

# Telink SIG Mesh

# SDK Developer Handbook

AN-17120400-E7

Ver1.6.0 2024.04.23

## Keyword

SIG Mesh

## Brief

This document is Telink SIG Mesh SDK Developer Handbook.

Teim Semiconductor



Published by Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd, Zhangjiang Hi-Tech Park, Shanghai, China

© Telink Semiconductor All Rights Reserved

### Legal Disclaimer

This document is provided as-is. Telink Semiconductor reserves the right to make improvements without further notice to this document or any products herein. This document may contain technical inaccuracies or typographical errors. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein.

Copyright © 2024 Telink Semiconductor (Shanghai) Co., Ltd.

### Information

For further information on the technology, product and business term, please contact Telink Semiconductor Company www.telink-semi.com

For sales or technical support, please send email to the address of:

telinksales@telink-semi.com

telinksupport@telink-semi.com



## **Revision History**

Version	Changes
V1.0.0	Initial release.
V1.1.0	This is the second release, compare with last version, the following parts have been updated: 1. SDK Overview; 2. Global Configuration Introduction; 3. 8268 Mesh Project Introduction; 4. Provisioner(Gateway) Project Introduction; 5. SWITCH Project Introduction
V1.2.0	This is the third release, compare with last version, the following parts have be updated: 1 SDK Overview; 4 Debugging Tool Instruction; 11 Mesh LPN Project Introduction; 8 Global Configuration File Introduction. The following parts are added: 2 MCU Basic Modules; 5 Factory Test Mode; 6 Important SDK Modules; 7 Vendor Model Introduction; 9 8258 MESH Project Introduction; 13 Connect with a Platform; 14 Factory Reset; 15 SIG Remote provision Demo; 16 Fast bind Mode(PROVISION_FLOW_SIMPLE_EN Mode); 17 Private Fast provision Function Demo; 18 Private online status Function Demo; 19 OTA Test Brief; 20 Network Sharing; 21 Control Nodes via INI Demo
V1.3.0	This is the fourth release, compare with last version, the following parts have be updated: Delete draft feature
V1.4.0	This is the fifth release, compare with last version, corrected some terminology.
V1.5.0	This is the sixth release, compare with last version, new chapters 21~34 chapters have been added, and the following sections have been updated: 2. MCU basic modules; 3. commonly used modules in SDK; 7. use of Vendor model; 10. Gateway; 11. LPN; 12. Switch.
V1.6.0	This is the seventh release, compare with last version, new chapter Android & iOS APP User Guide is added.



## Contents

Rev	vision I	History	3				
1	SDK	DK Overview					
	1.1	SDK File Architecture	28				
		1.1.1 main.c	30				
		1.1.2 app_config.h	30				
		1.1.3 BLE stack entry	31				
	1.2	Demo Project	31				
	1.3	LIGHT TYPE SEL Introduction	33				
	1.4	Version ID(VID) and Product ID(PID) Configuration	35				
	1.5	Mobile App Introduction	36				
			36				
			36				
		1512 iOS Δρο	30				
		1513 App Operating Instructions	37				
	16	Mash Application Dasket Ty/Dy Drocoscing	יכ דכ				
	1.0	16.1 Dacket Transmission Exaction	יכ דכ				
			יכ סכ				
			20				
		1.6.4 Decket Reception Flow	40				
			40				
	1 7	Taliak Datus Mathed Istanduation	41				
	1.7		41				
			41				
_			43				
2	MCU		47				
	2.1	Flash and RAM map	47				
		2.1.1 Flash Map Introduction	47				
		2.1.2 RAM map (8258 64K)	48				
	2.2	Checking of Stack Overflow and Retention RAM Overflow	50				
		2.2.1 Checking Method of Stack Overflow	50				
		2.2.1.1 Checking Method of Normal Stack Overflow 5	50				
		2.2.1.2 Checking Method of irq_stack Overflow	51				
		2.2.2 RAM Remaining Size Analysis	51				
		2.2.3 Checking Whether The Stack Overflows Using 8258 as An Example	55				
		2.2.3.1 Checking Whether The Normal Stack Overflows	55				
		2.2.3.2 Checking Whether The Irq Stack Overflows	56				
		2.2.4 Size Calculation of Retention RAM	57				
	2.3	Startup File cstartup.s and Link File boot.link	58				
	2.4	Clock	58				
		2.4.1 System clock & System Timer	59				
		2.4.2 System Timer Usage	50				
3	Mes	h Spec Introduction	63				
	3.1	Layered architecture	63				
		3.1.1 Model layer	64				
		3.1.2 Foundation Model layer	64				

