A3BB914625BBFD3CB57DAD9183E19CB  
20  
21    >>80800000635C020100874104F98CCA31651AD2E63266144B2450FD6081D8FEA8  
22     CEB826E1FB10E8034E932446CAD19D201062DD1C7CB0BB293BF16A4BEFB2ED50  
23     0977E7197E01F26906E39B5F4C10BF1C41268230AF76BFFE3E7C5D00CF4A4D08  
24     888888888888888800  
25  
26 endpoint: received AUTH0 command  
27 Generate ephemeral keys:  
28     ephemeral public key X: 43D605526999F032E08F314F22EBCE051D1DAE53DC71F1C4D614B0337BB17F20  
29     ephemeral public key Y: 3F95D4C06AB8966D2B9A0D3C4BC446DB9343EBF27F9EF811F242A37118AD4F10  
30     ephemeral private key: E585C9EE89075F795452879AC38261ED0667C6396A34914DEE0681E8DC22A182  
31 Key Derivation:  
32 K: ACEDD14246C16AAF4561E177E567192454C06B2AAEEDB7E278980C25DD7994A2  
33 endpoint ephemeral public key X: 43D605526999F032E08F314F22EBCE051D1DAE53DC71F1C4D614B0337BB17F20  
34 reader ephemeral public key X: F98CCA31651AD2E63266144B2450FD6081D8FEA8CEB826E1FB10E8034E932446  
35 transaction identifier: BF1C41268230AF76BFFE3E7C5D00CF4A  
36 interface byte: 5E  
37 flag: 0000  
38 HKDF output: 55AC05B56D3081AAEF7BD8EB53F9B5BAAE126251B6224775A72770B3E10BC65D  
39 F663103140A7832C8AD155E7377E63303DE2D8811B60F1225CAD4FCBDE0BAA3C  
40 Kcmac: 55AC05B56D3081AAEF7BD8EB53F9B5BA  
41 Kenc: AE126251B6224775A72770B3E10BC65D  
42 Kmac: F663103140A7832C8AD155E7377E6330  
43 Krmac: 3DE2D8811B60F1225CAD4FCBDE0BAA3C  
44  
45    <<86410443D605526999F032E08F314F22EBCE051D1DAE53DC71F1C4D614B0337B  
46     B17F203F95D4C06AB8966D2B9A0D3C4BC446DB9343EBF27F9EF811F242A37118  
47     AD4F109000  
48  
49 generate authentication hash:  
50 sha256 hash input: 4D088888888888888862043D605526999F032E08F314F22EBCE051D1DAE53  
51 DC71F1C4D614B0337BB17F208720F98CCA31651AD2E63266144B2450FD6081D8
```



```
168 reader : mac verification successful
169 generate authentication hash:
170 sha256 hash input: 4D0888888888888888862043D605526999F032E08F314F22EBCE051D1DAE53
171 DC71F1C4D614B0337BB17F208720F98CCA31651AD2E63266144B2450FD6081D8
172 FEA8CEB826E1FB10E8034E9324464C10BF1C41268230AF76BFFE3E7C5D00CF4A
173 93044E887B4C
174 sha256 hash output: 48BCCE4843E4E87A01AEC830A1AAF6E7D1380D950C468F81BB5AD4CF40705040
175 reader: endpoint signature verification successful
176 request to read private mailbox offset:0 length:5
177 request to write private mailbox offset:0 length:5 data:FFEEEEEDDBB
178 request to read confidential mailbox offset:0 length:5
179 request to write confidential mailbox offset:0 length:5 data:AAEEEE33CC
180 wrap command:
181 plaintext:0088030000058A070000FFEEEEEDDBB89030000058B070000AAEEEE33CC
182 MAC chaining value:3B30EA2AB387B0FEB5D9D899B944E17B
183 cipher|mac:F094F8445A84E178484E167B1FD08DBB2C30C61EE0CA41FCE4F6B6A139740988
184 3B30EA2AB387B0FE
185
186 >>84C9000028F094F8445A84E178484E167B1FD08DBB2C30C61EE0CA41FCE4F6B6
187 A139740983B30EA2AB387B0FE00
188
189 endpoint: received EXCHANGE command
190 unwrap command:
191 cipher|mac:F094F8445A84E178484E167B1FD08DBB2C30C61EE0CA41FCE4F6B6A139740988
192 3B30EA2AB387B0FE
193 MAC chaining value:0000000000000000000000000000000000000000000000000000000000000000
194
195 plaintext:0088030000058A070000FFEEEEEDDBB89030000058B070000AAEEEE33CC
196 wrap response:
197 plaintext:05AAAAAAAAAA05BBBBBBBBBB
198 ICV:925FE52E4E2793662479418FE2C8A3C0
199 cleartext payload:05AAAAAAAAAA05BBBBBBBBBB80000000
200 MAC chaining value:3B30EA2AB387B0FEB5D9D899B944E17B
201 cipher|mac:73DFAE8DF93751D1169C4295565220F2B10F64BA83956435
202
203 <<73DFAE8DF93751D1169C4295565220F2B10F64BA839564359000
204
205 unwrap response:
206 cipher|mac:73DFAE8DF93751D1169C4295565220F2B10F64BA83956435
207 plaintext:05AAAAAAAAAA05BBBBBBBBBB
```

1

2 A.2. Fast Transaction Cryptography Flow

3 Listing A-2: Fast Transaction Cryptography Flow

```
1 Vehicle key pair:
2 vehicle public key X: F47EB42A771052580C086EFDA3084AA3FF7A67CE23393A0373C63487DF1A63
3 vehicle public key Y: 7D1FB34B2D2E7D5C8F92097A0619B5C5CC6C5850AF74C019EBBEC4273358AA94
4 vehicle private key: 86B9E3843D949890FD50E49C8542DB575BAC41D344F17588DDAFE4535521CE55
5 Endpoint long term key pair:
6 endpoint public key X: 07B857B9B7F1147E20F4DBE6723CE5F46EF8670CBA20F56297F515C8265E4E42
7 endpoint public key Y: 5F1FC9B5DAFB62DAAFB5DC9AA6F8B2EDC1CDD43E20A614EF2F8703FA1459721C
8 endpoint private key: 13849B3560961053D985DA5E0FA2AF05EAA99DD73EC50CC4A87029A24D7C331B
9
10 >>00A4040005AAAAAAAAAA00
11
12 endpoint: received SELECT command
```

```
13 |<<C0201009000
14 |
15 |
16 |Generate ephemeral keys:
17 |ephemeral public key X: 82BF9E948ECBDD73C10C7EB7D34D5BEB31CF2908B09ADAC701CB4B1F116F5467
18 |ephemeral public key Y: C9187749054455AA1231FA6562D6D4198779FEC2F4F36DB8D9D6EF2082EEA75B
19 |ephemeral private key: E82CED017293885DC5D157A6DAC87013A72B3182F94939BFAA92BB9A367E7966
20 |
21 |>>80800100635C02010087410482BF9E948ECBDD73C10C7EB7D34D5BEB31CF2908
22 |B09ADAC701CB4B1F116F5467C9187749054455AA1231FA6562D6D4198779FEC2
23 |F4F36DB8D9D6EF2082EEA75B4C10F92F7260B588238C1E2A4825AD4D7D2E4D08
24 |88888888888888800
25 |
26 |endpoint: received AUTH0 command
27 |Generate ephemeral keys:
28 |ephemeral public key X: 0EA56A82A1AD7FC2C739FBB793C0BC3B8935C2ED46B672EFCB98F7DF124FA7FF
29 |ephemeral public key Y: A4155A91F0FCBB007C61E6C574F0F87D3CAF1F41EDE0DF87F43DB664B2C81540
30 |ephemeral private key: BCD0A7ECE3BD6A27A2E0597DAA647028E9058D415921A282C0B18DE90E6EFA94
31 |Key Derivation:
32 |K: B1E9126FBB4FFCA027AE116FC242A1F93093082DE8661B3CD1942078DEB384FD
33 |endpoint ephemeral public key X:
34 |0EA56A82A1AD7FC2C739FBB793C0BC3B8935C2ED46B672EFCB98F7DF124FA7FF
35 |vehicle ephemeral public key X: 82BF9E948ECBDD73C10C7EB7D34D5BEB31CF2908B09ADAC701CB4B1F116F5467
36 |transaction identifier: F92F7260B588238C1E2A4825AD4D7D2E
37 |interface byte: 5E
38 |flag: 0100
39 |HKDF output: 960BA7366865139D9EFAEFB53B8CB18713736C529481B67615811D4EA9F49650
40 |AAA090CC271484E68BB5B6EE18655AAFA5B82D564DCD9961DEE2830C6550E1C1
41 |Kcmac: 960BA7366865139D9EFAEFB53B8CB187
42 |Kenc: 13736C529481B67615811D4EA9F49650
43 |Kmac: AAA090CC271484E68BB5B6EE18655AAF
44 |Krmac: A5B82D564DCD9961DEE2830C6550E1C1
45 |generate cryptogram:
46 |Kcmac: 960BA7366865139D9EFAEFB53B8CB187
47 |endpoint public key X: 07B857B9B7F1147E20F4DBE6723CE5F46EF8670CBA20F56297F515C8265E4E42
48 |vehicle public key X: F47EB42A771052580C086EFDAAA3084AA3FF7A67CE23393A0373C63487DF1A63
49 |vehicle identifier: 888888888888888
50 |transaction identifier: F92F7260B588238C1E2A4825AD4D7D2E
51 |input of AES-CMAC: 00000000000000000000003200008001F47EB42A771052580C086EFDAAA3084A
52 |A3FF7A67CE23393A0373C63487DF1A6307B857B9B7F1147E20F4DBE6723CE5F4
53 |6EF8670CBA20F56297F515C8265E4E42F92F7260B588238C1E2A4825AD4D7D2E
54 |888888888888888
55 |cryptogram: E5B79C3D703D1BE1B26C2A999DB2975B
56 |
57 |<<8641040EA56A82A1AD7FC2C739FBB793C0BC3B8935C2ED46B672EFCB98F7DF12
58 |4FA7FFA4155A91F0FCBB007C61E6C574F0F87D3CAF1F41EDE0DF87F43DB664B2
59 |C815409D10E5B79C3D703D1BE1B26C2A999DB2975B9000
60 |
61 |Key Derivation:
62 |K: B1E9126FBB4FFCA027AE116FC242A1F93093082DE8661B3CD1942078DEB384FD
63 |endpoint ephemeral public key X:
64 |0EA56A82A1AD7FC2C739FBB793C0BC3B8935C2ED46B672EFCB98F7DF124FA7FF
65 |vehicle ephemeral public key X: 82BF9E948ECBDD73C10C7EB7D34D5BEB31CF2908B09ADAC701CB4B1F116F5467
66 |transaction identifier: F92F7260B588238C1E2A4825AD4D7D2E
67 |interface byte: 5E
68 |flag: 0100
69 |HKDF output: 960BA7366865139D9EFAEFB53B8CB18713736C529481B67615811D4EA9F49650
70 |AAA090CC271484E68BB5B6EE18655AAFA5B82D564DCD9961DEE2830C6550E1C1
```

```
69 Kcmac: 960BA7366865139D9EFAEFB53B8CB187
70 Kenc: 13736C529481B67615811D4EA9F49650
71 Kmac: AAA090CC271484E68BB5B6EE18655AAF
72 Krmac: A5B82D564DCD9961DEE2830C6550E1C1
73 generate cryptogram:
74 Kcmac: 960BA7366865139D9EFAEFB53B8CB187
75 endpoint public key X: 07B857B9B7F1147E20F4DBE6723CE5F46EF8670CBA20F56297F515C8265E4E42
76 vehicle public key X: F47EB42A771052580C086EFDA3084AA3FF7A67CE23393A0373C63487DF1A63
77 vehicle identifier: 8888888888888888
78 transaction identifier: F92F7260B588238C1E2A4825AD4D7D2E
79 input of AES-CMAC: 00000000000000000000003200008001F47EB42A771052580C086EFDA3084A
80 A3FF7A67CE23393A0373C63487DF1A6307B857B9B7F1147E20F4DBE6723CE5F4
81 6EF8670CBA20F56297F515C8265E4E42F92F7260B588238C1E2A4825AD4D7D2E
82 8888888888888888
83 cryptogram: E5B79C3D703D1BE1B26C2A999DB2975B
84 vehicle: matching cryptogram found
```

1

2 A.3. *Standard Transaction Cryptography Flow (with Reader Intent for Fast* 3 *Transaction)*

4 *Listing A-3: Standard Transaction Cryptography Flow (with Reader Intent for Fast Transaction)*

```
1 Vehicle key pair:
2    vehicle public key X: F47EB42A771052580C086EFDA3084AA3FF7A67CE23393A0373C63487DF1A63
3    vehicle public key Y: 7D1FB34B2D2E7D5C8F92097A0619B5C5CC6C5850AF74C019EBBEC4273358AA94
4    vehicle private key: 86B9E3843D949890FD50E49C8542DB575BAC41D344F17588DDAFE4535521CE55
5    Endpoint long term key pair:
6    endpoint public key X: 07B857B9B7F1147E20F4DBE6723CE5F46EF8670CBA20F56297F515C8265E4E42
7    endpoint public key Y: 5F1FC9B5DAFB62DAAFB5DC9AA6F8B2EDC1CDD43E20A614EF2F8703FA1459721C
8    endpoint private key: 13849B3560961053D985DA5E0FA05EAA99DD73EC50CC4A87029A24D7C331B
9
10 >>00A4040005AAAAAAAAAA00
11
12 endpoint: received SELECT command
13
14 <<5C0201009000
15
16 Generate ephemeral keys:
17    ephemeral public key X: F98CCA31651AD2E63266144B2450FD6081D8FEA8CEB826E1FB10E8034E932446
18    ephemeral public key Y: CAD19D201062DD1C7CB0BB293BF16A4BEFB2ED500977E7197E01F26906E39B5F
19    ephemeral private key: B0FBA5FB966FDD3BE4096FA65307AB0A7A3BB914625BBFD3CB57DAD9183E19CB
20
21 >>80800100635C020100874104F98CCA31651AD2E63266144B2450FD6081D8FEA8
22 CEB826E1FB10E8034E932446CAD19D201062DD1C7CB0BB293BF16A4BEFB2ED50
23 0977E7197E01F26906E39B5F4C10BF1C41268230AF76BFFE3E7C5D00CF4A4D08
24 88888888888888800
25
26 endpoint: received AUTH0 command
27 Generate ephemeral keys:
28    ephemeral public key X: 43D605526999F032E08F314F22EBCE051D1DAE53DC71F1C4D614B0337BB17F20
29    ephemeral public key Y: 3F95D4C06AB8966D2B9A0D3C4BC446DB9343EBF27F9EF811F242A37118AD4F10
30    ephemeral private key: E585C9EE89075F795452879AC38261ED0667C6396A34914DEE0681E8DC22A182
31    Key Derivation:
32    K: ACEDD14246C16AAF4561E177E567192454C06B2AAEEDB7E278980C25DD7994A2
33    endpoint ephemeral public key X: 43D605526999F032E08F314F22EBCE051D1DAE53DC71F1C4D614B0337BB17F20
34    vehicle ephemeral public key X: F98CCA31651AD2E63266144B2450FD6081D8FEA8CEB826E1FB10E8034E932446
```

```
35 | transaction identifier: BF1C41268230AF76BFFE3E7C5D00CF4A
36 | interface byte: 5E
37 | flag: 0100
38 | HKDF output: C23F571D20E08CCC6AA08367D7800DC4A20BB6F231B3EE2F007E8B952C4B95B0
39 | B0C367A2867DD0530C560C5FD167D18E017C2FA77D4C653C4794529B3F69B529
40 | Kcmac: C23F571D20E08CCC6AA08367D7800DC4
41 | Kenc: A20BB6F231B3EE2F007E8B952C4B95B0
42 | Kmac: B0C367A2867DD0530C560C5FD167D18E
43 | Krmac: 017C2FA77D4C653C4794529B3F69B529
44 | generate cryptogram:
45 |   Kcmac: C23F571D20E08CCC6AA08367D7800DC4
46 |   endpoint public key X: 07B857B9B7F1147E20F4DBE6723CE5F46EF8670CBA20F56297F515C8265E4E42
47 |   vehicle public key X: F47EB42A771052580C086EFDA3084AA3FF7A67CE23393A0373C63487DF1A63
48 |   vehicle identifier: 8888888888888888
49 |   transaction identifier: BF1C41268230AF76BFFE3E7C5D00CF4A
50 |   input of AES-CMAC: 000000000000000000000000000000003200008001F47EB42A771052580C086EFDA3084A
51 |   A3FF7A67CE23393A0373C63487DF1A6307B857B9B7F1147E20F4DBE6723CE5F4
52 |   6EF8670CBA20F56297F515C8265E4E42BF1C41268230AF76BFFE3E7C5D00CF4A
53 |   8888888888888888
54 |   cryptogram: BD75825ECE29A6B84ADA79D9BF719839
55 |
56 | <<86410443D605526999F032E08F314F22EBCE051D1DAE53DC71F1C4D614B0337B
57 | B17F203F95D4C06AB8966D2B9A0D3C4BC446DB9343EBF27F9EF811F242A37118
58 | AD4F109D10BD75825ECE29A6B84ADA79D9BF7198399000
59 |
60 | vehicle: no matching cryptogram found
61 | generate authentication hash:
62 |   sha256 hash input: 4D0888888888888888888862043D605526999F032E08F314F22EBCE051D1DAE53
63 |   DC71F1C4D614B0337BB17F208720F98CCA31651AD2E63266144B2450FD6081D8
64 |   FEA8CEB826E1FB10E8034E9324464C10BF1C41268230AF76BFFE3E7C5D00CF4A
65 |   9304415D9569
66 |   sha256 hash output: 9A2A933D4B90F5A9CFB0B5524E36B10D3669B91F2526F6C0FC2369BD98A327A0
67 |
68 | >>80810000429E40CCE7447AC8D0112C24AE4A261AF63EBA7B585126FFA4CE4C06
69 | 1D11D97B98151CB7D85BDCCA539D152B544B97647DD5CD38DCBDBD82EF93F5B5
70 | 796FFF3C2C0FD700
71 |
72 | endpoint: received AUTH1 command
73 | generate authentication hash:
74 |   sha256 hash input: 4D0888888888888888888862043D605526999F032E08F314F22EBCE051D1DAE53
75 |   DC71F1C4D614B0337BB17F208720F98CCA31651AD2E63266144B2450FD6081D8
76 |   FEA8CEB826E1FB10E8034E9324464C10BF1C41268230AF76BFFE3E7C5D00CF4A
77 |   9304415D9569
78 |   sha256 hash output: 9A2A933D4B90F5A9CFB0B5524E36B10D3669B91F2526F6C0FC2369BD98A327A0
79 | endpoint: successful signature verification
80 | Derive Kdh:
81 |   ephemeral public key X: F98CCA31651AD2E63266144B2450FD6081D8FEA8CEB826E1FB10E8034E932446
82 |   ephemeral public key Y: CAD19D201062DD1C7CB0BB293BF16A4BEFB2ED500977E7197E01F26906E39B5F
83 |   ephemeral private key: E585C9EE89075F795452879AC38261ED0667C6396A34914DEE0681E8DC22A182
84 |   transaction identifier: BF1C41268230AF76BFFE3E7C5D00CF4A
85 |   ecdh shared secret: 5A2E155294C7D0074DAE3E133D20A6DA60F0A15A02A3A909268BC15D160323FA
86 |   sha256 input: 5A2E155294C7D0074DAE3E133D20A6DA60F0A15A02A3A909268BC15D160323FA
87 |   00000001BF1C41268230AF76BFFE3E7C5D00CF4A
88 |   derived Kdh: 18B0CDC20B916B22D2E5D87FDA544D7BD809D171DA00E103A72DF0FF5E5CD185
89 |   Key Derivation:
90 |     Kdh: 18B0CDC20B916B22D2E5D87FDA544D7BD809D171DA00E103A72DF0FF5E5CD185
91 |   endpoint ephemeral public key X: 43D605526999F032E08F314F22EBCE051D1DAE53DC71F1C4D614B0337BB17F20
92 |   vehicle ephemeral public key X: F98CCA31651AD2E63266144B2450FD6081D8FEA8CEB826E1FB10E8034E932446
```