		8.1.3 Access layer	64
		8.1.4 Transport layer	64
		8.1.5 Network layer	64
		8.1.6 Bearer layer	65
	3.2	Architectural concepts	65
		B.2.1 States	65
		3.2.2 Bound states	65
		8.2.3 Messages	65
		3.2.4 Node & Elements	65
		8.2.5 Models	65
		3.2.6 Publish & subscribe	66
		8.2.7 Security	66
		3.2.8 Sequence Number Storage	66
		B.2.9 Friendship	67
		8.2.10 Features	67
		3.2.11 Mesh Topology	68
	3.3	1esh networking	68
		B.3.1 Network layer	68
		B.3.2 Access layer	70
		B.3.3 Transport layer	71
		8.3.4 Mesh beacon	71
		8.3.5 IV update flow	71
		B.3.6 Heartbeat	72
		B.3.7 Health	72
4	Deb	gging Tool Instructions	75
	4.1	Download Firmware	75
	4.2	BLE Connection and Adding Light in Gateway USB Mode	80
	4.3	BLE Connection and Adding Light in Gateway UART Mode	85
	4.4	BLE Connection and Adding Light in GATT master dongle Mode	85
	4.5	Control Corresponding Nodes	86
		I.5.1 UI Display and on/off Control of Single/All Node(s)	86
		I.5.2 Group Control (Subscription Demo)	88
		I.5.3 Configure Node Parameter with UI	89
	4.6	ime model operation	94
	4.7	Scene model operation	96
	4.8	Scheduler model operation	99
5	Fact	ry Test Mode	103
	5.1	Purpose	103
	5.2	actory Test Mode Parameters	103
	5.3	Default Test-able Commands	103
6	Imp	tant SDK Modules	104
	6.1	Configure Mesh SDK Default Feature	104
	6.2	Common Macro Definitions	104
		5.2.1 LIGHT_CNT and ELE_CNT_EVERY_LIGHT	105
		5.2.2 ONPOWER_UP_SELECT	105
		0.2.3 MESH_POWERUP_BASE_TIME	105
		5.2.4 Checking Whether a Node has been Provisioned	106

• Telink

	6.3	Definit	tion of th	e Number of Elements of a Node $\ldots$	106
	6.4	Group	ing Featu	rres and Share-model	107
	6.5	Metho	d for a N	ode to Get the Group Number	108
	6.6	Heartt	beat dem	onstration	109
	6.7	Mesh	ADV Senc	J Timing	110
	6.8	API fo	r Mesh A[	DV Payload Setting	111
		6.8.1	Unprovis	sioned Device Beacon	111
		6.8.2	Mesh Pr	ovisioning Service Advertising	111
		6.8.3	Mesh Se	cure Network Beacon	111
		6.8.4	Mesh Pr	oxy ADV	111
	6.9	Mesh	Receiving	ransmitting Self-defined Packet	112
	6.10	Metho	d to Mod	ify the Maximum Number of Nodes in a Mesh Network	113
	6.11	Telink	Customiz	zed Mode for Sending Mesh Messages via Extended Broadcast Package ex-	
		tend_a	adv		114
		6.11.1	Functior	n Introduction	114
		6.11.2	Test Met	thods	115
			6.11.2.1	Node Configuration	115
			6.11.2.2	Provisioner Configuration	116
			6.11.2.3	Precaution	116
	6.12	Applic	ation of S	Soft Timer	116
		6.12.1	Introduc	tion of Soft Timer	116
		6.12.2	Soft Tim	ner Initialization	117
		6.12.3	Query P	rocessing for Soft Timer $\ldots$	117
		6.12.4	Task Cor	nfiguration of Soft Timer $\ldots$	118
		6.12.5	Task Del	letion of Soft Timer	118
		6.12.6	Example	e of Soft_timer Cycle Send Command	118
	6.13	Use of	f the Long	g Sleep Interface	119
		6.13.1	Functior	n Name	119
		6.13.2	Use Met	hods	119
	6.14	Wakeu	Jp Source	e Identification Interface	120
		6.14.1	API Fund	ction Name	120
		6.14.2	Use Met	hods	121
	6.15	Key So	canning .		121
		6.15.1	Matrix K	leyboard Mode	122
		6.15.2	Button N	Mode	122
7	Ven	dor Mo	del Intro	duction	123
	7.1	Adding	g vendor	model	123
	7.2	Adding	g vendor	command register reference	123
		7.2.1	vendor_	opcode	123
		7.2.2	Steps of	Adding Vendor Opcode	123
			7.2.2.1	Add Definition of Vendor Opcode	124
			7.2.2.2	Add Registration of Vendor Opcode	124
			7.2.2.3	mesh_cmd_sig_func_t introduction	125
			7.2.2.4	Adding Command Callbacks	126
		_	7.2.2.5	Add TID Registration	126
		7.2.3	Example	e of Adding a Knowledge-command	127
		7.2.4	Add Una	acknowledged command	128

		7.2.5	Publish function re	egistration						. 128
	7.3	Add th	e Vendor Opcode S	Subcommand						. 129
		7.3.1	Vendor Subcomma	and Range						. 129
		7.3.2	Steps of Adding Ve	endor Subcomm	and					. 129
			7.3.2.1 Add the [	Definition of the	Vendor Sub	command				. 129
			7.3.2.2 Add Regi	stration of the V	endor Subco	mmand .				. 130
			7.3.2.3 vd_group	_g_func_t Intro	duction					. 130
			7.3.2.4 Adding S	ubcommands Ca	allback Funct	ions				. 130
		7.3.3	Adding Acknowled	lge Type Subcon	nmand					. 131
		7.3.4	Add Subcommand	s of Type Unack	nowledge .					. 132
		7.3.5	Write API for Send	ling VD_GROUP_	_G_ON Com	mand				. 132
		7.3.6	Example of Adding	g an Empty Vend	dor Subcomr	nand				. 132
8	Glot	oal Con	iguration File Intr	oduction						. 133
	8.1	mesh_	config.h							. 133
	8.2	mesh_	node.h							. 136
	8.3	app_n	esh.h							. 136
		8.3.1	Macro introductior	۱		,				. 136
		8.3.2	Function introduct	ion						. 137
	8.4	арр_р	ovision.c							. 138
	8.5	mesh_	node.c							. 138
	8.6	mesh_	common.c file intro	oduction						. 138
	8.7	cmd_i	iterface.h file intro	duction						. 144
	8.8	vendo	_model.c file intro	duction	, O					. 144
	00	mach								4 4 5
	0.9	mesn_	test_cmd.c file intr	oduction						. 145
9	o.9 825	8 MESI	test_cmd.c file intr Project Introduct	oduction		· · · · · ·	· · · · · ·	· · · · · ·	· · · · · ·	. 145 . 146
9	0.9 <b>825</b> 9.1	B MESH app_c	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h	oduction t <b>ion</b> .		· · · · · · · · · · · · · · · · · · ·	· · · · · · · ·	•••••• •••••	· · · · · · ·	. 145 . 146 . 146
9	825 9.1 9.2	app_c	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction	oduction t <b>ion</b>		· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · ·	. 145 . 146 . 146 . 147
9	825 9.1 9.2	<b>8 MESI</b> app_c app.c 9.2.1	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>i</i>	oduction tion	Adv respons	• • • • • • • • • • • • • • • • • • •	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · ·	. 145 . <b>146</b> . 146 . 147 . 147
9	825 9.1 9.2	8 MESH app_c app.c 9.2.1 9.2.2	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F	oduction t <b>ion</b> 	Adv respons		· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	<ul> <li>. 145</li> <li>. 146</li> <li>. 147</li> <li>. 147</li> <li>. 147</li> </ul>
9	8.9 825 9.1 9.2	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle	oduction t <b>ion</b> 	Adv respons	e packet .	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	<ul> <li>. 145</li> <li>. 146</li> <li>. 147</li> <li>. 147</li> <li>. 147</li> <li>. 147</li> <li>. 147</li> </ul>
9	8.9 825 9.1 9.2	<b>8 MESH</b> app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop ()	oduction	Adv respons	e packet .	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> </ul>
9	825 9.1 9.2	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init()	oduction	Adv respons	e packet .	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·	<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> </ul>
9	8.9 825 9.1 9.2	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui()	oduction	Adv respons	e packet .	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·		<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>148</li> <li>149</li> </ul>
9	9.3 9.3	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction	oduction	Adv respons	e packet .	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·		<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>148</li> <li>149</li> <li>149</li> </ul>
9	9.3 9.4	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction .	oduction	Adv respons	e packet .		· · · · · · · · · · · · · · · · · · ·		<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> </ul>
9	9.3 9.4 <b>Prov</b>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () void proc_ui() t.c file introduction . <b>(Gateway) Projec</b>	oduction	Adv respons	e packet .		· · · · · · · · · · · · · · · · · · ·		<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>148</li> <li>149</li> </ul>
9	<ul> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c <b>visione</b> Provis	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () void proc_ui() void proc_ui() t.c file introduction file introduction . <b>(Gateway) Projec</b> oner Function Intro	oduction	Adv respons	e packet .		· · · · · · · · · · · · · · · · · · ·		<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> </ul>
9	<ul> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c visione Provis 10.1.1	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () void proc_ui() void proc_ui() t.c file introduction file introduction . <b>(Gateway) Project</b> oner Function Intro adv-bearer and gate	oduction	Adv respons	e packet .		· · · · · · · · · · · · · · · · · · ·		<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> </ul>
9	<ul> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c Visione Provis 10.1.1	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> app_event_handle main_loop () void proc_ui() t.c file introduction file introduction . <b>(Gateway) Project</b> oner Function Intro adv-bearer and gate oner Principle	oduction	Adv respons	e packet .				<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> <li>154</li> </ul>
9	<ul> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> <li>10.2</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c visione Provis 10.1.1 Provis 10.2.1	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction file introduction . <b>(Gateway) Project</b> oner Function Intro adv-bearer and gate oner Principle Command Interact	oduction	Adv respons	e packet .				<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> <li>154</li> <li>154</li> </ul>
9	<ul> <li>9.3</li> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> <li>10.2</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c Visione Provis 10.1.1 Provis 10.2.1 10.2.2	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction file introduction . <b>(Gateway) Project</b> oner Function Intro adv-bearer and gate oner Principle Command Interact	oduction	Adv respons	e packet .				<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> <li>154</li> <li>154</li> <li>155</li> </ul>
9	<ul> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> <li>10.2</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c Visione Provis 10.1.1 Provis 10.2.1 10.2.2 10.2.3	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction file introduction . <b>(Gateway) Project</b> oner Function Intro adv-bearer and gat oner Principle Command Interact Timing Sequence of Timing Sequence of the sequence of the sequence of the sequence of the sequence of the sequence of the sequence of the sequ	oduction	Adv respons	e packet .				<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> <li>154</li> <li>154</li> <li>155</li> <li>158</li> </ul>
9	<ul> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> <li>10.2</li> <li>10.3</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c visione Provis 10.1.1 Provis 10.2.1 10.2.2 10.2.3 app.c	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction . <b>(Gateway) Project</b> oner Function Intro adv-bearer and gate oner Principle Command Interact Timing Sequence of ile introduction	oduction	Adv respons	e packet .				<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> <li>154</li> <li>154</li> <li>155</li> <li>158</li> <li>159</li> </ul>
9	<ul> <li>8.9</li> <li>825</li> <li>9.1</li> <li>9.2</li> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> <li>10.2</li> <li>10.3</li> <li>10.4</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c Visione Provis 10.1.1 Provis 10.2.1 10.2.2 10.2.3 app.c	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction . <b>(Gateway) Projec</b> oner Function Intro adv-bearer and gate oner Principle Command Interact Timing Sequence of ile introduction poner operation and	oduction	Adv respons	e packet .				<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> <li>154</li> <li>154</li> <li>155</li> <li>158</li> <li>159</li> <li>160</li> </ul>
9	<ul> <li>8.9</li> <li>825</li> <li>9.1</li> <li>9.2</li> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> <li>10.2</li> <li>10.3</li> <li>10.4</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c Visione Provis 10.1.1 Provis 10.2.1 10.2.2 10.2.3 app.c Provis 10.4.1	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction . file introduction . <b>(Gateway) Project</b> oner Function Introduction adv-bearer and gate oner Principle Command Interact Timing Sequence of ile introduction oner operation and Format of SIG_ME	oduction	Adv respons	e packet .	.         .			<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> <li>154</li> <li>154</li> <li>155</li> <li>158</li> <li>159</li> <li>160</li> <li>160</li> </ul>
9	<ul> <li>8.9</li> <li>825</li> <li>9.1</li> <li>9.2</li> <li>9.3</li> <li>9.4</li> <li>Prov</li> <li>10.1</li> <li>10.2</li> <li>10.3</li> <li>10.4</li> </ul>	8 MESH app_c app.c 9.2.1 9.2.2 9.2.3 9.2.4 9.2.5 9.2.6 app_a light.c Visione Provis 10.1.1 Provis 10.2.1 10.2.2 10.2.3 app.c Provis 10.4.1 10.4.2	test_cmd.c file intr <b>Project Introduct</b> onfig_8258.h ile introduction Customization of <i>P</i> Configuration of F app_event_handle main_loop () user_init() void proc_ui() t.c file introduction . t.c file introduction . <b>(Gateway) Project</b> oner Function Introduction . Command Interact Timing Sequence of ile introduction command Interact Timing Sequence of ile introduction progration and Format of SIG_ME SIG model format	oduction	Adv respons Adv respons	e packet .				<ul> <li>145</li> <li>146</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>147</li> <li>148</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>149</li> <li>154</li> <li>154</li> <li>154</li> <li>154</li> <li>155</li> <li>158</li> <li>159</li> <li>160</li> <li>160</li> <li>161</li> </ul>