```
209 wrap response:  
210 plaintext:05AAAAAAAAA05BBBBBBBBBB  
211 ICV:8A8829B8A956A893BE53B4A0E050E8FC  
212 cleartext payload:05AAAAAAAAA05BBBBBBBB80000000  
213 MAC chaining value:8B6EC24F1A0591AAF0C1AB48C49029B5  
214 cipher|mac:56C5CA7C53338B7927DA2B6E8113FD5189CB0DF19EDD4B  
215  
216 <<56C5CA7C53338B7927DA2B6E8113FD5189CB0DF19EDD4B9000  
217  
218 unwrap response:  
219 cipher|mac:56C5CA7C53338B7927DA2B6E8113FD5189CB0DF19EDD4B  
220 plaintext:05AAAAAAAAA05BBBBBBBBBB
```

1 A.4. X.509 Schemes

2 A.4.1. X.509 Basic Certificate Schema

3 *Listing A-4: X.509 v3 schema*

```
1 Certificate ::= SEQUENCE  
2 {  
3   tbsCertificate     TBSCertificate,  
4   signatureAlgorithm AlgorithmIdentifier,  
5   signatureValue      BIT STRING  
6 }  
7  
8 TBSCertificate ::= SEQUENCE  
9 {  
10  version          [0] EXPLICIT Version DEFAULT v1,  
11  serialNumber      CertificateSerialNumber,  
12  signature         AlgorithmIdentifier,  
13  issuer            Name,  
14  validity          Validity,  
15  subject           Name,  
16  subjectPublicKeyInfo SubjectPublicKeyInfo,  
17  issuerUniqueID    [1] IMPLICIT UniqueIdentifier OPTIONAL,  
18  -- If present, version SHALL be v2 or v3  
19  subjectUniqueID   [2] IMPLICIT UniqueIdentifier OPTIONAL,  
20  -- If present, version SHALL be v2 or v3  
21  extensions         [3] EXPLICIT Extensions OPTIONAL  
22  -- If present, version SHALL be v3  
23 }  
24  
25 Version ::= INTEGER { v1(0), v2(1), v3(2) }  
26  
27 CertificateSerialNumber ::= INTEGER  
28  
29 Validity ::= SEQUENCE  
30 {  
31   notBefore    Time,  
32   notAfter     Time  
33 }  
34  
35 Time ::= CHOICE  
36 {  
37   utcTime      UTCTime,  
38   generalTime  GeneralizedTime  
39 }
```

```
40
41 UniquedIdentifier ::= BIT STRING
42
43 SubjectPublicKeyInfo ::= SEQUENCE
44 {
45   algorithm      AlgorithmIdentifier,
46   subjectPublicKey BIT STRING
47 }
48
49 Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
50
51 Extension ::= SEQUENCE
52 {
53   extnID      OBJECT IDENTIFIER,
54   critical    BOOLEAN DEFAULT FALSE,
55   extnValue   OCTET STRING
56           -- contains the DER encoding of an ASN.1 value
57           -- corresponding to the extension type identified
58           -- by extnID
59 }
60
61 AlgorithmIdentifier ::= SEQUENCE
62 {
63   algorithm      OBJECT IDENTIFIER,
64   parameters    ANY DEFINED BY algorithm OPTIONAL
65 }
66
67 Name ::= CHOICE { -- only one possibility for now --
68   rdnSequence RDNSequence }
69
70 RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
71
72 RelativeDistinguishedName ::= SET SIZE (1..MAX) OF AttributeTypeAndValue
73
74 AttributeTypeAndValue ::= SEQUENCE
75 {
76   type      AttributeType,
77   value     AttributeValue
78 }
79
80 AttributeType ::= OBJECT IDENTIFIER
81
82 AttributeValue ::= ANY -- DEFINED BY AttributeType
```

1

2 A.4.2. X.509 Key Usage Extension

3 *Listing A-5: X.509 Key Usage extension*

```
1 id-ce-keyUsage OBJECT IDENTIFIER ::= { id-ce 15 }
2
3 KeyUsage ::= BIT STRING
4 {
5   digitalSignature      (0),
6   nonRepudiation       (1), -- recent editions of X.509 have
7                           -- renamed this bit to contentCommitment
8   keyEncipherment      (2),
9   dataEncipherment     (3),
```

```
10 keyAgreement      (4),  
11 keyCertSign       (5),  
12 cRLSign          (6),  
13 encipherOnly     (7),  
14 decipherOnly     (8)  
15 }
```

1 A.4.3. X.509 Basic Constraints Extension

2 *Listing A-6: X.509 Basic Constraints extension*

```
1 id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }  
2  
3 BasicConstraints ::= SEQUENCE  
4 {  
5   cA           BOOLEAN DEFAULT FALSE,  
6   pathLenConstraint  INTEGER (0..MAX) OPTIONAL  
7 }
```

3 A.4.4. X.509 Authority Key Identifier Extension

4 *Listing A-7: X.509 Authority Key Identifier extension*

```
1 AuthorityKeyIdentifier ::= SEQUENCE  
2 {  
3   keyIdentifier        [0] KeyIdentifier      OPTIONAL,  
4   authorityCertIssuer [1] GeneralNames      OPTIONAL,  
5   authorityCertSerialNumber [2] CertificateSerialNumber  OPTIONAL  
6 }  
7  
8 KeyIdentifier ::= OCTET STRING
```

5 A.4.5. X.509 Subject Key Identifier Extension

6 *Listing A-8: X.509 Subject Key Identifier extension*

```
1 SubjectKeyIdentifier ::= KeyIdentifier  
2  
3 KeyIdentifier ::= OCTET STRING
```

7 A.4.6. X.509 ECDSA Signature

8 *Listing A-9: X.509 ECDSA Signature Format*

```
1 Ecdsa-Sig-Value ::= SEQUENCE  
2 {  
3   r   INTEGER,  
4   s   INTEGER  
5 }
```

9 A.4.7. Issuer and Verifier Rules

- 10 The following rules supersede the issuance and verification rules defined in RFC 5280 [3]. If no
11 specific rules are stated, the RFC 5280 [3] rules shall apply. In the following tables, “defined”
12 values refer to content of Listing 15-5, Listing 15-13, and Listing 15-16.
13 The device OEM Server shall ensure consistency of endpoint issuer CN format and instance
14 CA subject CN format (both either UTF8 or printableString).

- 1 The issuer CN format (either UTF8String or PrintableString) in the external CA certificate shall be the same as the subject CN format in the vehicle OEM root certificate.
- 3 The subject CN format (either UTF8String or PrintableString) in the external CA certificate shall be the same as the subject CN format in the device OEM root certificate.

5 *Table A-1: Issuer and Verifier Rules Common to External CA, Instance CA and Endpoint*

Item	Verifier Rule	Issuer Rule
tbsCertificate.version	Shall reject if not v3.	Shall issue only v3 certificates.
tbsCertificate.serialNumber	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Shall be compliant with RFC 5280 rules for conforming CAs.
tbsCertificate.signature.algorithm	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Shall contain defined OID in algorithm attribute. Shall not contain parameters attribute.
tbsCertificate.(issuer subject) rdnSequence path validation and issuer/subject name chaining	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Shall issue certificates chain with valid issuer/subject name chaining as per RFC 5280.
tbsCertificate.(issuer subject) rdnSequence	Shall accept multiple RelativeDistinguishedName.	Shall contain only 1 RelativeDistinguishedName.
tbsCertificate.(issuer subject) rdnSequence.[].[].type	Shall accept multiple AttributeTypeAndValue per RelativeDistinguishedName.	The RelativeDistinguishedName shall contain only 1 AttributeTypeAndValue.
tbsCertificate.(issuer subject) rdnSe-	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	OID shall be CommonName.
tbsCertificate.(issuer subject) rdnSequence.[].[].value item type	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Shall be PrintableString or UTF8String.
tbsCertificate.(issuer subject) rdnSequence.[].[].value item type	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Item type for issuer field in child certificate shall be the same type as subject of parent certificate.
tbsCertificate.(issuer subject) rdnSequence.[].[].value item content	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Content length shall not exceed 30 bytes.
tbsCertificate.validitytime item type	Shall accept UTCTime or GeneralizedTime time types independently of the represented date. The expiration of the endpoint certificate shall not be checked.	Shall be compliant with RFC 5280 rules for conforming CAs.
tbsCertificate.validitytime item content	Should not verify unless verifier has a trusted source of time.	Shall be compliant with RFC 5280 rules for conforming CAs.
tbsCertificate.SubjectPublicKeyInfo.alg orithm	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Shall contain defined OID in algorithm attribute. Shall contain defined OID in parameters attribute.
signatureAlgorithm.algorithm	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Shall contain defined OID in algorithm attribute. Shall not contain parameters attribute.
signatureValue	Shall be verified as per RFC 5280 rules.	Shall be compliant with RFC5280 rules.

1 *Table A-2: Issuer and Verifier Rules For Extensions Common to External CA, Instance CA and Endpoint*

Item	Verifier Rule	Issuer Rule
tbsCertificate.extensions handling of unrecognized extensions	Unrecognized extensions shall be handled as per RFC 5280 rules.	Extensions not listed in this specification shall be marked as non-critical.
tbsCertificate.extensions OIDs of unrecognized extensions	Shall reject if OIDs are not compliant with RFC 5280 OID format rules.	Shall only issue OIDs compliant with RFC 5280 format rules.
tbsCertificate.extensions handling of duplicate extensions	Shall reject if 2 extensions or more have the same OID.	Shall not issue with 2 or more extensions having the same OID.
tbsCertificate.extensions handling of extension order	Shall accept any extension order.	Extensions can be ordered arbitrarily.
tbsCertificate.extensions.[].AuthorityKeyIdentifier	Shall reject if AuthorityKeyIdentifier extension is not present. Shall accept if extension is marked as critical.	Shall contain a AuthorityKeyIdentifier extension with the defined extnValue. Shall be marked as non-critical.
tbsCertificate.extensions.[].KeyUsage	Shall reject if KeyUsage extension is not present. Shall reject if extnValue is different from the defined one. Shall reject if extension is not marked as critical.	Shall contain a KeyUsage extension with the defined extnValue. Shall be marked as critical.

2

3 *Table A-3: Issuer and Verifier Rules For Extensions Specific to External CA*

Item	Verifier Rule	Issuer Rule
tbsCertificate.extensions.[].externalCA	Shall reject if extension is not present. Shall reject if extnValue format or content is different from the defined one. Shall reject if extension is not marked as critical.	Shall contain an extension with the defined OID. Shall contain an extension with defined format and content. Shall be marked as critical.
tbsCertificate.extensions.[].SubjectKeyIdentifier	Shall reject if SubjectKeyIdentifier extension is not present. Shall also accept if the extension is marked as critical.	Shall contain a SubjectKeyIdentifier extension with the defined extnValue. Shall be marked as non-critical.
tbsCertificate.extensions.[].BasicConstraints	Shall reject if BasicConstraints extension is not present. Shall reject if extnValue format or content is not the defined one. Shall reject if extension is not marked as critical.	Shall contain a BasicConstraints extension. The extnValue attribute shall comply with the defined format and content. Shall be marked as critical.

4

5 *Table A-4: Issuer and Verifier Rules For Extensions Specific to Instance CA*

Item	Verifier Rule	Issuer Rule
tbsCertificate.extensions.[].instanceCA	Shall reject if extension is not present. Shall reject if extnValue format or content is different from the defined one. Shall reject if the extension is not marked as critical.	Shall contain an extension with the defined OID. Shall contain an extension with defined format and content. Shall be marked as critical.
tbsCertificate.extensions.[].SubjectKeyIdentifier	Shall reject if SubjectKeyIdentifier extension is not present. Shall accept if the extension is marked as critical.	Shall contain a SubjectKeyIdentifier extension with the defined extnValue. Shall be marked as non-critical.
tbsCertificate.extensions.[].BasicConstraints	Shall reject if BasicConstraints extension is not present. Shall reject if extnValue format or content is not the defined one. Shall reject if the extension is not marked as critical.	Shall contain a BasicConstraints extension. The extnValue attribute shall comply with the defined format and content. Shall be marked as critical.

6

1

Table A-5: Issuer and Verifier Rules For Extensions Specific to Endpoint

Item	Verifier Rule	Issuer Rule
tbsCertificate.extensions.[].endpoint	Shall reject if extension is not present. Shall reject if extnValue format or content is different from the defined one. Shall reject if the extension is not marked as critical.	Shall contain an extension with the defined OID. Shall contain an extension with defined format and content. Shall be marked as critical.
tbsCertificate.extensions.[].Basic-Constraints	Shall accept if BasicConstraints extension is absent. Shall reject if extnValue format or content is not the defined one. Shall reject if the extension is not marked as critical.	Shall contain a BasicConstraints extension. The extnValue attribute shall comply with the defined format and content. Shall be marked as critical.

2

3 Vehicle Public Key Certificate [K], intermediate CA, Vehicle OEM CA Certificate (signed by
4 Device OEM) [M], and vehicle OEM CA (sub)root certificates shall follow the same rules as the
5 external CA certificate with the following exceptions:

6 For the Vehicle OEM Root CA certificate:

7

Table A-6: Issuer and Verifier Rules for Vehicle OEM Root CA certificate

Item	Verifier Rule	Issuer Rule
tbsCertificate.extensions.[].-AuthorityKeyIdentifier	Shall not verify presence or content of this field.	May contain a AuthorityKeyIdentifier extension with the defined extnValue. If present it shall be marked as non-critical.
tbsCertificate.extensions.[].Key-Usage	Shall not verify presence or content of this field.	May contain a KeyUsage extension with the defined extnValue. If present it shall be marked as critical.

8 For the Vehicle Public Key Certificate

9

Table A-7: Issuer and Verifier Rules for Vehicle Public Key Certificate

Item	Verifier Rule	Issuer Rule
tbsCertificate.(issuer subject) rdnSequence.[].[].value item content	May verify only if the verifier has the capability to strictly follow RFC 5280 verification rules.	Content length shall not exceed 32 bytes.
tbsCertificate.extensions.[].SubjectKeyIdentifier	Shall accept if SubjectKeyIdentifier extension is not present. If the extension is present, it shall also be accepted if marked as critical.	Should contain a SubjectKeyIdentifier extension with the defined extnValue. If present, it shall be marked as non-critical.

10

11 Appendix B.

12 B.1. Instance Configurations

13 instance AID = See Section [15.3.2.1](#)

- 1 maximum number of endpoints = <Device OEM policy applies>
- 2 maximum number of Instance CAs = <Device OEM policy applies>
- 3 SHARING_PASSWORD_LENGTH = <vehicle OEM policy, length of 4 to 8 digits (including 4 and 8)>
- 4
- 5 The vehicle OEM should choose a suitable combination of password length and allowed number
- 6 of attempts to enter the password in the vehicle UI, following the recommendations in [\[39\]](#).

7 *B.2. Certificate Field Formats*

- 8 All certificates, except [C] and [N] (which are attestations) as specified in Section [16](#), shall
- 9 comply with X.509 formatting as per RFC 5280 [\[3\]](#).
- 10 All certificates shall use 256-bit ECC public keys.
- 11 All certificates shall append the server environment indicator to their subject CN (“-P”, “-E”, see
- 12 below) in addition to specific naming requirements for the subject CN as listed below.

13 B.2.1. Key Identifier Calculation

- 14 The key identifier corresponds to the field Subject Key Identifier in Section 4.2.1.2, method (1)
- 15 of RFC 5280 [\[3\]](#).

16 B.2.2. OID Assignment

- 17 Each certificate shall be identifiable by the following OID values:

18 Vehicle Public Key Certificates (leaf) [K]

19 1.3.6.1.4.1.41577.5.1

20 External CA Certificates (intermediate)[F]

21 1.3.6.1.4.1.41577.5.2

22 Instance CA Certificates (intermediate) [E]

23 1.3.6.1.4.1.41577.5.3

24 Endpoint Certificates (leaf)[H]

25 1.3.6.1.4.1.41577.5.4

26 Vehicle OEM Encryption Certificate (VehicleOEM.Enc.Cert)

27 1.3.6.1.4.1.41577.5.5

28 Vehicle OEM Signature Certificate (VehicleOEM.Sig.Cert)

29 1.3.6.1.4.1.41577.5.6

30 Device.Enc.Cert (optional as only the PK is sent in the trackKey request)

31 1.3.6.1.4.1.41577.5.7

32 Vehicle Intermediate Certificate

33 1.3.6.1.4.1.41577.5.8

34 Vehicle OEM CA Certificate [J]

35 1.3.6.1.4.1.41577.5.9

36 Vehicle OEM CA Certificate (Signed by Device OEM CA) [M] (optional as only used by some

37 implemenation)

1 1.3.6.1.4.41577.5.10

2 Extensions with the above OIDs shall be marked as critical.

3 B.2.3. Endpoint Certificate [H]

4 The **endpoint certificate** shall be formatted as follows:

5 Subject (CN):

6 DIGK.[OWNR, FRND].[identifier (4-byte printable ASCII)]|[optional: free text]

7 The subject field holds the endpoint_identifier value provided at Digital Key creation. The
8 endpoint_identifier and thus the subject shall be unique per Digital Key applet instance. The 4-
9 byte identifier is chosen by the device to ensure uniqueness. The free text may be added for
10 debugging or other identification purposes. It shall not contain dots (“.”) or equal-to (“=”)
11 signs. The subject field free text is separated by an equal to [=] sign from the identifier. The equal-to
12 (“=”) delimiter shall only be included if the optional free text is used.

13 Examples:

14 DIGK.OWNR.63K7=JOHN_APPLESEED

15 DIGK.FRND.Nd75=75924635

16 The issuer field equals the subject of Instance CA Certificate. The allowed character set per
17 element is A–Z, a–z, 0–9, coded in ASCII.

18 B.2.4. Instance CA Certificate [E]

19 The **Instance CA Certificate** shall be formatted as follows:

20 Subject (CN):

21 <Vehicle OEM Identifier>

22 The vehicle OEM identifier is defined in [35]. It shall have a length of 4 characters.

23 The issuer CN of the Instance CA certificate shall be equal to the subject CN of the external
24 Certificate. The format of text used in the issuer CN of the Instance CA certificate (either
25 PrintableString or UTF8String) shall also match the format of the text used in the subject CN of
26 the external Certificate.

27 B.2.5. External CA Certificate [F]

28 The external CA certificate issuer CN equals the subject CN of the vehicle OEM root certificate
29 and the subject CN equals the subject CN of the device OEM root certificate. The format of the
30 subject CN in the vehicle OEM root certificate and the device OEM root certificate shall follow
31 the same rules as outlined in section A.4.7.

32 The CN fields for subject and issuer in the external CA certificate shall not exceed 30 bytes in
33 DER coding.

34 B.2.6. Vehicle Public Key/Leaf Certificate [K]

35 The **Vehicle Public Key Certificate** [K] shall be formatted as follows:

36 The subject field of [K] shall contain the following information as a readable string:

37 V.[vehicle_oem].[datacenter/region].[brand].[vehicle_identifier]

38 All elements are separated by a “.” (dot).

1 The allowed character set per element is A–Z, a–z, 0–9, coded in ASCII.
2 [vehicle_oem] defines the Instance CA assignment. It shall match the Vehicle OEM Identifier in
3 Table 2-1 in [\[35\]](#).

4 Datacenter/region field contains information required by the server, e.g., to know which
5 endpoint/region and server environment to connect to in order to reach the vehicle. It shall be 3
6 bytes.

7 It is strongly recommended to have one worldwide endpoint per server environment with the
8 following definition:

- 9 • WWP: region=WorldWide, environment=PROD
- 10 • WWE: region=WorldWide, environment=E2E (Testing)

11 If required, possible regional values are:

- 12 • EUE: region=EUR, environment=E2E
- 13 • EUP: region=EUR, environment=PROD
- 14 • USE: region=USA, environment=E2E
- 15 • USP: region=USA, environment=PROD
- 16 • CHE: region=CHN, environment=E2E
- 17 • CHP: region=CHN, environment=PROD

18 The [brand] field represents the Vehicle Brand Name and should have a granularity that is used
19 to identify the app to be launched for this vehicle. It shall be 4 bytes.

20 The [vehicle_identifier] is used to associate vehicle and certificate, without using the VIN. The
21 vehicle_identifier requires 16 bytes of ASCII to represent the 8-byte value.

22 The following part of the subject field is used as ROUTING_INFORMATION for key sharing:
23 vehicle_oem.datacenter/region.brand

24 The vehicle_identifier is not part of the routing information.

25 B.2.7. Intermediate Vehicle Certificate

26 An intermediate certificate may be used to attest to the vehicle public key. The Vehicle OEM CA
27 attests to the intermediate certificate:

28 Vehicle OEM CA —> Intermediate Vehicle Certificate —> Vehicle Public Key

29 Subject CN is defined by the vehicle OEM. It shall terminate with an indication of the
30 environment in which the certificate is valid.

31 Subject (CN):

32 <Vehicle OEM defined>-INTERMEDIATE-<region>-<environment>

33 Example:

34 Vehicle OEM CA-INTERMEDIATE-WW-P

35 B.2.8. Vehicle OEM CA Certificate [J]

36 The Vehicle OEM CA certificate is trusted by an offline process by each device OEM. It is also
37 embedded in the vehicle.

1 Issuer CN and subject CN are defined by the vehicle OEM. Both shall terminate with an
2 indication of the environment in which the certificate is valid.

3 Subject (CN):

4 < Vehicle OEM defined>-<environment>

5 Example:

6 Vehicle OEM CA-P

7 **B.2.9. Vehicle OEM CA Certificate (signed by Device OEM) [M]**

8 The Vehicle OEM CA certificate (signed by Device OEM) issuer CN equals subject CN of the
9 Device OEM root certificate[D] and the subject CN equals the subject CN of the vehicle OEM
10 root certificate [J]. The CN field for subject and issuer in the Vehicle OEM CA certificate
11 (signed by Device) shall not exceed 30 bytes in DER coding.

12 **B.2.10. Vehicle OEM Privacy Encryption Certificate**

13 The Vehicle OEM privacy encryption certificate shall be formatted as follows:

14 Subject (CN):

15 <Vehicle OEM defined>-ENC-<region>-<environment>

16 Issuer:

17 Vehicle OEM-ENC-WW-P

18 **B.2.11. Vehicle OEM Signature Verification Certificate**

19 The Vehicle OEM signature certificate should be formatted as follows:

20 Subject (CN):

21 <Vehicle OEM defined>-TERM-<region>-<environment>

22 Example:

23 Vehicle OEM-TERM-WW-P

24 **B.2.12. Validity Period**

25 The fields “not before” and “not after” of the certificates shall be set to the following values:

26 **External CA Certificate, issued by Vehicle OEM**

27 not before = date and time of certificate creation

28 not after = <Vehicle OEM policy applies>

29 **Instance CA Certificate [E]**

30 not before = date and time of certificate creation

31 not after = <device and vehicle OEM policies apply>

32 The expiration of the Instance CA Certificate shall be checked by the vehicle, the owner device,
33 and the key tracking server.

1 **Endpoint Certificate**

2 not before = issuance date and time

3 not after = 99991231235959Z (as defined in [3])

4 The expiration date of the endpoint certificate should not be checked anywhere in the system.

5 *Note:* The validity definition in the friend's attestation package determines the lifetime of the key
6 in the system.

7 **Vehicle OEM Privacy Encryption Certificate**

8 not before = issuance date

9 not after = <vehicle OEM policy applies>

10 **Vehicle OEM Signature Certificate**

11 not before = issuance date

12 not after = <vehicle OEM policy applies>

13 The undefined expiration date value 99991231235959Z shall be accepted by the framework and
14 applet implementation.

15

16 **Appendix C.**

17 *C.1. External CA Certificate*

18 The following examples of certificate for illustration purpose using a “mock” environment
19 (“-M”). The certificates may vary within the RFC5280 limits, depending on vehicle OEM
20 issuance policies.