		10.4.4 Burn Nodes	162
		10.4.5 Add Light via Provisioner	163
		10.4.6 app_key binding	167
		10.4.7 Light on/off Control	168
		10.4.8 Provisioner Control Flow Chart	170
		10.4.9 Smart Provision	171
		10.4.9.1 Difference between Smart Provision and Normal Networking	171
		10.4.9.2 Principle Decription	171
		10.4.9.3 Function Decription	172
		10.4.9.4 Testing Process	172
11	Mes	h LPN Project Introduction	173
	11.1	LPN Node and Implementation Method	173
		11.1.1 LPN and friend	173
		11.1.2 Friendship Parameters	173
		11.1.3 Establish Friendship	174
		11.1.4 Friendship Message Exchange	175
		11.1.5 Security	176
		11.1.6 Friendship Termination	176
	11.2	Friendship Sleep and Working Mechanism	176
		11.2.1 FN Receive Packet Processing Interface	176
		11.2.2 Processing Interface for Packets Sent by FN to LPN	178
		11.2.3 LPN Packet Processing Interface	180
		11.2.4 FriendShip Sleep Mechanism	183
		11.2.5 Friendship Working Mechanism	183
		11.2.6 Mechanism for LPN to Receive a Destination Address as a Group Number	185
	11.3	Common Parameter Configuration for LPN	185
		11.3.1 Friend Node	185
		11.3.2 Low Power Node	185
	11.4	LPN Demonstration	186
		11.4.1 Hardware	186
		11.4.2 Test method	186
	11.5	app.c file introduction	190
	11.6	mesh_lpn.c file introduction	191
12	Swit	tch Project Introduction	1 <b>9</b> 3
	12.1	Switch function introduction	193
	12.2	Switch principle	193
	12.3	app.c file introduction	193
	12.4	Key Event Detection Process	194
		12.4.1 Code Block	194
	12.5	Switch Engineering Long Press Handling Logic	195
	12.6	Example of Sending Commands Using the Soft_timer Cycle	196
	12.7	Configuration of Switch Part	196
		12.7.1 key table	196
		12.7.2 Configure IOs for Drive Pins and Scan Pins	196
		12.7.3 Turn on/off Light via Switch	197
	12.8	Switch Operation	198
	12.9	Flow chart for Switch RC       Flow       Flow	201