```
21      Certificate:  
22       Data:  
23       Version: 3 (0x2)  
24       Serial Number:  
25       42:76:cb:ad:b3:1a:8c:4a:72:e0:1c:5a:5c:ce:59:6f:67:42:d9:c5  
26       Signature Algorithm: ecdsa-with-SHA256  
27       Issuer: CN=MOCK-CAR-ROOT-CA-E  
28       Validity  
29       Not Before: Jan 1 00:00:00 2019 GMT  
30       Not After : Jan 1 00:00:00 2049 GMT  
31       Subject: CN=Device OEM Root Certificate  
32       Subject Public Key Info:  
33       Public Key Algorithm: id-ecPublicKey  
34       Public-Key: (256 bit)  
35       pub:
```

```
1          04:86:86:92:e0:41:b2:25:58:39:c7:85:62:03:b2:
2          d9:ae:fd:28:5c:f9:87:94:ec:72:69:37:cb:58:87:
3          63:28:91:d8:99:4a:85:c0:29:9a:d3:2d:f8:09:4d:
4          75:55:51:56:25:57:5e:74:1b:0d:82:42:cc:08:b2:
5          6a:c7:de:43:5e
6          ASN1 OID: prime256v1
7          NIST CURVE: P-256
8          X509v3 extensions:
9          X509v3 Authority Key Identifier:
10         key-
11        id:23:18:E5:56:71:F0:8E:AE:21:21:42:A8:17:72:0F:B8:17:EE:93:BF
12
13         X509v3 Subject Key Identifier:
14         65:17:11:D3:7B:96:37:00:21:3F:F2:8C:9C:4F:55:02:E4:D8:51:76
15         X509v3 Basic Constraints: critical
16         CA:TRUE, pathlen:1
17         1.3.6.1.4.1.41577.5.2: critical
18         .....
19         X509v3 Key Usage: critical
20         Certificate Sign
21         Signature Algorithm: ecdsa-with-SHA256
22         30:44:02:20:0c:54:51:3f:a1:cd:04:82:dc:49:13:f5:37:4c:
23         fa:5e:7e:4c:40:61:86:55:0b:12:3a:71:93:a1:22:22:2a:69:
24         02:20:04:2c:2f:06:65:3d:68:a2:81:cf:65:bc:57:85:b0:ca:
25         34:91:8b:51:18:83:59:8b:91:1e:58:89:a2:bd:cd:e1
26
```

27 C.2. Vehicle OEM Intermediate Certificate

```
28         Certificate:
29         Data:
30         Version: 3 (0x2)
31         Serial Number:
32         6d:da:2b:a0:69:c7:55:5a:db:84:a3:ee:dd:a3:83:fc:8a:33:b3:97
33         Signature Algorithm: ecdsa-with-SHA256
34         Issuer: CN=MOCK-CAR-ROOT-CA-E
35         Validity
36         Not Before: Jan 1 00:00:00 2019 GMT
37         Not After : Jan 1 00:00:00 2049 GMT
38         Subject: CN=MOCK-CAR-INTERMEDIATE-US-E
```

```
1      Subject Public Key Info:  
2          Public Key Algorithm: id-ecPublicKey  
3              Public-Key: (256 bit)  
4                  pub:  
5                      04:84:22:42:f6:18:2b:a1:c1:13:8d:32:b7:7f:b9:  
6                          f7:f3:7b:70:03:4b:9f:04:44:3a:5b:ea:3c:18:8b:  
7                              ea:db:36:49:0a:7e:95:f9:1a:4c:16:2a:cf:c3:40:  
8                                  1c:3a:4f:4e:5a:59:25:1d:45:24:3a:c8:54:4a:66:  
9                                      5c:b9:51:42:2f  
10             ASN1 OID: prime256v1  
11             NIST CURVE: P-256  
12             X509v3 extensions:  
13                 X509v3 Authority Key Identifier:  
14                     key-  
15             id:23:18:E5:56:71:F0:8E:AE:21:21:42:A8:17:72:0F:B8:17:EE:93:BF  
16  
17                 X509v3 Subject Key Identifier:  
18                     E6:97:50:57:10:09:13:C3:AB:07:29:D4:53:3F:4E:11:29:9A:34:83  
19                 X509v3 Basic Constraints: critical  
20                     CA:TRUE, pathlen:0  
21                     1.3.6.1.4.1.41577.5.8: critical  
22                         0....  
23                 X509v3 Key Usage: critical  
24                     Certificate Sign  
25             Signature Algorithm: ecdsa-with-SHA256  
26                     30:45:02:20:08:b4:7d:06:3f:c8:f0:27:95:5d:5d:99:62:ee:  
27                         92:91:5e:79:5d:99:4a:4b:5a:a7:6a:f4:2b:6c:16:1d:8c:85:  
28                         02:21:00:d8:34:99:b3:49:3c:5b:ee:55:39:98:0b:64:9f:58:  
29                         f1:43:54:e6:04:de:a3:9d:22:b3:cc:01:ad:a7:e2:40:04
```

30 C.3. Vehicle OEM Key/Leaf Certificate

```
31     Certificate:  
32         Data:  
33             Version: 3 (0x2)  
34             Serial Number:  
35                 0d:7a:48:40:4c:a2:24:14:45:85:d1:64:51:94:80:7b:ea:69:79:22  
36             Signature Algorithm: ecdsa-with-SHA256  
37             Issuer: CN=MOCK-CAR-INTERMEDIATE-US-E  
38             Validity
```

```
1      Not Before: Jan 1 00:00:00 2019 GMT
2      Not After : Jan 1 00:00:00 2049 GMT
3      Subject: CN=V.MOCK.USE.BRND.1111111111111111
4      Subject Public Key Info:
5          Public Key Algorithm: id-ecPublicKey
6              Public-Key: (256 bit)
7                  pub:
8                      04:1c:84:2b:af:21:cc:2b:28:3b:83:03:6b:a5:26:
9                          dd:bc:e9:c2:01:f6:3e:80:dc:73:50:88:c4:f3:ca:
10                         90:ff:f2:a4:08:ba:74:4f:78:b6:68:c8:8b:63:d0:
11                         90:a3:e3:48:d2:4d:29:d1:53:f5:31:15:e0:8a:31:
12                         4b:ff:d6:86:01
13          ASN1 OID: prime256v1
14          NIST CURVE: P-256
15          X509v3 extensions:
16              X509v3 Authority Key Identifier:
17                  key-
18          id:E6:97:50:57:10:09:13:C3:AB:07:29:D4:53:3F:4E:11:29:9A:34:83
19
20          X509v3 Subject Key Identifier:
21              57:11:1D:D5:E7:5E:24:75:AB:5F:07:BC:96:FB:69:89:FB:7D:0E:AA
22          X509v3 Basic Constraints: critical
23              CA:FALSE
24              1.3.6.1.4.1.41577.5.1: critical
25              0....
26          X509v3 Key Usage: critical
27              Digital Signature
28          Signature Algorithm: ecdsa-with-SHA256
29              30:44:02:20:48:95:c3:bb:16:d0:32:bb:15:c3:0e:2d:50:22:
30              cd:f8:78:a1:4e:87:6c:6b:08:71:09:12:91:fe:48:ba:cf:12:
31              02:20:43:17:ca:c5:d4:4c:67:53:52:7a:f9:d0:64:cb:fb:83:
32              4e:01:78:a0:b5:53:1f:af:83:3d:38:9b:f9:64:ed:b3
```

33 C.4. Vehicle OEM Privacy Encryption Certificate

```
34      Certificate:
35          Data:
36              Version: 3 (0x2)
37              Serial Number:
38                  7f:86:4e:b6:84:72:d0:52:26:47:92:9d:1e:ab:ae:e4:ef:74:bd:f9
```

```
1     Signature Algorithm: ecdsa-with-SHA256
2     Issuer: CN=MOCK-CAR-ROOT-CA-E
3     Validity
4         Not Before: Jan 1 00:00:00 2019 GMT
5         Not After : Jan 1 00:00:00 2049 GMT
6     Subject: CN=MOCK-CAR-ENC-US-E
7     Subject Public Key Info:
8         Public Key Algorithm: id-ecPublicKey
9         Public-Key: (256 bit)
10        pub:
11            04:2b:1d:13:34:c9:d3:16:1c:ca:d0:f5:b9:22:6b:
12                ca:df:ad:5f:c9:b7:83:b2:7f:4b:51:34:84:b7:76:
13                2e:e6:35:33:4d:a4:dc:9b:c7:54:92:3e:21:cb:7b:
14                22:ae:f7:0b:ca:16:6a:12:f0:35:f3:f2:d5:77:aa:
15                a3:b4:8b:d5:f5
16        ASN1 OID: prime256v1
17        NIST CURVE: P-256
18    X509v3 extensions:
19        X509v3 Authority Key Identifier:
20            key-
21            id:23:18:E5:56:71:F0:8E:AE:21:21:42:A8:17:72:0F:B8:17:EE:93:BF
22
23        X509v3 Subject Key Identifier:
24            C3:6C:36:D4:97:A1:31:5B:BB:F2:17:10:A3:E9:6D:12:4D:8D:07:5C
25        X509v3 Basic Constraints: critical
26            CA:FALSE
27            1.3.6.1.4.11577.5.5: critical
28            0....
29        X509v3 Key Usage: critical
30            Key Agreement, Encipher Only
31        Signature Algorithm: ecdsa-with-SHA256
32            30:46:02:21:00:f0:f2:bd:59:75:73:8b:60:3f:a9:a4:2d:7d:
33            70:23:4e:73:5a:63:73:59:25:d4:fe:42:4b:33:68:a4:64:5a:
34            e5:02:21:00:b0:df:b3:40:2f:7e:0f:2b:e3:81:43:f8:c5:b3:
35            b1:56:33:4e:7a:87:30:7c:73:64:4d:ad:03:a6:23:01:87:a0
```

36 *C.5. Vehicle OEM Signature Verification Certificate*

```
37    Certificate:
38    Data:
```

```
1      Version: 3 (0x2)
2      Serial Number:
3          0b:30:c4:1b:39:98:0a:44:41:4b:36:54:55:4f:fc:b5:28:8f:11:cf
4      Signature Algorithm: ecdsa-with-SHA256
5      Issuer: CN=MOCK-CAR-ROOT-CA-E
6      Validity
7          Not Before: Jan 1 00:00:00 2019 GMT
8          Not After : Jan 1 00:00:00 2049 GMT
9      Subject: CN=MOCK-CAR-TERM-US-E
10     Subject Public Key Info:
11         Public Key Algorithm: id-ecPublicKey
12             Public-Key: (256 bit)
13             pub:
14                 04:62:9e:51:60:2d:98:4c:e3:ea:77:ae:93:6f:e9:
15                 75:7a:34:a9:16:25:99:81:6a:12:df:17:f4:85:84:
16                 ad:77:ee:81:c7:d4:7f:8e:2d:50:29:27:4e:58:32:
17                 ac:69:0a:52:0d:20:22:fb:84:6a:75:c6:a1:fd:e4:
18                 45:ef:d6:a9:b0
19             ASN1 OID: prime256v1
20             NIST CURVE: P-256
21     X509v3 extensions:
22         X509v3 Authority Key Identifier:
23             key-
24             id:23:18:E5:56:71:F0:8E:AE:21:21:42:A8:17:72:0F:B8:17:EE:93:BF
25
26         X509v3 Subject Key Identifier:
27             29:F8:B9:AE:B8:4A:19:76:CF:90:A6:61:D3:0F:49:24:34:6E:31:EC
28         X509v3 Basic Constraints: critical
29             CA:FALSE
30             1.3.6.1.4.1.41577.5.6: critical
31             0....
32         X509v3 Key Usage: critical
33             Digital Signature
34         Signature Algorithm: ecdsa-with-SHA256
35             30:45:02:20:3d:d9:06:f8:f6:c0:be:c1:af:88:3c:7f:97:7d:
36             ad:7c:ad:ee:e7:62:09:71:7f:8f:4d:7c:cd:1f:19:d1:85:70:
37             02:21:00:89:bc:ad:9a:60:8a:b9:49:85:97:9e:d8:e1:5c:25:
38             f7:c9:55:82:7e:9a:b7:18:9c:5a:2e:ed:d5:fe:d0:e8:c9
```

1 **Appendix D. Owner Pairing Test Vectors [To be updated]**

2 *D.1. Derivation of z0, z1 with Scrypt*

3 Computed by the Vehicle OEM Server as well as the device.

4 **Parameters**

5 **Password:** "pleaseletmein" (ASCII, 13 bytes)
6 **Salt:** "yellowsubmarines" (ASCII, 16 bytes)
7 **Nscrypt:** 32768 (cost parameter)
8 **r:** 8 (block size)
9 **p:** 1 (parallelization parameter)
10 **dkLen:** 80 (output length)

11 **Results**

12 **dk:** (full 80 bytes)
13 6408161bbc64a41827cf072c62efdae535739e51
14 3ad3050e66a9f53eb69c15bb3c220f0acb5f7f02
15 dd008ce70d974431369ef8e569dee33977fa9fd2
16 e13a55c8c80c244a05559264b0498c1f5216f327
17 **z0:** (left 40 bytes)
18 6408161bbc64a41827cf072c62efdae535739e51
19 3ad3050e66a9f53eb69c15bb3c220f0acb5f7f02
20 **z1:** (right 40 bytes)
21 dd008ce70d974431369ef8e569dee33977fa9fd2
22 e13a55c8c80c244a05559264b0498c1f5216f327

23 *D.2. Computation of w0, w1*

24 w0 and w1 computed per FIPS 186-4, B.5.1 Per-Message Secret Number Generation
25 Using Extra Random Bits.

26 $w_0 = (z_0 \bmod (n - 1)) + 1$

27 $w_1 = (z_1 \bmod (n - 1)) + 1$

28 With n being the order of base point G as defined for NIST P-256.

29 **Parameters**

30 **n:** (32 bytes)
31 ffffffff00000000ffffffffffffffff
32 bce6faada7179e84f3b9cac2fc632551
33 **z0:** (40 bytes)
34 6408161bbc64a41827cf072c62efdae535739e51
35 3ad3050e66a9f53eb69c15bb3c220f0acb5f7f02
36 **z1:** (40 bytes)
37 dd008ce70d974431369ef8e569dee33977fa9fd2

```
1      e13a55c8c80c244a05559264b0498c1f5216f327
2      Results
3      w0: (32 bytes)
4          e433ab43428320b24fab82f915d1db11 4acd72f8a4bf4fbf3c712b94bcc2f013
5      w1: (32 bytes)
6          44363d157f471221b1e75e596ff4714a
7          712b9578301665d84ec17004952523a8
```

8 *D.3. Computation of L*

9 Only computed on the Vehicle OEM Server.

10 $L = w1 \times G$

11 L is the result of scalar multiplication on curve NIST P-256. G is the base point as defined for
12 NIST P-256.

13 **Parameters**

```
14      w1: (32 bytes)
15          44363d157f471221b1e75e596ff4714a
16          712b9578301665d84ec17004952523a8
17      G.x: (32 bytes)
18          6b17d1f2e12c4247f8bce6e563a440f2 77037d812deb33a0f4a13945d898c296
19      G.y: (32 bytes)
20          4fe342e2fe1a7f9b8ee7eb4a7c0f9e16 2bce33576b315ececbb6406837bf51f5
```

21 **Results**

```
22      L.x: (32 bytes)
23          ff69eb6086938b3cce2c9e64dcaceala 925918e75e8c17948d316322d370123f
24      L.y: (32 bytes)
25          69132aed7398919e6e6614f7627b0a54
26          060c5a8c0d93d2754166ab10fea6a8ff
```

27 *D.4. Computation of X*

28 Computed by the device.

29 $X = x \times G + w0 \times M$

30 x is a random scalar. G is the base point as defined for NIST P-256 and M a fixed point as
31 defined by the spec.

32 **Parameters**

```
33      x: (32 bytes)
34          000102030405060708090a0b0c0d0e0f
35          101112131415161718191a1b1c1d1e1f
36      w0: (32 bytes)
```

```
1      e433ab43428320b24fab82f915d1db11
2      4acd72f8a4bf4fbf3c712b94bcc2f013
3      G.x: (32 bytes)
4      6b17d1f2e12c4247f8bce6e563a440f2
5      77037d812deb33a0f4a13945d898c296
6      G.y: (32 bytes)
7      4fe342e2fe1a7f9b8ee7eb4a7c0f9e16
8      2bce33576b315eccebb6406837bf51f5
9      M: (65 bytes)
10     04
11     886e2f97ace46e55ba9dd7242579f299
12     3b64e16ef3dcab95af497333d8fa12f
13     5ff355163e43ce224e0b0e65ff02ac8e
14     5c7be09419c785e0ca547d55a12e2d20

15      Results
16      xG.x: (32 bytes)
17      7a593180860c4037c83c12749845c8ee
18      1424dd297fadcb895e358255d2c7d2b2
19      xG.y: (32 bytes)
20      a8ca25580f2626fe579062ff1b99ff91 c24a0da06fb32b5be20148c9249f5650
21      w0M.x: (32 bytes)
22      0d556afe7d4cf1b87a9baa429ab9fb57 5de4f36d412cbd7d0dc095779590f5aa
23      w0M.y: (32 bytes)
24      5f941aae1286700e4dab2ff38b4eeb2d 6ceb5dd3f7072bc84ee74360d0f7ad0c
25      x.x: (32 bytes)
26      f44555207a617fd90900dba5c8e6f81e ddbd87590873a63b9057dda9f138dbc1
27      x.y: (32 bytes)
28      6f453195f6452ce71d399052435952b8
29      9a10b927435574f5e3707eae031c40e0
```

30 *D.5. Computation of Y*

31 Computed by the vehicle.

32
$$Y = y \times G + w_0 \times N$$

33 y is a random scalar. G is the base point as defined for NIST P-256 and N a fixed point as
34 defined by the spec.

35 **Parameters**

```
36      y: (32 bytes)
37      f1e1d1c1b1a191817161514131211101
38      f0e0d0c0b0a090807060504030201000
```

```
1      w0: (32 bytes)
2      e433ab43428320b24fab82f915d1db11
3      4acd72f8a4bf4fbf3c712b94bcc2f013
4      G.x: (32 bytes)
5      6b17d1f2e12c4247f8bce6e563a440f2
6      77037d812deb33a0f4a13945d898c296
7      G.y: (32 bytes)
8      4fe342e2fe1a7f9b8ee7eb4a7c0f9e16
9      2bce33576b315eccecb6406837bf51f5
10     N: (65 bytes)
11     04
12     d8bb6c639c62937b04d997f38c37707
13     19c629d7014d49a24b4f98baa1292b49
14     07d60aa6bfa de45008a636337f5168c6
15     4d9bd36034808cd564490b1e656edbe7

16     Results
17     yG.x: (32 bytes)
18     d05d3f33d7a67710f70275600e2905e4
19     4f436e3331a1f479f7de0b650f679d12
20     yG.y: (32 bytes)
21     1c404aa84e6f75fb75d0526ff48e004b 4ea689d7fc619a05e0ba37dfbef340bf
22     w0N.x: (32 bytes)
23     4f8cd0a6e01386dac1b2e1af01e1e56a 65d7beccfc b7ed39437308f4bd99d3b6
24     w0N.y: (32 bytes)
25     e89a7da518b37a32f9acfe497da5fd16 05b80820b4ce92406569de793811d938
26     Y.x: (32 bytes)
27     b6fdaf3f6949869d68f667108b75e4ce 74847e8953d1e3c6aae21699e8027211
28     Y.y: (32 bytes)
29     c2d9b2b2a906cc7ea7020715dec44e95 659e3fc8994f635b95e7c9ea5c362cbe

30     D.6. Computation of Z, V (by device)
31     Z, V are keys shared by the device and vehicle.
32     T = Y - w0 × N
33     Z = x × T
34     V = w1 × T

35     Parameters
36     x: (32 bytes)
37     000102030405060708090a0b0c0d0e0f
```

```
1          101112131415161718191a1b1c1d1e1f
2          w0: (32 bytes)
3          e433ab43428320b24fab82f915d1db11
4          4acd72f8a4bf4fbf3c712b94bcc2f013
5          w1: (32 bytes)
6          44363d157f471221b1e75e596ff4714a
7          712b9578301665d84ec17004952523a8
8          y: (65 bytes)
9          04
10         b6fdaf3f6949869d68f667108b75e4ce
11         74847e8953d1e3c6aae21699e8027211
12         c2d9b2b2a906cc7ea7020715dec44e95
13         659e3fc8994f635b95e7c9ea5c362cbe
14         N: (65 bytes)
15         04
16         d8bb6c639c62937b04d997f38c37707
17         19c629d7014d49a24b4f98baa1292b49
18         07d60aa6bfafe45008a636337f5168c6
19         4d9bd36034808cd564490b1e656edbe7

20         Results
21         T.x: (32 bytes)
22         d05d3f33d7a67710f70275600e2905e4
23         4f436e3331a1f479f7de0b650f679d12
24         T.y: (32 bytes)
25         1c404aa84e6f75fb75d0526ff48e004b
26         4ea689d7fc619a05e0ba37dfbef340bf
27         Z.x: (32 bytes)
28         89af176e8122e67c00dbcea089bc4993 5634132b1b226030d2b14f16b3e73351
29         Z.y: (32 bytes)
30         c2254b1b62477fdc976379f5ae7c57c6 ac2b31ef9a032c33e4677c5c3acbb1d3
31         V.x: (32 bytes)
32         2f229f13e1ebfb6442a67eebdafb23b2 f6e656597384035a8a1e50ad95d24211
33         V.y: (32 bytes)
34         339e90a669dd8a56fb2524fb6e6c784b 89019c1130c2def98143fb46dcc507d2
```

35 *D.7. Computation of Z, V (by vehicle)*

36 Z, V are keys shared by the device and vehicle.

37 $Z = y \times (X - w_0 \times M)$

38 $V = y \times L$

```
1      Parameters
2      y: (32 bytes)
3      f1e1d1c1b1a191817161514131211101
4      f0e0d0c0b0a090807060504030201000
5      w0: (32 bytes)
6      e433ab43428320b24fab82f915d1db11
7      4acd72f8a4bf4fbf3c712b94bcc2f013
8      x: (65 bytes)
9      04
10     f44555207a617fd90900dba5c8e6f81e
11     ddbd87590873a63b9057dda9f138dbc1
12     6f453195f6452ce71d399052435952b8
13     9a10b927435574f5e3707eae031c40e0
14     M: (65 bytes)
15     04
16     886e2f97ace46e55ba9dd7242579f299
17     3b64e16ef3dcab95af497333d8fa12f
18     5ff355163e43ce224e0b0e65ff02ac8e
19     5c7be09419c785e0ca547d55a12e2d20
20     L: (65 bytes)
21     04
22     ff69eb6086938b3cce2c9e64dcacea1a
23     925918e75e8c17948d316322d370123f
24     69132aed7398919e6e6614f7627b0a54
25     060c5a8c0d93d2754166ab10fea6a8ff