	12.10	OFlow chart for sleep processing	202
	12.1	1 Modify the destination address of button send command	202
	12.12	2IV Index Update Mode for Switch	204
13	Соп	nect with a Platform	205
	13.1	Normal Mode	205
		13.1.1 No OOB provision mode	205
		13.1.2 Static OOB provision mode	205
		13.1.2.1 Light Node Burn Static oob	205
		13.1.2.2 Light node Device uuid	205
		13.1.2.3 User Customized uuid Method	206
		13.1.2.4 Provisioner static oob database	206
		13.1.2.5 Test steps	207
	13.2	Ali Tmall Genies Platform	208
		13.2.1 Configuration	208
		13.2.2 Apply tri-truple from Ali	209
		13.2.3 Use SDK Default tri-truple	209
		13.2.4 Provision via Tmall Genie	209
		13.2.5 Provision via Firmware	210
		13.2.6 Dual Modes of static oob and no oob	210
	13.3	Xiaomi Xiao'ai Platform	210
		13.3.1 Configuration	210
		13.3.2 Certification Data Setting	211
		13.3.3 Provision Test	211
	13.4	Dual Vendor Mode (Tmall Genies and Xiaomi Xiaoai)	211
		13.4.1 Function Introduction	211
		13.4.2 Configuration	212
14	Fact	cory Reset $\ldots$ $\ldots$ $\ldots$ $\ldots$ $\ldots$ $\ldots$	213
	14.1	8258_mesh/8269_mesh Node	213
		14.1.1 Function Introduction	213
		14.1.2 Default trigger action	213
		14.1.3 Method to modify power-on sequence	214
		14.1.4 The function of the previous mesh network can be restored after the reset action is	
		triggered	215
	14.2	Gateway Node + Host Computer	215
	14.3	GATT master dongle + Host Computer	217
	14.4	LPN Node	217
	14.5	Switch Node	217
15	Fast	bind Mode (PROVISION_FLOW_SIMPLE_EN Mode)	218
	15.1	Function Introduction	218
	15.2	Configuration	218
	15.3	Function Demonstration	218
		15.3.1 Firmware Configuration	218
		15.3.2 APP Interface Configuration	219
16	Priv	rate Fast provision Function	220
	16.1	Function Introduction	220
	16.2	Configuration	220
	16.3	Function Demo	220

17	Priv	rate online status function demo	223
	17.1	Function Introduction	223
	17.2	Configuration	223
	17.3	Packet Format	223
	17.4	SIG_MESH_TOOL Firmware Demo	225
18	Telir	nk Proprietary OTA Test Brief	226
	18.1	GATT master dongle OTA for firmware update of BLE directly connected nodes	226
	18.2	OTA OTA where the Gateway node updates its firmware	228
19	Net	work Sharing	230
	19.1	Share Mode of App share from Gateway or GATT Master Dongle	230
	19.2	Share Mode of Gateway or GATT Master Dongle share from App	234
20	Con	trol Nodes via INI Demo	237
	20.1	Provision Device	237
	20.2	2 Configuration Operations	241
		20.2.1 Key add/bind Operation	241
		20.2.2 Subscription Configuration	243
		20.2.3 Publish configuration	243
		20.2.4 Relay/Friend Function Configuration	243
		20.2.5 Heartbeat setting	244
	20.3	Control Operations	244
		20.3.1 Control Generic model Demo	244
		20.3.2 CTL model	246
		20.3.3 HSL model	246
		20.3.4 Vendor model	247
		20.3.5 Gateway Transmit Long Packet to LPN	248
21	Sum	mary of mesh_1.1_feature $\ldots$	251
22	Cert	tify_base_provision_certificate Mode	252
	22.1	Function	252
	22.2	. Test Using the Code's Default Certificate and Compiling It Directly into Firmware $\ldots$ $\ldots$ $\ldots$	252
		22.2.1 Code Configuration	252
	22.3	Testing Ways to Use Newly Generated Certificates	253
		22.3.1 Code Configuration	253
		22.3.1.1 Open a Git_bash Terminal	253
		22.3.1.2 Generate Root Certificates	254
		22.3.1.3 Run Gen-intermediate.bash to Create an Intermediate Certificate	256
		22.3.1.4 Configure Device Certificate Parameters	256
		22.3.1.5 Run Gen-device.bash to Generate the Device Certificate	257
		22.3.1.6 Burn the Certificate into the Device's Flash	258
		22.3.1.7 Codes Described Below	259
23	Rem	note Provision Functional Description and Development Instructions	260
	23.1	Remote Provision Function Introductions	260
		23.1.1 Introduction to Remote_provision Network Interaction Process	260
		23.1.2 Remote Provision Opcode and Flowchart	261
	23.2	Testing Remote Provisioning with the App	264
		23.2.1 Test Conditions	264
		23.2.2 Firmware SDK Code Configuration	264
		23.2.3 App Settings	264