26     Results
27     z.x: (32 bytes)
28     89af176e8122e67c00dbcea089bc4993
29     5634132b1b226030d2b14f16b3e73351
30     z.y: (32 bytes)
31     c2254b1b62477fdc976379f5ae7c57c6
32     ac2b31ef9a032c33e4677c5c3acbb1d3
33     v.x: (32 bytes)
34     2f229f13e1ebfb6442a67eebdafb23b2
35     f6e656597384035a8a1e50ad95d24211
36     v.y: (32 bytes)
37     339e90a669dd8a56fb2524fb6e6c784b
38     89019c1130c2def98143fb46dcc507d2
```

1 **D.8. Derivation of K, CK, SK**

2 K is the secret shared by the device and vehicle.

3 $K = \text{SHA-256}(\text{len}(X) \parallel X \parallel \text{len}(Y) \parallel Y \parallel \text{len}(Z) \parallel Z \parallel \text{len}(V) \parallel V \parallel \text{len}(w_0) \parallel w_0)$

4 len(x) denotes the length of a string in bytes, represented as an 8-byte little-endian number.

5 **Parameters**

6 **w0:** (32 bytes)

7 e433ab43428320b24fab82f915d1db11

8 4acd72f8a4bf4fbf3c712b94bcc2f013

9 **len(w0):** (8 bytes)

10 2000000000000000

11 **X:** (65 bytes)

12 04

13 f44555207a617fd90900dba5c8e6f81e

14 ddbd87590873a63b9057dda9f138dbc1

15 6f453195f6452ce71d399052435952b8

16 9a10b927435574f5e3707eae031c40e0

17 **Y:** (65 bytes)

18 04

19 b6fdaf3f6949869d68f667108b75e4ce

20 74847e8953d1e3c6aae21699e8027211

21 c2d9b2b2a906cc7ea7020715dec44e95

22 659e3fc8994f635b95e7c9ea5c362cbe

23 **Z:** (65 bytes)

24 04

25 89af176e8122e67c00dbcea089bc4993

26 5634132b1b226030d2b14f16b3e73351

27 c2254b1b62477fdc976379f5ae7c57c6

28 ac2b31ef9a032c33e4677c5c3acbb1d3

29 **V:** (65 bytes)

30 04

31 2f229f13e1ebfb6442a67eebdafb23b2

32 f6e656597384035a8a1e50ad95d24211

33 339e90a669dd8a56fb2524fb6e6c784b

34 89019c1130c2def98143fb46dcc507d2

35 **len(X/Y/Z/V):** (8 bytes)

36 4100000000000000

37 **Results**

38 **K:** (32 bytes)

1 381fc44894aedc0fd37257fdca763ee4
2 9f695017ba5e1df74a1b8bbf27fe1e0d
3 **CK:** (16 bytes)
4 381fc44894aedc0fd37257fdca763ee4
5 **SK:** (16 bytes)
6 9f695017ba5e1df74a1b8bbf27fe1e0d

7 *D.9. Derivation of Evidence Keys K1, K2*

8 K1 and K2 are the MAC keys used for key confirmation.
9 HKDF-SHA-256 is used as the KDF, defined by IETF RFC5869.

10 **Parameters**
11 **L:** 32
12 IKM/CK: (16 bytes)
13 381fc44894aedc0fd37257fdca763ee4
14 **Info:** "ConfirmationKeys" (ASCII, 16 bytes)
15 5b020101 (Agreed SPAKE2+ protocol version, 4 bytes)
16 5c0401010100 (Tag 5C of the SPAKE2+ REQUEST command, 6 bytes)

17 **Results**

18 **K1:** (16 bytes)
19 ab667cae7ffe27505265d0f2026e146ea
20 **K2:** (16 bytes)
21 af679bf88b734e1cb7cd0243fb21a589

22 Note that this calculation is an example of key derivation that uses an undefined protocol version
23 0101. Its intent is to show the key derivation algorithm in principle.

24 *D.10. Computation of Evidences M1, M2*

25 M1 and M2 are the MACs used for key confirmation.
26 M1 = CMAC(K1, X)
27 M2 = CMAC(K2, Y)

28 **Parameters**
29 **K1:** (16 bytes)
30 ab667cae7ffe27505265d0f2026e146ea
31 **K2:** (16 bytes)
32 af679bf88b734e1cb7cd0243fb21a589
33 **X:** (65 bytes)
34 04
35 f44555207a617fd90900dba5c8e6f81e
36 ddbd87590873a63b9057dda9f138dbc1

1 6f453195f6452ce71d399052435952b8
2 9a10b927435574f5e3707eae031c40e0
3 **Y:** (65 bytes)
4 04
5 b6fdaf3f6949869d68f667108b75e4ce
6 74847e8953d1e3c6aae21699e8027211
7 c2d9b2b2a906cc7ea7020715dec44e95
8 659e3fc8994f635b95e7c9ea5c362cbe

9 **Results**

10 **M1:** (16 bytes)
11 110d49f8c5a896e11d4dde4c3b9704d2
12 **M2:** (16 bytes)
13 23d1a618ad3acbfd7a9bd19fd1737107

14 *D.11. Derivation of System Keys*

15 HKDF-SHA-256 is used as the KDF, defined by IETF RFC 5869.

16 **Parameters**

17 **L:** 64
18 **IKM/SK:** (16 bytes)
19 9f695017ba5e1df74a1b8bbf27fe1e0d
20 **Info:** "SystemKeys" (ASCII, 10 bytes)

21 **Results**

22 **Kenc:** (16 bytes)
23 161886cb9ae7403d8dbccfe36b8a0426
24 **Kmac:** (16 bytes)
25 6387ba65479cb7eb9df97bd48ac33159
26 **Krmac:** (16 bytes)
27 41677fb6398459199f1e569760df91c1
28 **LONG_TERM_SHARED_SECRET:** (16 bytes)
29 5c4e19da553524e386fa1eca91e8ad0e

30 **Appendix E. NFC-F Support [WCC1]**

31 *E.1. Introduction*

32 This Appendix defines a protocol to perform the contactless transactions defined in this
33 document over NFC-F.

34 Support for NFC-F is an optional feature of the Digital Key protocol. The requirements in this
35 appendix apply only if an implementation supports this option.

1 Implementation may include support for NFC-F to be able to use NFC-F devices for holding
2 Digital Keys or if the contactless transactions defined in this document are used as part of a
3 larger NFC-F transaction.

4 The protocol in this appendix defines how the device exchanges APDUs with the vehicle over
5 NFC-F. The APDU-based message exchanges defined in this document are not modified in any
6 way when run over NFC-F. Support for NFC-F only affects the contactless interface; it has no
7 impact on the device-internal interfaces to communicate between the Digital Key applet and
8 Digital Key framework or on the communication with the Device OEM Server or Vehicle OEM
9 Server.

10 The following definitions are specific to this appendix:

- 11 • **Protocol Error:** Receiving a response with a wrong syntax or receiving a correct
12 response when it is not expected.
- 13 • **Timeout Error:** No response has been received within the defined waiting time.
- 14 • **Transmission Error:** Error caused by the receipt of an incorrect frame. This includes the
15 case that the CRC in the NFC-F frame is not correct.

16 The following abbreviations are specific to this appendix:

17 RW_ACK	Acknowledge send from vehicle to device
18 DEVICE_ACK	Acknowledge send from device to vehicle
19 C-APDU	Command APDU as defined in [1]
20 fc	Carrier Frequency
21 NACK	Negative acknowledgement
22 NFC-F	NFC Type F RF Technology
23 NFCID2	NFC-F identifier as defined in [DIGITAL].
24 R-APDU	Response APDU as defined in [1]
25 RWT	Response waiting time
26 RWI	Response waiting time integer

27 *E.2. Device Requirement*

28 To support NFC-F, the following requirements apply on the device side:

- 29 • The protocol defined in this appendix shall be implemented by the Digital Key applet.
- 30 • The SE shall be compliant with the GlobalPlatform card specification and shall
31 implement GlobalPlatform Contactless Services – Amendment C [21], including support
32 for “Type F”.
- 33 • The Digital Key applet shall register the System Code ACCC_h as part of its contactless
34 protocol parameters for Type F (Sub-tag 80_h of the Type F anti-collision parameters entry
35 tag A0_h; see [21]).
- 36 • The device shall configure the NFC controller to route frames containing the NFCID2 of
37 System Code ACCC_h to the SE that hosts the Digital Key applet.
- 38 • The Digital Key applet shall register an NFCID2 (as defined in [DIGITAL]) as part of its
39 contactless protocol parameters for Type F. The NFCID2 is registered as the “PICC

- 1 Identifier” contained in sub-tag 81_h of the Type F anti-collision parameters entry tag A0_h
2 (see [21]).
- 3 • The following applies only if owner pairing over NFC-F has to be supported:
- 4 ○ The protocol defined in this appendix shall be implemented by the Digital Key
5 framework
- 6 ○ The Digital Key framework shall use the System Code 4CCC_h for NFC-F.
- 7 ○ The Digital Key framework shall use an NFCID2 in the range of
8 02FE000000000000_h to 02FEFFFFFFFFFF_h. The PAD0 field in the
9 SENSF_RES as defined in [17] shall be configured to be 01FE_h.
- 10 ○ The device shall configure the NFC controller to route frames containing the
11 NFCID2 of System Code 4CCC_h to the host with the Digital Key framework.

12 To support NFC-F, the following requirements apply on the vehicle side:

- 13 • The vehicle shall implement the protocol defined in this appendix. The protocol shall be
14 available for communication on the door NFC readers and console NFC reader.
- 15 • If configured to support NFC-F, the NFC readers shall include polling for NFC-F. The
16 polling shall use the system code for the Digital Key applet except in case of owner
17 pairing, in which case the system code for the Digital Key framework shall be used.

18 *Note:* As only devices that support the requested system code will respond to the polling, there is
19 no risk that the NFC link is established using NFC-F technology without the Digital Key
20 protocol being available.

21 On either side, the APDUs exchanged as part of the Digital Key protocol shall be encapsulated
22 as defined in the following sections before sending and shall be extracted accordingly after
23 receiving.

24 *E.3. Preconditions*

25 This specification is based on NFC-F technology, more specifically on the Type 3 Tag Platform
26 as defined in [17] and [18].

27 The vehicle and the device shall be able to send and receive NFC-F frames as defined in [17].
28 The device shall be able to respond to a SENSF_REQ command as defined in [17].

29 *E.4. Commands and responses*

30 The formats for NFC-F commands and responses defined in this section shall be transported
31 inside the Payload field of the NFC-F Data and Payload format as defined in [17].

32 The LEN byte and the CRC are not included in these format definitions. The LEN field has to be
33 prepended and the CRC to be appended to obtain the content for the data field of an NFC-F
34 frame.

35 The LEN field is supposed to be part of the command or response and therefore constitutes the
36 first byte of each command and response.

37 *E.4.1. Encapsulation Command (ENCAPSULATION_CMD)*

38 The ENCAPSULATION_CMD is used to transfer APDU data from the vehicle to the device.

39 The ENCAPSULATION_CMD shall be formatted as defined in Table E-1.

1 *Table E-1: ENCAPSULATION_CMD format*

Byte 1	Byte 2	Bytes 3–10	Bytes 11–n
C2 _h	P1	NFCID2	PAYLOAD

2 **P1**

3 The P1 field shall be formatted as defined in Table E-2.

4 *Table E-2: Format of the P1 field*

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
Message Type				Message Features				
0	0	0	0	0	0	0	0	CAPDU_DATA
					x			
					Chaining, if set to 1 _b			
0	0	0	1	0	0	0	0	RW_ACK
0	0	1	0	0	0	0	0	NACK
Any other value								RFU

- 5 In the following sections, the name in the “Meaning” column is often used to refer to an
6 ENCAPSULATION_CMD with a specific “Message Type” value.
7 Chaining set to 1_b indicates that the Payload field contains a segment of a larger data. If chaining
8 is set to 1_b, the Command is referred to as “CAPDU_DATA indicating chaining.” If “indicating
9 chaining” is not explicitly mentioned, the bit is set to 0_b.
10 The device shall ignore commands that have an RFU value.

11 **NFCID2**

12 The vehicle shall set the NFCID2 to the value included in the SENSF_RES response of the
13 targeted device. The device shall compare the NFCID2 contained in this command with its own
14 NFCID2. If they are not equal, the device shall not respond to this command.

15 **PAYLOAD**

16 For RW_ACK and NACK messages, the Payload field shall be empty.
17 The content of the Payload field for CAPDU_DATA is specified in Appendix [E-5](#).

18 **E.4.2. Encapsulation Response (ENCAPSULATION_RSP)**

19 The ENCAPSULATION_RSP is used to transfer APDU data from the device to the vehicle.
20 The ENCAPSULATION_RSP shall be formatted as defined in Table E-3.

21 *Table E-3: ENCAPSULATION_RSP format*

Byte 1	Byte 2	Bytes 3–10	Bytes 11–n
C3 _h	P1	NFCID2	PAYLOAD

22 **P1**

23 The P1 field shall be formatted as defined in Table E-4.

1

Table E-4: Format of the P1 field

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
	Message Type			Message Features				
0	0	0	0	0	0	0		RAPDU_DATA
							x	Chaining, if set to 1 _b
0	0	0	1	0	0	0	0	DEVICE_ACK
Any other value								RFU

- 2 In the following sections, the name in the “Meaning” column is often used to refer to an
3 ENCAPSULATION_RSP with a specific “Message Type” value.
4 Chaining set to 1_b indicates that the Payload field contains a segment of a larger data. If chaining
5 is set to 1_b, the response is referred to as “RAPDU_DATA indicating chaining.”. If “indicating
6 chaining” is not explicitly mentioned, the bit is set to 0_b.

7 **NFCID2**

- 8 The device shall set the NFCID2 to its own NFCID2. The vehicle shall verify that the NFCID2
9 in the response is the same as the one sent in the previous command. If the NFCID2 is different,
10 the vehicle shall ignore the response.

11 **PAYLOAD**

- 12 For DEVICE_ACK messages, the Payload field shall be empty.
13 The content of the Payload field for RAPDU_DATA is specified in Appendix [E.5](#).

14 *E.5. Protocol Operation*

15 E.5.1. Protocol activation

- 16 The protocol is implicitly activated if the device responds to a SENSF_REQ with a SENSF_RES
17 that includes an NFCID2 that is associated with one of the system codes defined in Appendix [E-](#)
18 [2](#) (see [\[18\]](#) for more information on activation of a Type 3 Tag Platform and data exchange with
19 a Type 3 Tag Platform).

- 20 The vehicle shall ignore the values of PAD1, MRTI_{CHECK}, and MRTI_{UPDATE} as defined in [\[17\]](#).

21 E.5.2. General rules

- 22 After sending an ENCAPSULATION_CMD, the vehicle shall be ready to receive an
23 ENCAPSULATION_RSP. After receiving an ENCAPSULATION_CMD without error, the
24 device shall respond with an ENCAPSULATION_RSP.

- 25 If the length of the C-APDU is 244 bytes or less, the C-APDU shall be transported in the
26 Payload field of a CAPDU_DATA message as defined in Appendix [E.4.1](#).

- 27 If the length of the R-APDU is 244 bytes or less, the R-APDU shall be transported in the
28 Payload field of an RAPDU_DATA message as defined in Appendix [E.4.2](#).

- 29 For APDUs with a larger size, refer to Appendix [E.5.3](#).

30 E.5.3. Chaining

1 The chaining feature allows transmitting APDUs that do not fit in a single command or response
2 by dividing it into segments. Each segment is transmitted in a separate command or response.

3 **C-APDU**

4 If the length of the C-APDU is above 244 bytes, the C-APDU shall be segmented and transferred
5 in multiple CAPDU_DATA messages.

6 All but the last segment shall be transported in the Payload field of a CAPDU_DATA message
7 indicating chaining as defined in Appendix [E.4.1](#). C-APDU segments that are transported in a
8 CAPDU_DATA message indicating chaining shall have a Payload field with a size of 244 bytes.
9 The last segment shall be transported in the Payload field of a CAPDU_DATA message
10 indicating no chaining.

11 The vehicle shall send the first segment of the C-APDU in a CAPDU_DATA message indicating
12 chaining. After receiving a CAPDU_DATA message indicating chaining, the device shall
13 acknowledge the receipt of the segment by sending a DEVICE_ACK as defined in Appendix
14 [E.4.2](#). After receiving a DEVICE_ACK, the vehicle shall send the next segment in a
15 CAPDU_DATA message indicating chaining, or, if it is the last segment, in a CAPDU_DATA
16 message indicating no chaining.

17 **R-APDU**

18 If the length of the R-APDU is greater than 244 bytes, the R-APDU shall be segmented and
19 transferred in multiple RAPDU_DATA messages.

20 All but the last segment shall be transported in the Payload field of an RAPDU_DATA message
21 indicating chaining as defined in Appendix [E.4.2](#). R-APDU segments that are transported in an
22 RAPDU_DATA message indicating chaining shall have a Payload field with a size of 244 bytes.
23 The last segment shall be transported in the Payload field of an RAPDU_DATA message
24 indicating no chaining.

25 The device shall send the first segment of the R-APDU in an RAPDU_DATA message
26 indicating chaining. After receiving an RAPDU_DATA message indicating chaining, the vehicle
27 shall acknowledge receipt of the segment by sending a RW_ACK as defined in Appendix [E.4.1](#).
28 After receiving a RW_ACK, the device shall send the next segment in an RAPDU_DATA
29 message indicating chaining, or, if it is the last segment, in an RAPDU_DATA message
30 indicating no chaining.

31 **E.5.4. Timing requirements**

32 The waiting time used by the protocol is announced in the SENSF_RES. The protocol uses one
33 waiting time, referred to as RWT.

34 The SENSF_RES as defined in [\[17\]](#) has the following format:

35 *Table E-5: SENSF_RES format*

Byte 1	Bytes 2–9	Bytes 10–11	Bytes 12–14	Byte 15	Byte 16	Byte 17	Bytes 18–19
01 _h	NFCID2	PAD0	Undefined Values (PAD1)	MRTI _{CHECK}	MRTI _{UPDATE}	Undefined Values (PAD2)	[RD]

36 In this specification, Byte 17 (PAD2) is referred to as RWTI. The device shall use a value for
37 RWTI that reflects the maximum time it requires to respond to a CAPDU_DATA message.

1 The Digital Key applet shall register its RWTI as part of its contactless protocol parameters for
2 Type F. The RWTI corresponds to the last byte of the “Response Time Descriptor” contained in
3 Sub-tag 81_h of the Type F anti-collision parameters entry tag A0_h (see [21]). The recommended
4 value for the first two bytes of the “Response Time Descriptor” (PAD0 of the SENSF_RES) is
5 01FF_h.

6 The format of the RWTI byte shall be as defined in Table E-6:

7

b7	b6	b5	b4	b3	b2	b1	b0	Meaning
					x	x	x	Parameter A
	x	x	x					Parameter B
x	x							Parameter E

8

9 The RWT shall be calculated according to the following formula:

$$10 \quad RWT = T \times ((A+1) + 8 \times (B+1)) \times 4^E$$

11 where T is a fixed duration with the following value: $T = 256 \times 16/f_c$ (~ 0.3020 ms).

12 After receiving the last byte of a ENCAPSULATION_CMD message, the device shall wait no
13 longer than RWT before sending the first byte of an ENCAPSULATION_RSP (which is the
14 LEN byte). If the vehicle does not receive the first byte of an ENCAPSULATION_RSP within
15 RWT + ΔRWT , the vehicle shall treat this as a timeout error.

16 The value of ΔRWT shall be 20 ms.

17 E.5.5. Error handling

18 Device

19 The device shall not attempt any error recovery. The device shall stay in receive mode when it
20 detects an error.

21 Vehicle

22 If a timeout error occurs, the vehicle shall resend the last command.

23 If the vehicle can detect transmission errors and if it receives a response with a transmission
24 error, the vehicle shall send a NACK as defined in Appendix [E.4.1](#).

25 After receiving a NACK, the device shall resend the last ENCAPSULATION_RSP.

26 The vehicle shall attempt sequential error recovery (without receiving a valid response in
27 between) up to a maximum of two times.

28 If not specified otherwise, the handling of protocol errors by the vehicle is implementation-
29 specific.

30 E.5.6. Protocol deactivation

31 The vehicle deactivates the protocol either by selecting another System Code via a SENSF_REQ
32 or by switching off the operating field.

1 **Appendix F. Digital Key Framework API**

2 This is a normative appendix, specifying API requirements.

3 *F.1. Overview*

4 The API requirements specified in this document enable mobile applications to access the Digital
5 Key framework running the Android system in mobile devices. This specification provides
6 interface definitions for an application developer to allow implementation on Android mobile
7 platforms. The Android APIs are defined in Android Digital Key Framework API [29].

8 *F.2. Functional Requirements*

9 The Digital Key framework shall provide common Digital Key service functionality via a set of
10 OS-specific APIs for Vehicle OEM apps. The following high-level services shall be provided to
11 the Vehicle OEM app:

12 Owner pairing, key sharing, and key management. At a minimum, these APIs shall be able to be
13 used to, respectively:

- 14 • Trigger owner pairing
- 15 • Trigger key sharing
- 16 • List Digital Keys

17 Additionally, the API should provide interfaces to enable the following functionalities:

- 18 • Owner pairing:
 - 19 ◦ Trigger owner pairing and provide owner pairing password
 - 20 ◦ Request/receive owner pairing status updates
- 21 • Key sharing:
 - 22 ◦ For owner: Provide entitlements (including proprietary ones), friendly name, and retrieve
23 sharing URL and sharing password from the framework.
 - 24 ◦ For friend: Receive/send sharing URL and capture acceptance/consent.
 - 25 ◦ Request/receive friend key status updates.
 - 26 ◦ Send notification to Vehicle OEM app when a friend key is created on friend device.
- 27 • Key management:
 - 28 ◦ Terminate Digital Key
 - 29 ◦ List all owner/shared Digital Keys with status, entitlements, friendly names, vehicle
30 identifier (vehicleIdentifier), and Digital Key identifier (keyID).
 - 31 ◦ Deeplink to owner/shared keys to jump from Vehicle OEM app directly to the key in the
32 device native representation.
 - 33 ◦ Sign vehicle identifier (vehicleIdentifier) and Digital Key identifier (keyID) with
34 attestable key from Digital Key applet instance on friend device to verify the identifiers
35 in Vehicle OEM Server.
 - 36 ◦ Sign data with attestable key from Digital Key applet instance on owner or friend device
37 to verify signature in Vehicle OEM Server.
 - 38 ◦ Read/write information related to Digital Key (e.g., information stored in private mailbox
39 and Instance CA ID).

- RKE Functionality
 - Trigger event based RKE action
 - Trigger enduring RKE action
 - Callback providing the arbitrary data from request continuation confirmation while action endures
 - Sending of arbitrary data in confirm continuation message as answer to the request above
- Vehicle Management
 - Retrieve vehicle state
 - subscribing to a certain function id range
 - requesting a certain function status value from the vehicle
 - reading the stored status data.

Appendix G.

G.1. Ranging Session Parameter Tables [WCC3]

As part of the ranging session configuration, each ranging session is configured with the following values:

- $N_{\text{RAN}}^k \subseteq \mathbb{N}^+$, set of all positive natural numbers.
- $N_{\text{RAN_S}}^k \subseteq \mathbb{N}^+$, set of all positive natural numbers.
- $N_{\text{Chap_per_Slot}}^k \subseteq \{3,4,6,8,9,12,24\}$
- $N_{\text{Slot_per_Round}}^k \subseteq \{6,8,9,12,16,18,24,32,36,48,72,96\}$
- $N_{\text{Slot_per_Round}}^k \subseteq \{6,8,9,12,16,18\}$ when the number of responder in each ranging round $N_{\text{Responder}}^k \leq 10$.

Table G-1 shows which combinations of $N_{\text{Chap_per_Slot}}^k$ and $N_{\text{Slot_per_Round}}^k$ result in valid ranging session configurations. During the handshake process for the k -th ranging session over the Bluetooth LE control channel (for details see Section 20.5.2), the Table G-1 is used to negotiate a valid ranging session configuration that maximizes the number of hopping rounds N_{Round}^k for a given set of constraint parameters $N_{\text{Chap_per_Slot}}^k$ and $N_{\text{Slot_per_Round}}^k$. With $N_{\text{Responder}}^k \leq 10$, the only the shaded part of the Table G-1 below will be used during this handshake process.

Table G-1: Number of Valid Ranging Rounds per 96 ms.

Number of Slots (per 96 ms)		$N_{\text{Slot_per_Round}}^k$												
$N_{\text{Chap_per_Slot}}^k$	Resulting T_{Slot}^k		6	8	9	12	16	18	24	32	36	48	72	96
	3	1.000 ms	16	12	N/A	8	6	N/A	4	3	N/A	2	N/A	1
	4	1.333 ms	12	9	8	6	N/A	4	3	N/A	2	N/A	1	N/A

Number of Slots (per 96 ms)			$N^k_{\text{Slot_per_Round}}$											
	6	2.000 ms	8	6	N/A	4	3	N/A	2	N/A	N/A	1	N/A	N/A
	8	2.667 ms	6	N/A	4	3	N/A	2	N/A	N/A	1	N/A	N/A	N/A
	9	3.000 ms	N/A	4	N/A	N/A	2	N/A	N/A	1	N/A	N/A	N/A	N/A
	12	4.000 ms	4	3	N/A	2	N/A	N/A	1	N/A	N/A	N/A	N/A	N/A
	24	8.000 ms	2	N/A	N/A	1	N/A							

1 *G.2. Hopping Sequence*

2 This section describes the method to generate the hopping sequence

3 *G.3. Default Hopping Sequence:*

4 This method is mandatory and shall be supported by any DK device.

5 For the k -th ranging session with N_{Round}^k ranging rounds per block and hoping key

6 $HOP_Key_RW^k$, the following function is used to generate the hopping round Sk index for
7 ranging block i :

$$8 \quad S^k(i, HOP_Key_RW^k, N_{\text{Round}}^k) = \left(\left(\left(i + HOP_Key_RW^k \right) \& 0xFFFF \right)^2 \bmod (2^{16} - 15) \right) N_{\text{Round}}^k \gg 16$$

9 The initiator shall generate and send the hopping key $HOP_Key_RW^k$ to the responder-device
10 during the ranging session initiation process (see Section [20.5.2](#)).

11 *G.4. AES-Based Hopping Sequence:*

12 This method is optional and may be supported by any DK device. It uses AES to generate the
13 hopping sequence.

14 For the k -th ranging session with N_{Round}^k ranging rounds per block and hopping key

15 $HOP_Key_RW^k$, the following function is used to generate the AES-based hopping round
16 index for ranging block i :

$$17 \quad S^k(i, HOP_Key_RW^k, N_{\text{Round}}^k) = \left(\left(\left(\text{AES}(i, HOP_Key_RW^k) \& 0xFFFF \right) N_{\text{Round}}^k \right) \gg 16 \right)$$

18 For the AES function, AES-128 shall be used. Here, the notation $\text{AES}(\text{plaintext}, \text{key})$ is used
19 where the $\text{AES}()$ operation pads each of the inputs to the left with 0x00 to reach the AES block
20 size.

21 **Appendix H. [WCC3]**

22 The following Python code is contributed to IEEE Mentor as in [\[36\]](#) and implements a
23 minimum-phase conversion from [\[34\]](#).

24

```
def min_phase(firseq, fft_pts):
    """A function to convert a FIR impulse response to min-phase
```

```
1      A function that calculates the minimum-phase equivalent of a
2 FIR
3      impulse response based on a Discrete Hilbert Transform
4 applied in the frequency domain.
5      """
6      # Recommended fft_pts >= 32768
7      #
8      # For guidelines on the required number of FFT points, see:
9      # N. Damera-Venkata, B. L. Evans and S. R. McCaslin,
10     # "Design of optimal minimum-phase digital FIR filters using
11     # discrete Hilbert transforms," in IEEE Transactions on
12 Signal
13     # Processing, vol. 48, no. 5, pp. 1491-1495, May 2000.
14 max_phase = np.real(np.fft.ifft( \
15     np.exp(hilbert(np.real(np.log(np.fft.fft( \
16         firseq, fft_pts)))))))
17 tmp_min_phase = max_phase[::-1] # reverse max phase
18 return tmp_min_phase[0:len(firseq)] # maintain length
19
20
```

21 **Appendix I. [WCC3]**

22 Pulse shapes using β of 0.45 with 32x oversampling per chip with total span of 8 chips are
23 provided here for informative reference only. The samples are left to right, row by row from
24 early to late samples. PulseShape 0x0 follows the mathematical equation in Section 21.5 and
25 PulseShape 0x1 is converted from PulseShape 0x0 with the code shown in [Appendix H](#).
Implementation may choose to gradually smooth the edge samples toward zero.

27 *I.1. PulseShape 0x0*

28	-0.0089385	-0.0080918	-0.0070627	-0.0058655	-0.0045178	-0.0030411	-0.0014601
29	0.0001978	0.0019028	0.0036231	0.0053257	0.0069768	0.0085426	0.0099899
30	0.0112867	0.0124031	0.0133117	0.0139884	0.0144129	0.0145693	0.0144466
31	0.0140389	0.0133461	0.0123737	0.0111332	0.0096417	0.0079225	0.0060041
32	0.0039205	0.0017102	-0.0005840	-0.0029158	-0.0052361	-0.0074940	-0.0096374
33	-0.0116144	-0.0133739	-0.0148670	-0.0160477	-0.0168743	-0.0173102	-0.0173247
34	-0.0168943	-0.0160030	-0.0146437	-0.0128178	-0.0105365	-0.0078208	-0.0047014
35	-0.0012189	0.0025766	0.0066255	0.0108598	0.0152038	0.0195746	0.0238837
36	0.0280380	0.0319412	0.0354955	0.0386030	0.0411676	0.0430968	0.0443030

1	0.0447061	0.0442347	0.0428277	0.0404366	0.0370264	0.0325774	0.0270859
2	0.0205660	0.0130496	0.0045875	-0.0047503	-0.0148749	-0.0256783	-0.0370341
3	-0.0487983	-0.0608102	-0.0728941	-0.0848606	-0.0965085	-0.1076271	-0.1179985
4	-0.1273998	-0.1356065	-0.1423943	-0.1475430	-0.1508388	-0.1520776	-0.1510679
5	-0.1476337	-0.1416170	-0.1328811	-0.1213124	-0.1068231	-0.0893529	-0.0688710
6	-0.0453773	-0.0189032	0.0104872	0.0426970	0.0775967	0.1150248	0.1547883
7	0.1966642	0.2404012	0.2857218	0.3323244	0.3798865	0.4280672	0.4765109
8	0.5248505	0.5727115	0.6197154	0.6654839	0.7096429	0.7518264	0.7916805
9	0.8288673	0.8630684	0.8939885	0.9213591	0.9449406	0.9645258	0.9799418
10	0.9910521	0.9977580	1.0000000	0.9977580	0.9910521	0.9799418	0.9645258
11	0.9449406	0.9213591	0.8939885	0.8630684	0.8288673	0.7916805	0.7518264
12	0.7096429	0.6654839	0.6197154	0.5727115	0.5248505	0.4765109	0.4280672
13	0.3798865	0.3323244	0.2857218	0.2404012	0.1966642	0.1547883	0.1150248
14	0.0775967	0.0426970	0.0104872	-0.0189032	-0.0453773	-0.0688710	-0.0893529
15	-0.1068231	-0.1213124	-0.1328811	-0.1416170	-0.1476337	-0.1510679	-0.1520776
16	-0.1508388	-0.1475430	-0.1423943	-0.1356065	-0.1273998	-0.1179985	-0.1076271
17	-0.0965085	-0.0848606	-0.0728941	-0.0608102	-0.0487983	-0.0370341	-0.0256783
18	-0.0148749	-0.0047503	0.0045875	0.0130496	0.0205660	0.0270859	0.0325774
19	0.0370264	0.0404366	0.0428277	0.0442347	0.0447061	0.0443030	0.0430968
20	0.0411676	0.0386030	0.0354955	0.0319412	0.0280380	0.0238837	0.0195746
21	0.0152038	0.0108598	0.0066255	0.0025766	-0.0012189	-0.0047014	-0.0078208
22	-0.0105365	-0.0128178	-0.0146437	-0.0160030	-0.0168943	-0.0173247	-0.0173102
23	-0.0168743	-0.0160477	-0.0148670	-0.0133739	-0.0116144	-0.0096374	-0.0074940
24	-0.0052361	-0.0029158	-0.0005840	0.0017102	0.0039205	0.0060041	0.0079225
25	0.0096417	0.0111332	0.0123737	0.0133461	0.0140389	0.0144466	0.0145693
26	0.0144129	0.0139884	0.0133117	0.0124031	0.0112867	0.0099899	0.0085426
27	0.0069768	0.0053257	0.0036231	0.0019028	0.0001978	-0.0014601	-0.0030411
28	-0.0045178	-0.0058655	-0.0070627	-0.0080918			
29							
30	<i>I.2. PulseShape 0x2</i>						
31	0.0193408	0.0273888	0.0369563	0.0481487	0.0610612	0.0757769	0.0923654
32	0.1108803	0.1313579	0.1538153	0.1782489	0.2046337	0.2329215	0.2630407

1	0.2948957	0.3283664	0.3633086	0.3995544	0.4369126	0.4751700	0.5140929
2	0.5534282	0.5929059	0.6322413	0.6711370	0.7092864	0.7463760	0.7820887
3	0.8161076	0.8481184	0.8778138	0.9048965	0.9290824	0.9501048	0.9677167
4	0.9816948	0.9918417	0.9979891	1.0000000	0.9977710	0.9912337	0.9803566
5	0.9651456	0.9456452	0.9219380	0.8941451	0.8624247	0.8269719	0.7880160
6	0.7458198	0.7006760	0.6529055	0.6028537	0.5508875	0.4973912	0.4427634
7	0.3874123	0.3317516	0.2761968	0.2211602	0.1670473	0.1142522	0.0631537
8	0.0141115	-0.0325377	-0.0764831	-0.1174435	-0.1551694	-0.1894459	-0.2200942
9	-0.2469734	-0.2699813	-0.2890554	-0.3041727	-0.3153495	-0.3226406	-0.3261383
10	-0.3259704	-0.3222984	-0.3153151	-0.3052419	-0.2923256	-0.2768351	-0.2590582
11	-0.2392974	-0.2178666	-0.1950870	-0.1712832	-0.1467796	-0.1218960	-0.0969449
12	-0.0722271	-0.0480290	-0.0246194	-0.0022470	0.0188624	0.0385076	0.0565141
13	0.0727348	0.0870515	0.0993751	0.1096460	0.1178335	0.1239357	0.1279780
14	0.1300121	0.1301142	0.1283833	0.1249388	0.1199185	0.1134756	0.1057763
15	0.0969971	0.0873217	0.0769383	0.0660371	0.0548067	0.0434326	0.0320936
16	0.0209603	0.0101925	-0.0000627	-0.0096722	-0.0185187	-0.0265018	-0.0335384
17	-0.0395639	-0.0445319	-0.0484147	-0.0512024	-0.0529029	-0.0535408	-0.0531563
18	-0.0518044	-0.0495529	-0.0464813	-0.0426788	-0.0382426	-0.0332763	-0.0278875
19	-0.0221866	-0.0162841	-0.0102898	-0.0043100	0.0015533	0.0072041	0.0125540
20	0.0175230	0.0220408	0.0260473	0.0294936	0.0323420	0.0345664	0.0361523
21	0.0370968	0.0374078	0.0371038	0.0362131	0.0347730	0.0328289	0.0304330
22	0.0276434	0.0245228	0.0211373	0.0175550	0.0138452	0.0100764	0.0063158
23	0.0026280	-0.0009263	-0.0042910	-0.0074151	-0.0102543	-0.0127706	-0.0149333
24	-0.0167193	-0.0181129	-0.0191063	-0.0196988	-0.0198975	-0.0197161	-0.0191746
25	-0.0182993	-0.0171212	-0.0156759	-0.0140028	-0.0121439	-0.0101431	-0.0080455
26	-0.0058961	-0.0037395	-0.0016186	0.0004259	0.0023562	0.0041382	0.0057420
27	0.0071423	0.0083185	0.0092556	0.0099438	0.0103786	0.0105611	0.0104975
28	0.0101990	0.0096813	0.0089643	0.0080718	0.0070304	0.0058694	0.0046200
29	0.0033145	0.0019857	0.0006661	-0.0006126	-0.0018203	-0.0029293	-0.0039144
30	-0.0047540	-0.0054301	-0.0059290	-0.0062416	-0.0063635	-0.0062950	-0.0060415
31	-0.0056132	-0.0050251	-0.0042966	-0.0034510	-0.0025154	-0.0015200	-0.0004973
32	0.00051848	0.00149222	0.00238888	0.00317447	0.00381706	0.00428794	0.00456281
33	0.00462307	0.00445708	0.00406159	0.00344317	0.00261972	0.00162203	0.00049542

1 -0.0006985 -0.0018802 -0.0029501 -0.0037867 -0.0042448 -0.0041529 -0.0033108
2 -0.0014877 0.00158115 0.00619674 0.0000000319
3

4 Appendix J. [WCC3]

5 J.1. UWB Test Vector

6 Values are Hexadecimal unless labeled Decimal.
7 STSIndex0_Decimal: 123456789
8 STSIndex0_Hex_BigEndian: 075bcd15
9 URSK: ed07a80d2beb00f785af2627c96ae7c118504243cb2c3226b3679daa0f7e616c
10 mUPSK1: 94ce7a78b643d0cac4e2e4f8b5f55a4d
11 mUPSK2: a93a052ba546f74aa1b771dfb8c89f503a199bcc7af64ad2881c94b320a1243e
12 UAD: 8b779d0709d9af2c324320afb5301a8f
13 Keysource-low: 8b77
14 Keysource-high: 9d07
15 DestinationShortAddress: 09d9
16 SourceLongAddress: af2c324320afb530
17 mURSK: 1cac2a6a5b04490c86089bb989c7d35987c47efc3fff71133bd98e6d7966de70
18 Salt: c56ee5c153daf28f917c6081c8bb63ca
19 PaddedSalt: 00000000000000000000000000000000c56ee5c153daf28f917c6081c8bb63ca
20 Selected_Protocol_Version: 0100
21 Selected_UWB_Config_Id: 0000
22 UWB_Session_Id: 12345678
23 STS_Index0: 075bcd15
24 Number_Responder_Nodes: 01
25 Session_RAN_Multiplier: 0a
26 Number_Slot_per_Round: 06
27 Number_Chaps_per_Slot: 06
28 Selected_PulseShape_Combo: 00
29 Ranging Configuration: 0100000012345678075bcd15010a060600
30 SaltedHash: 79e865186cbd864b955682c51cdd34f8
31 STSIndex_Decimal: 123456789
32 KDF Cascade Context:
33 0000000000010101000100010100010101010000010100010000000100010001
34 URSK_KT: d8c951ca6029a8ff5c89199b6ae07387301056ac3462f93db1afb2009cd89d2f
35 dURSK: 4578aa54d663cb5aa88070cd01c604a7
36 dUDSK: 932e0f6d9bb393096fe24cc79a704447
37

```
1 Second Block (Block index i=1), first Ranging Round (Round index s=0)
2 STSIndex_Decimal: 123457269
3 KDF Cascade Context:
4 000000000001010100010001010001010100000101010001010100010001
5 URSK_KT: ebd253fbf2c57294284de578fee5de984f3489d740e6fd5ee50af537c714af6e
6 dURSK: 9d40e2080db7d2cda114838189d7ae9f
7 dUDSK: 011adc74ba889e7e59a8ee61889d3fec
```

8 J.2. UWB Test Vector without hopping

9 Based on the UWB Test Vector of J.1 the following exchange of UWB PPDUs may occur:

```
10 Values are Hexadecimal.
11 First Ranging block:
12 STS_Index for Pre Poll: 07 5B CD 15
13 Frame Counter for Pre Poll: 00 00 00 00
14 CMAC-Data(URSK_KT): 00 00 00 01 55 52 53 4B 5F 4B 54 00 00 00 00 00
15 00 01 01 01 00 01 00 01 01 00 01 01 01 01 00 00
16 01 01 00 01 00 00 00 01 00 01 00 01 00 00 01 00
17 CMAC-Data(URSK_KT): 00 00 00 02 55 52 53 4B 5F 4B 54 00 00 00 00 00
18 00 01 01 01 00 01 00 01 01 00 01 01 01 01 00 00
19 01 01 00 01 00 00 00 01 00 01 00 01 00 00 01 00
20 URSK_KT: D8 C9 51 CA 60 29 A8 FF 5C 89 19 9B 6A E0 73 87
21 30 10 56 AC 34 62 F9 3D B1 AF B2 00 9C D8 9D 2F
22 CMAC-Data(URSK): 00 00 00 01 55 52 53 4B 00 79 E8 65 18 6C BD 86
23 4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 80
24 dURSK: 45 78 AA 54 D6 63 CB 5A A8 80 70 CD 01 C6 04 A7
25 CMAC-Data(UDSK): 00 00 00 01 55 44 53 4B 00 79 E8 65 18 6C BD 86
26 4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 80
27 dUDSK: 93 2E 0F 6D 9B B3 93 09 6F E2 4C C7 9A 70 44 47
28 PrePoll MHR: 49 2B D9 09 16 00 00 00 00 77 8B 07 9D AA 05 00
29 69 DF 04 01 0D 80 3F
30 PrePoll Payload: 78 56 34 12 16 CD 5B 07 00 00 00 00 00
31 AES-CCM* key K: 94 CE 7A 78 B6 43 D0 CA C4 E2 E4 F8 B5 F5 5A 4D
32 Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 00 06
33 AddAuthData: 00 17 49 2B D9 09 16 00 00 00 00 77 8B 07 9D AA
34 05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00
35 PlainTextData: 78 56 34 12 16 CD 5B 07 00 00 00 00 00 00 00 00
36 AuthData: 00 17 49 2B D9 09 16 00 00 00 00 77 8B 07 9D AA
37 05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00
38 78 56 34 12 16 CD 5B 07 00 00 00 00 00 00 00 00
```

```
1      B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 00 06 00 0D
2      X: DD 38 D8 1E 45 5A 1B AD A3 58 8C 62 6C 90 EF E7
3          A2 AB 2B AE DE C2 E5 B4 D1 42 9B AF 4D 48 2F 9B
4          E5 2E A2 52 9C 05 9F DD 0F 78 B9 BB A5 54 4D B8
5          CB 2A A9 D5 17 55 E2 40 00 93 E1 78 86 5F 63 D2
6      T: CB 2A A9 D5 17 55 E2 40
7      A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 00 00 06 00 00
8      A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 00 00 06 00 01
9      C: 4A 87 F6 F8 FD EC EF EA EA 19 34 00 43 4D CC 06
10     S0: 11 43 A9 A2 0F F6 E6 C3 B6 CE 61 6C 47 85 65 96
11     CipherText: 32 D1 C2 EA EB 21 B4 ED EA 19 34 00 43
12     U: DA 69 00 77 18 A3 04 83
13     PrePoll encrypted Payload: 32 D1 C2 EA EB 21 B4 ED EA 19 34 00 43
14                     DA 69 00 77 18 A3 04 83
15     STS-V for      POLL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 4F DB 1C DD 34 F8 -
16                     1C DD 35 17
17     STS for       POLL PPDU: E8 AE 41 D3 4E 61 87 10 95 A6 2B 65 EF 1F 15 50 ..
18                     9D 5F 97 B1 EB 2B 09 55 40 B2 87 72 4B D6 88 76
19     STS-V for 1. RESPONSE PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 4F DC 1C DD 34 F8 -
20                     1C DD 35 17
21     STS for 1. RESPONSE PPDU: FA 26 6A E7 77 40 FA 4A E6 81 2A F3 E9 69 9D A2 ..
22                     1D 71 EA 7E 7D 0E A1 47 CC EB F3 18 DF 1D EC 9A
23     STS-V for      FINAL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 4F DD 1C DD 34 F8 -
24                     1C DD 35 17
25     STS for       FINAL PPDU: 51 76 79 08 78 9D 7D B7 49 1C 87 4B 8E 2E AF 7E ..
26                     87 8E C1 D3 3E D9 79 80 E5 E3 2D 7E 17 97 46 E0
27     FinaData MHR: 49 2B D9 09 16 01 00 00 00 77 8B 07 9D AA 05 00
28                     69 DF 04 02 19 80 3F
29     FinaData Payload: 78 56 34 12 00 00 00 00 00 18 CD 5B 07 00 00 3C
30                     0F 01 00 FF 04 9E 07 29 00
31     AES-CCM* key K: 93 2E 0F 6D 9B B3 93 09 6F E2 4C C7 9A 70 44 47
32    Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 01 06
33     AddAuthData: 00 17 49 2B D9 09 16 01 00 00 00 77 8B 07 9D AA
34                     05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00
35     PlainTextData: 78 56 34 12 00 00 00 00 00 18 CD 5B 07 00 00 3C
36                     0F 01 00 FF 04 9E 07 29 00 00 00 00 00 00 00 00 00
37     AuthData: 00 17 49 2B D9 09 16 01 00 00 00 77 8B 07 9D AA
38                     05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00
39                     78 56 34 12 00 00 00 00 00 18 CD 5B 07 00 00 3C
```

```
1          OF 01 00 FF 04 9E 07 29 00 00 00 00 00 00 00 00 00 00 00 00  
2          B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 01 06 00 19  
3          X: 1A C5 15 F3 E1 77 DB A3 1D 20 DC 57 35 10 69 AD  
4          3A 51 46 95 AD A3 CB 01 EC 16 C4 BB E2 3F 28 D5  
5          BA CC 1B 33 47 D5 13 BB 39 40 56 13 78 AC C9 77  
6          8B EF 16 7F BC DA 85 BD AF B6 BB 8B 80 F0 4C E3  
7          B0 DA 6F BE 11 35 0C 2F 5C 80 50 DF 0B 7C CF D0  
8          T: B0 DA 6F BE 11 35 0C 2F  
9          A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 01 06 00 00  
10         A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 01 06 00 01  
11         A2: 01 AF 2C 32 43 20 AF B5 30 00 00 00 01 06 00 02  
12         C: E2 55 9D 5B EA 8C 3E 87 44 4C 08 F3 7B 2D 5F 10  
13         3C F2 DB 4A AB 57 E7 43 B7 78 D5 4D C2 AD C6 38  
14         S0: 2B 36 52 7B F4 57 8F FD CB C0 82 81 DD CD BB 6F  
15         CipherText: 9A 03 A9 49 EA 8C 3E 87 44 54 C5 A8 7C 2D 5F 2C  
16             33 F3 DB B5 AF C9 E0 6A B7  
17         U: 9B EC 3D C5 E5 62 83 D2  
18         FinaData encrypted Payload: 9A 03 A9 49 EA 8C 3E 87 44 54 C5 A8 7C 2D 5F 2C  
19             33 F3 DB B5 AF C9 E0 6A B7 9B EC 3D C5 E5 62 83 D2  
20 Second Ranging block:  
21         STS_Index for Pre Poll: 07 5B CE F5  
22         Frame Counter for Pre Poll: 00 00 00 02  
23         CMAC-Data(URSK_KT): 00 00 00 01 55 52 53 4B 5F 4B 54 00 00 00 00 00 00  
24             00 01 01 01 00 01 00 01 01 00 01 01 01 01 01 00 00  
25             01 01 01 00 01 01 01 01 00 01 00 01 00 00 01 00 00  
26         CMAC-Data(URSK_KT): 00 00 00 02 55 52 53 4B 5F 4B 54 00 00 00 00 00 00  
27             00 01 01 01 00 01 00 01 01 00 01 01 01 01 01 00 00  
28             01 01 01 00 01 01 01 01 00 01 00 01 00 00 01 00 00  
29         URSK_KT: EB D2 53 FB F2 C5 72 94 28 4D E5 78 FE E5 DE 98  
30             4F 34 89 D7 40 E6 FD 5E E5 0A F5 37 C7 14 AF 6E  
31         CMAC-Data(URSK): 00 00 00 01 55 52 53 4B 00 79 E8 65 18 6C BD 86  
32             4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 00 80  
33         dURSK: 9D 40 E2 08 0D B7 D2 CD A1 14 83 81 89 D7 AE 9F  
34         CMAC-Data(UDSK): 00 00 00 01 55 44 53 4B 00 79 E8 65 18 6C BD 86  
35             4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 00 80  
36         dUDSK: 01 1A DC 74 BA 88 9E 7E 59 A8 EE 61 88 9D 3F EC  
37         PrePoll MHR: 49 2B D9 09 16 02 00 00 00 77 8B 07 9D AA 05 00  
38             69 DF 04 01 0D 80 3F  
39         PrePoll Payload: 78 56 34 12 F6 CE 5B 07 01 00 00 00 00 00
```

```
1 AES-CCM* key K: 94 CE 7A 78 B6 43 D0 CA C4 E2 E4 F8 B5 F5 5A 4D
2 Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 02 06
3 AddAuthData: 00 17 49 2B D9 09 16 02 00 00 00 77 8B 07 9D AA
4 05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00
5 PlainTextData: 78 56 34 12 F6 CE 5B 07 01 00 00 00 00 00 00 00 00
6 AuthData: 00 17 49 2B D9 09 16 02 00 00 00 77 8B 07 9D AA
7 05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00
8 78 56 34 12 F6 CE 5B 07 01 00 00 00 00 00 00 00 00
9 B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 02 06 00 0D
10 X: 2F A9 D9 44 AA 74 A0 6F C7 CD AD 33 25 74 8A 5D
11 7E 8C 86 E7 4D 2D EF 05 8F AC AE 3C 02 88 3C 85
12 BC 7F 2D BB EE FE 33 9B 1B 77 F0 85 CE B1 0A AD
13 94 EF E7 D4 A6 04 BE 54 1F 47 96 EE E7 23 BA 94
14 T: 94 EF E7 D4 A6 04 BE 54
15 A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 02 06 00 00
16 A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 02 06 00 01
17 C: BA 38 C2 2E 8C E7 D5 C8 DC 89 27 31 E7 77 D0 84
18 S0: 6B 6D EE 0F 3E D9 56 7B C1 69 45 5B 33 14 8F 4D
19 CipherText: C2 6E F6 3C 7A 29 8E CF DD 89 27 31 E7
20 U: FF 82 09 DB 98 DD E8 2F
21 PrePoll encrypted Payload: C2 6E F6 3C 7A 29 8E CF DD 89 27 31 E7
22 FF 82 09 DB 98 DD E8 2F
23 STS-V for POLL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 51 BB 1C DD 34 F8 -
24 1C DD 35 17
25 STS for POLL PPDU: BE 4E 6F 9D 5B 1C 4B 76 51 AF 10 47 40 BA 22 C3 ..
26 45 24 6A 04 FB AE 29 81 58 D2 70 E9 1D EB F6 C3
27 STS-V for 1. RESPONSE PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 51 BC 1C DD 34 F8 -
28 1C DD 35 17
29 STS for 1. RESPONSE PPDU: 68 D3 38 09 A2 6A 26 08 47 0E D7 CC 14 67 2F D1 ..
30 59 F4 F9 EE 2D EC DB CE 03 0D 76 4C 75 A1 7C FA
31 STS-V for FINAL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 51 BD 1C DD 34 F8 -
32 1C DD 35 17
33 STS for FINAL PPDU: C0 B1 17 8D 50 60 C2 D6 70 0C C2 18 29 F6 BC 45 ..
34 B7 26 0E C9 6E 2C DE BB 1E 2E 2D 42 D3 58 26 39
35 FinaData MHR: 49 2B D9 09 16 03 00 00 00 77 8B 07 9D AA 05 00
36 69 DF 04 02 19 80 3F
37 FinaData Payload: 78 56 34 12 01 00 00 00 00 F8 CE 5B 07 00 00 3C
38 0F 01 00 04 05 9E 07 29 00
39 AES-CCM* key K: 01 1A DC 74 BA 88 9E 7E 59 A8 EE 61 88 9D 3F EC
```

```
1     Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 03 06
2      AddAuthData: 00 17 49 2B D9 09 16 03 00 00 00 77 8B 07 9D AA
3                  05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00
4      PlainTextData: 78 56 34 12 01 00 00 00 00 F8 CE 5B 07 00 00 3C
5                  0F 01 00 04 05 9E 07 29 00 00 00 00 00 00 00 00
6      AuthData: 00 17 49 2B D9 09 16 03 00 00 00 77 8B 07 9D AA
7                  05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00
8                  78 56 34 12 01 00 00 00 00 F8 CE 5B 07 00 00 3C
9                  0F 01 00 04 05 9E 07 29 00 00 00 00 00 00 00 00
10     B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 03 06 00 19
11     X: AA 21 1A C9 71 A4 FA B8 39 20 22 11 DA 71 B8 B1
12             44 44 62 2D E7 20 05 78 AF F1 86 A8 30 6C 5C B8
13             A8 00 46 D8 0A 92 EB F0 3D CD F5 AD F3 E3 92 F2
14             B0 77 A4 86 F9 E3 0D AE EB AC C8 67 BF 6A 45 32
15             42 DF 7B 1E B6 60 4B 9A 95 DC D0 49 82 33 93 B3
16     T: 42 DF 7B 1E B6 60 4B 9A
17     A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 03 06 00 00
18     A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 03 06 00 01
19     A2: 01 AF 2C 32 43 20 AF B5 30 00 00 00 03 06 00 02
20     C: 89 E7 B9 4F 63 EA 41 AC BA 5A BF 44 41 71 6C 01
21             0C C4 F9 0B CC 09 FB 62 66 80 74 6F 8E 5A 76 F0
22     S0: 09 7D 64 94 03 79 2F 64 E3 48 D8 DC BA 59 AC A5
23     CipherText: F1 B1 8D 5D 62 EA 41 AC BA A2 71 1F 46 71 6C 3D
24             03 C5 F9 0F C9 97 FC 4B 66
25     U: 4B A2 1F 8A B5 19 64 FE
26     FinaData encrypted Payload: F1 B1 8D 5D 62 EA 41 AC BA A2 71 1F 46 71 6C 3D
27     03 C5 F9 0F C9 97 FC 4B 66 4B A2 1F 8A B5 19 64 FE
```

28 J.3. UWB Test Vector with continuous hopping

29 Based on the UWB Test Vector of J.1 the following exchange of UWB PPDUs may occur:

```
30     Values are Hexadecimal.
31     Hop_Mode_Key for Default Hopping Sequence: CC 5D D7 9F
32     Rounds per Block (Nround): 00 50
33     First Ranging block:
34     STS_Index for Pre Poll: 07 5B CD 15
35     Frame Counter for Pre Poll: 00 00 00 00
36     CMAC-Data (URSK_KT): 00 00 00 01 55 52 53 4B 5F 4B 54 00 00 00 00 00
37             00 01 01 01 00 01 00 01 01 00 01 01 01 01 00 00
38             01 01 00 01 00 00 00 01 00 01 00 01 00 00 01 00
```

```
1 CMAC-Data (URSK_KT): 00 00 00 02 55 52 53 4B 5F 4B 54 00 00 00 00 00 00
2 00 01 01 01 00 01 00 01 01 00 01 01 01 01 01 00 00
3 01 01 00 01 00 00 00 01 00 01 00 01 00 00 00 01 00
4 URSK_KT: D8 C9 51 CA 60 29 A8 FF 5C 89 19 9B 6A E0 73 87
5 30 10 56 AC 34 62 F9 3D B1 AF B2 00 9C D8 9D 2F
6 CMAC-Data (URSK): 00 00 00 01 55 52 53 4B 00 79 E8 65 18 6C BD 86
7 4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 00 00 80
8 dURSK: 45 78 AA 54 D6 63 CB 5A A8 80 70 CD 01 C6 04 A7
9 CMAC-Data (UDSK): 00 00 00 01 55 44 53 4B 00 79 E8 65 18 6C BD 86
10 4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 00 00 80
11 dUDSK: 93 2E 0F 6D 9B B3 93 09 6F E2 4C C7 9A 70 44 47
12 PrePoll MHR: 49 2B D9 09 16 00 00 00 00 77 8B 07 9D AA 05 00
13 69 DF 04 01 0D 80 3F
14 PrePoll Payload: 78 56 34 12 16 CD 5B 07 00 00 00 00 00 00
15 AES-CCM* key K: 94 CE 7A 78 B6 43 D0 CA C4 E2 E4 F8 B5 F5 5A 4D
16 Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 00 00 06
17 AddAuthData: 00 17 49 2B D9 09 16 00 00 00 00 00 77 8B 07 9D AA
18 05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00 00
19 PlainTextData: 78 56 34 12 16 CD 5B 07 00 00 00 00 00 00 00 00 00
20 AuthData: 00 17 49 2B D9 09 16 00 00 00 00 00 77 8B 07 9D AA
21 05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00 00
22 78 56 34 12 16 CD 5B 07 00 00 00 00 00 00 00 00 00 00
23 B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 00 06 00 0D
24 X: DD 38 D8 1E 45 5A 1B AD A3 58 8C 62 6C 90 EF E7
25 A2 AB 2B AE DE C2 E5 B4 D1 42 9B AF 4D 48 2F 9B
26 E5 2E A2 52 9C 05 9F DD 0F 78 B9 BB A5 54 4D B8
27 CB 2A A9 D5 17 55 E2 40 00 93 E1 78 86 5F 63 D2
28 T: CB 2A A9 D5 17 55 E2 40
29 A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 00 06 00 00
30 A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 00 06 00 01
31 C: 4A 87 F6 F8 FD EC EF EA EA 19 34 00 43 4D CC 06
32 S0: 11 43 A9 A2 0F F6 E6 C3 B6 CE 61 6C 47 85 65 96
33 CipherText: 32 D1 C2 EA EB 21 B4 ED EA 19 34 00 43
34 U: DA 69 00 77 18 A3 04 83
35 PrePoll encrypted Payload: 32 D1 C2 EA EB 21 B4 ED EA 19 34 00 43
36 DA 69 00 77 18 A3 04 83
37 STS-V for POLL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 4F DB 1C DD 34 F8 -
38 1C DD 35 17
39 STS for POLL PPDU: E8 AE 41 D3 4E 61 87 10 95 A6 2B 65 EF 1F 15 50 ..
```

```
1          9D 5F 97 B1 EB 2B 09 55 40 B2 87 72 4B D6 88 76
2      STS-V for 1. RESPONSE PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 4F DC 1C DD 34 F8 -
3          1C DD 35 17
4      STS for 1. RESPONSE PPDU: FA 26 6A E7 77 40 FA 4A E6 81 2A F3 E9 69 9D A2 ..
5          1D 71 EA 7E 7D 0E A1 47 CC EB F3 18 DF 1D EC 9A
6      STS-V for FINAL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 4F DD 1C DD 34 F8 -
7          1C DD 35 17
8      STS for FINAL PPDU: 51 76 79 08 78 9D 7D B7 49 1C 87 4B 8E 2E AF 7E ..
9          87 8E C1 D3 3E D9 79 80 E5 E3 2D 7E 17 97 46 E0
10     Round Index of next Ranging Block (Default Hopping): 00 3E
11     FinaData MHR: 49 2B D9 09 16 01 00 00 00 77 8B 07 9D AA 05 00
12         69 DF 04 02 19 80 3F
13     FinaData Payload: 78 56 34 12 00 00 01 3E 00 18 CD 5B 07 00 00 3C
14         0F 01 00 C1 04 9E 07 2C 00
15     AES-CCM* key K: 93 2E 0F 6D 9B B3 93 09 6F E2 4C C7 9A 70 44 47
16    Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 01 06
17     AddAuthData: 00 17 49 2B D9 09 16 01 00 00 00 77 8B 07 9D AA
18         05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00
19     PlainTextData: 78 56 34 12 00 00 01 3E 00 18 CD 5B 07 00 00 3C
20         0F 01 00 C1 04 9E 07 2C 00 00 00 00 00 00 00 00 00
21     AuthData: 00 17 49 2B D9 09 16 01 00 00 00 77 8B 07 9D AA
22         05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00
23         78 56 34 12 00 00 01 3E 00 18 CD 5B 07 00 00 3C
24         0F 01 00 C1 04 9E 07 2C 00 00 00 00 00 00 00 00 00
25     B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 01 06 00 19
26     X: 1A C5 15 F3 E1 77 DB A3 1D 20 DC 57 35 10 69 AD
27         3A 51 46 95 AD A3 CB 01 EC 16 C4 BB E2 3F 28 D5
28         BA CC 1B 33 47 D5 13 BB 39 40 56 13 78 AC C9 77
29         F0 6B A2 25 F6 49 B4 7C EF C1 6B F9 0C C7 82 04
30         64 97 62 53 2E CE 36 69 95 42 5D A6 CE 40 95 54
31     T: 64 97 62 53 2E CE 36 69
32     A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 01 06 00 00
33     A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 01 06 00 01
34     A2: 01 AF 2C 32 43 20 AF B5 30 00 00 00 01 06 00 02
35     C: E2 55 9D 5B EA 8C 3E 87 44 4C 08 F3 7B 2D 5F 10
36         3C F2 DB 4A AB 57 E7 43 B7 78 D5 4D C2 AD C6 38
37     S0: 2B 36 52 7B F4 57 8F FD CB C0 82 81 DD CD BB 6F
38     CipherText: 9A 03 A9 49 EA 8C 3F B9 44 54 C5 A8 7C 2D 5F 2C
39         33 F3 DB 8B AF C9 E0 6F B7
```

```
1      U: 4F A1 30 28 DA 99 B9 94
2      FinaData encrypted Payload: 9A 03 A9 49 EA 8C 3F B9 44 54 C5 A8 7C 2D 5F 2C
3                      33 F3 DB 8B AF C9 E0 6F B7 4F A1 30 28 DA 99 B9 94
4 Second Ranging block:
5      STS_Index for Pre Poll: 07 5B D0 69
6      Frame Counter for Pre Poll: 00 00 00 02
7          CMAC-Data(URSK_KT): 00 00 00 01 55 52 53 4B 5F 4B 54 00 00 00 00 00
8                      00 01 01 01 00 01 00 01 01 00 01 01 01 01 00 01
9                      00 00 00 00 00 01 01 00 01 00 00 01 00 00 01 00
10         CMAC-Data(URSK_KT): 00 00 00 02 55 52 53 4B 5F 4B 54 00 00 00 00 00
11                    00 01 01 01 00 01 00 01 01 00 01 01 01 01 00 01
12                    00 00 00 00 00 01 01 00 01 00 00 01 00 00 01 00
13     URSK_KT: 06 32 7C C5 A3 6C AF 38 6A E3 F1 3F C8 CC A4 40
14                 42 7A A6 D9 A4 71 A6 E8 9C E4 63 22 F5 79 FE 73
15     CMAC-Data(URSK): 00 00 00 01 55 52 53 4B 00 79 E8 65 18 6C BD 86
16                 4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 00 80
17     dURSK: 74 36 F1 90 14 51 4D C9 6F 90 01 FF 55 EC 87 47
18     CMAC-Data(UDSK): 00 00 00 01 55 44 53 4B 00 79 E8 65 18 6C BD 86
19                 4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 00 80
20     dUDSK: 42 C8 4F 42 03 CE 36 DF 09 58 FF 21 CB 39 F2 F0
21     PrePoll MHR: 49 2B D9 09 16 02 00 00 00 77 8B 07 9D AA 05 00
22                 69 DF 04 01 0D 80 3F
23     PrePoll Payload: 78 56 34 12 6A D0 5B 07 01 00 01 3E 00
24         AES-CCM* key K: 94 CE 7A 78 B6 43 D0 CA C4 E2 E4 F8 B5 F5 5A 4D
25        Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 02 06
26     AddAuthData: 00 17 49 2B D9 09 16 02 00 00 00 77 8B 07 9D AA
27                 05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00
28     PlainTextData: 78 56 34 12 6A D0 5B 07 01 00 01 3E 00 00 00 00
29     AuthData: 00 17 49 2B D9 09 16 02 00 00 00 77 8B 07 9D AA
30                 05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00
31                 78 56 34 12 6A D0 5B 07 01 00 01 3E 00 00 00 00
32     B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 02 06 00 0D
33     X: 2F A9 D9 44 AA 74 A0 6F C7 CD AD 33 25 74 8A 5D
34                 7E 8C 86 E7 4D 2D EF 05 8F AC AE 3C 02 88 3C 85
35                 BC 7F 2D BB EE FE 33 9B 1B 77 F0 85 CE B1 0A AD
36                 4D 10 94 4C 36 D7 C3 B4 5A 3F D1 A6 51 8A B4 30
37     T: 4D 10 94 4C 36 D7 C3 B4
38     A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 02 06 00 00
39     A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 02 06 00 01
```

```
1      C: BA 38 C2 2E 8C E7 D5 C8 DC 89 27 31 E7 77 D0 84
2      S0: 6B 6D EE 0F 3E D9 56 7B C1 69 45 5B 33 14 8F 4D
3      CipherText: C2 6E F6 3C E6 37 8E CF DD 89 26 0F E7
4      U: 26 7D 7A 43 08 0E 95 CF
5      PrePoll encrypted Payload: C2 6E F6 3C E6 37 8E CF DD 89 26 0F E7
6                      26 7D 7A 43 08 0E 95 CF
7      STS-V for      POLL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 53 2F 1C DD 34 F8 -
8                      1C DD 35 17
9      STS for       POLL PPDU: 8F 33 3E 3D D2 5C 0F 57 EB 8B 25 9D 3D 73 DB BC ..
10                     F4 12 FA A3 01 20 83 02 FF E0 04 AC 43 60 D3 C5
11      STS-V for   1. RESPONSE PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 53 30 1C DD 34 F8 -
12                     1C DD 35 17
13      STS for    1. RESPONSE PPDU: C7 2B 1A 1B CE 0C 86 C8 E8 28 2A 0B 44 3F F5 FB ..
14                     B3 CE 69 49 6D FD 40 61 D8 DA B9 DA 6A FA 88 D5
15      STS-V for   FINAL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 53 31 1C DD 34 F8 -
16                     1C DD 35 17
17      STS for    FINAL PPDU: 41 B5 25 D1 2F DE 4A DC 74 02 F5 EB F8 15 28 AB ..
18                     B2 58 5D 96 82 ED 93 A8 E3 2C FE A4 37 EA 00 D5
19      Round Index of next Ranging Block (Default Hopping): 00 25
20      FinaData MHR: 49 2B D9 09 16 03 00 00 00 77 8B 07 9D AA 05 00
21                     69 DF 04 02 19 80 3F
22      FinaData Payload: 78 56 34 12 01 00 01 25 00 6C D0 5B 07 00 00 3C
23                     0F 01 00 B1 04 9E 07 2C 00
24      AES-CCM* key K: 42 C8 4F 42 03 CE 36 DF 09 58 FF 21 CB 39 F2 F0
25      Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 03 06
26      AddAuthData: 00 17 49 2B D9 09 16 03 00 00 00 77 8B 07 9D AA
27                     05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00
28      PlainTextData: 78 56 34 12 01 00 01 25 00 6C D0 5B 07 00 00 3C
29                     0F 01 00 B1 04 9E 07 2C 00 00 00 00 00 00 00 00
30      AuthData: 00 17 49 2B D9 09 16 03 00 00 00 77 8B 07 9D AA
31                     05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00
32                     78 56 34 12 01 00 01 25 00 6C D0 5B 07 00 00 3C
33                     0F 01 00 B1 04 9E 07 2C 00 00 00 00 00 00 00 00
34      B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 03 06 00 19
35      X: F1 F4 F4 BB 98 C3 45 B1 A1 39 09 05 50 0C 89 B2
36                     BB 2C B1 43 80 48 6B E5 2B BD E6 E4 70 72 ED EA
37                     62 26 07 E2 92 0C B7 C0 C8 5A 3C 75 04 5F 2C D7
38                     C8 67 6D FA 41 A6 3A 92 B5 8B 9D 63 7F B1 9E C8
39                     4B BD B3 37 B3 B0 89 34 E2 15 B7 48 14 42 64 65
```

```
1      T: 4B BD B3 37 B3 B0 89 34
2      A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 03 06 00 00
3      A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 03 06 00 01
4      A2: 01 AF 2C 32 43 20 AF B5 30 00 00 00 03 06 00 02
5      C: 5F 13 1C E8 FF C1 D0 C6 9B 92 4E 47 B5 27 09 4C
6          2B C8 5C F0 57 B9 1D E5 9B 33 4E 9A 31 81 D2 46
7      S0: C8 E7 95 98 8E 7B 1A 69 48 2A 3B EF CE 43 C5 A7
8      CipherText: 27 45 28 FA FE C1 D1 E3 9B FE 9E 1C B2 27 09 70
9          24 C9 5C 41 53 27 1A C9 9B
10     U: 83 5A 26 AF 3D CB 93 5D
11     FinaData encrypted Payload: 27 45 28 FA FE C1 D1 E3 9B FE 9E 1C B2 27 09 70
12          24 C9 5C 41 53 27 1A C9 9B 83 5A 26 AF 3D CB 93 5D
13 Third Ranging block:
14     STS_Index for Pre Poll: 07 5B D1 B3
15     Frame Counter for Pre Poll: 00 00 00 04
16         CMAC-Data(URSK_KT): 00 00 00 01 55 52 53 4B 5F 4B 54 00 00 00 00 00
17             00 01 01 01 00 01 00 01 01 00 01 01 01 01 00 01
18             00 00 00 01 01 00 01 01 00 00 01 01 00 00 01 00
19         CMAC-Data(URSK_KT): 00 00 00 02 55 52 53 4B 5F 4B 54 00 00 00 00 00
20             00 01 01 01 00 01 00 01 01 00 01 01 01 01 01 00
21             00 00 00 01 01 00 01 01 00 00 01 01 00 00 01 00
22     URSK_KT: 2F B7 E8 39 2D E1 00 6E 6C CB 83 83 AB 12 B5 C0
23             29 85 F4 87 65 F1 C0 B7 89 7F 61 1C B1 67 D8 0D
24         CMAC-Data(URSK): 00 00 00 01 55 52 53 4B 00 79 E8 65 18 6C BD 86
25             4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 00 80
26     dURSK: A2 B7 89 A9 27 0E 74 85 D5 D9 48 D1 73 1B 92 4B
27         CMAC-Data(UDSK): 00 00 00 01 55 44 53 4B 00 79 E8 65 18 6C BD 86
28             4B 95 56 82 C5 1C DD 34 F8 00 00 00 00 00 00 00 80
29     dUDSK: CF 27 C7 7E 15 B9 E6 B9 88 62 90 32 FF 9D EE C9
30     PrePoll MHR: 49 2B D9 09 16 04 00 00 00 77 8B 07 9D AA 05 00
31             69 DF 04 01 0D 80 3F
32     PrePoll Payload: 78 56 34 12 B4 D1 5B 07 02 00 01 25 00
33         AES-CCM* key K: 94 CE 7A 78 B6 43 D0 CA C4 E2 E4 F8 B5 F5 5A 4D
34        Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 04 06
35         AddAuthData: 00 17 49 2B D9 09 16 04 00 00 00 77 8B 07 9D AA
36             05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00
37         PlainTextData: 78 56 34 12 B4 D1 5B 07 02 00 01 25 00 00 00 00
38         AuthData: 00 17 49 2B D9 09 16 04 00 00 00 77 8B 07 9D AA
39             05 00 69 DF 04 01 0D 80 3F 00 00 00 00 00 00 00 00
```

```
1          78 56 34 12 B4 D1 5B 07 02 00 01 25 00 00 00 00 00
2          B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 04 06 00 0D
3          X: 77 6D 21 76 CF AA D5 17 95 E7 C1 A0 D0 09 48 18
4          D5 5F B6 8F 77 C3 87 E6 31 CD A1 0C 49 C9 97 87
5          05 62 09 AD 05 80 F9 5B 25 76 7A A8 91 01 53 69
6          88 8E 6E 6E 50 AD FA 4D 36 31 F1 A6 7B A4 7C CA
7          T: 88 8E 6E 6E 50 AD FA 4D
8          A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 04 06 00 00
9          A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 04 06 00 01
10         C: AA 3A 5F C4 60 B9 7D 9D 43 60 01 22 99 E7 DF 11
11         S0: 49 68 F9 C5 37 C4 FD C6 D3 50 6F 48 65 D7 72 37
12         CipherText: D2 6C 6B D6 D4 68 26 9A 41 60 00 07 99
13         U: C1 E6 97 AB 67 69 07 8B
14         PrePoll encrypted Payload: D2 6C 6B D6 D4 68 26 9A 41 60 00 07 99 C1 E6 97
15                           AB 67 69 07 8B
16         STS-V for      POLL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 54 79 1C DD 34 F8 -
17                           1C DD 35 17
18         STS for       POLL PPDU: D9 9B 10 0A C2 38 BB 61 3C 9E 26 94 57 9D C9 C6 ..
19                           CF C4 28 0D E1 09 5E 22 80 54 64 38 B0 75 CC 30
20         STS-V for   1. RESPONSE PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 54 7A 1C DD 34 F8 -
21                           1C DD 35 17
22         STS for    1. RESPONSE PPDU: DD 39 EC 72 E6 CD AC FF E1 17 3B AA 3E 77 3C B3 ..
23                           4C F1 78 CB 1C 47 68 E8 9C 74 17 A0 25 5B 17 3D
24         STS-V for   FINAL PPDU: 79 E8 65 18 6C BD 86 4B 9C B2 54 7B 1C DD 34 F8 -
25                           1C DD 35 17
26         STS for    FINAL PPDU: 2F FD 4E F7 3A 8B EC 9E 6D 62 FF F8 B1 34 A0 7A ..
27                           53 D8 A3 FB 00 23 05 D7 AE 7D 5A DE 87 EB 87 C4
28         Round Index of next Ranging Block (Default Hopping): 00 0C
29         FinaData MHR: 49 2B D9 09 16 05 00 00 00 77 8B 07 9D AA 05 00
30                           69 DF 04 02 19 80 3F
31         FinaData Payload: 78 56 34 12 02 00 01 0C 00 B6 D1 5B 07 00 00 3C
32                           0F 01 00 C0 04 9E 07 2C 00
33         AES-CCM* key K: CF 27 C7 7E 15 B9 E6 B9 88 62 90 32 FF 9D EE C9
34        Nonce N: AF 2C 32 43 20 AF B5 30 00 00 00 05 06
35         AddAuthData: 00 17 49 2B D9 09 16 05 00 00 00 77 8B 07 9D AA
36                           05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00 00
37         PlainTextData: 78 56 34 12 02 00 01 0C 00 B6 D1 5B 07 00 00 3C
38                           0F 01 00 C0 04 9E 07 2C 00 00 00 00 00 00 00 00 00
39         AuthData: 00 17 49 2B D9 09 16 05 00 00 00 77 8B 07 9D AA
```

```
1      05 00 69 DF 04 02 19 80 3F 00 00 00 00 00 00 00 00
2      78 56 34 12 02 00 01 0C 00 B6 D1 5B 07 00 00 00 3C
3      0F 01 00 C0 04 9E 07 2C 00 00 00 00 00 00 00 00 00
4      B0: 59 AF 2C 32 43 20 AF B5 30 00 00 00 05 06 00 19
5      X: 70 B8 B0 47 F2 D5 2F CF 2C CB E6 9E 75 08 FE FE
6      51 1C 21 74 E3 D8 D9 70 39 5D F6 6B 9E AB 49 7A
7      B7 49 31 26 50 97 E6 06 5E 77 67 5F 33 9E 52 3F
8      23 00 69 8F 0D EA 5A D7 82 72 86 09 D3 36 6A 96
9      46 09 CE AE 51 E6 AF 42 7A 3D B2 C0 AD 68 E6 B1
10     T: 46 09 CE AE 51 E6 AF 42
11     A0: 01 AF 2C 32 43 20 AF B5 30 00 00 00 05 06 00 00
12     A1: 01 AF 2C 32 43 20 AF B5 30 00 00 00 05 06 00 01
13     A2: 01 AF 2C 32 43 20 AF B5 30 00 00 00 05 06 00 02
14     C: 1E 33 E7 49 01 D8 64 E1 DB 6C 6B E0 6C 3B CF 1D
15     41 77 C7 8E 21 A1 52 FE D5 67 DA 56 3A 1B B9 C4
16     S0: E5 07 B5 67 1A A8 DB 47 51 92 17 AB 6B D7 AC 21
17     CipherText: 66 65 D3 5B 03 D8 65 ED DB DA BA BB 6B 3B CF 21
18             4E 76 C7 4E 25 3F 55 D2 D5
19     U: A3 0E 7B C9 4B 4E 74 05
20     FinaData encrypted Payload: 66 65 D3 5B 03 D8 65 ED DB DA BA BB 6B 3B CF 21
21             4E 76 C7 4E 25 3F 55 D2 D5 A3 0E 7B C9 4B 4E 74 05
22
```