23.2.4 Test Steps	265
23.3 Gateway Remote Provision Host Computer Development Guide	267
23.3.1 Code and Tool Parameter Configuration for Gateway's Remote Provision	267
23.3.2 Phase 1 Network One or More Nodes in Normal pb_adv Style	269
23.3.3 Stage 2 Remote Provision Add Light	270
24 Mesh OTA and Guide for Host Computer Development	278
24.1 Mesh OTA Introduction	278
24.1.1 Mesh OTA Features and Modes	278
24.1.2 Introduction to Mesh OTA Modes and Reference Rates	278
24.1.3 Mesh OTA Firmware Distribution Method	278
24.1.4 Three Role Profiles of Mesh OTA	279
24.1.5 Mesh OTA Silent Upgrade Mode	279
24.1.6 Mods for Mesh OTA	279
24.2 Test Mesh OTA with App	280
24.3 Gateway Mesh OTA	280
24.3.1 Test and Command Sending and Receiving Process	280
24.3.1.1 Code Configuration	280
24.3.1.2 Networking Nodes	282
24.3.1.3 Select New Firmware	283
24.3.1.4 Download New Firmware to Local Gateway Dongle	283
24.3.1.5 Get the Version Information of the Nodes Currently on the Network $\ldots$ $\ldots$	284
24.3.1.6 Send fw_distribution_start_all Command	285
24.3.1.7 OTA Progress Reporting $\ldots$	285
24.3.1.8 Mesh OTA Completion Display Page	286
24.3.1.9 Device Flashes 6 Seconds Slowly	287
24.3.2 OTA Code Flow Summary	288
24.3.3 Gateway OTA Flowchart	288
24.3.4 Mesh OTA Related Commands	289
24.3.4.1 FW_DISTRIBUT_START	290
24.3.4.2 FW_UPDATE_METADATA_CHECK	291
24.3.4.3 CFG_MODEL_SUB_ADD	292
24.3.4.4 FW_UPDATE_INFO_GET	292
24.3.4.5 FW_UPDATE_START	292
24.3.4.6 BLOB_INFO_GET	292
24.3.4.7 BLOB_TRANSFER_START	292
24.3.4.8 BLOB_BLOCK_START	293
24.3.4.9 BLOB_CHUNK_TRANSFER	293
24.3.4.10BLOB_BLOCK_GET	293
24.3.4.11FW_UPDATE_GET	293
24.3.4.12FW_UPDATE_APPLY and FW_UPDATE_CANCEL	293
24.4 Gatt master dongle mode mesh OTA (kma_dongle)	293
24.4.1 Code Configuration	293
24.4.2 Networking Nodes	294
24.4.3 Select New Firmware	295
24.4.4 Get Version	296
24.4.5 OTA Start	297
24.4.6 OTA Finish	300

	301
24.4.8 Check for Success	301
24.5 LPN Mesh OTA	302
24.5.1 LPN Mesh OTA Gateway Mode Operation Procedure	302
24.5.1.1 Code Configuration	302
24.5.1.2 Networking Nodes	303
24.5.1.3 Select New Firmware	303
24.5.1.4 Get Version	303
24.5.1.5 OTA Start	303
24.5.1.6 OTA Finish	304
24.5.2 LPN Mesh OTA Gatt Master Dongle Mode	305
24.5.2.1 Code Configuration	305
24.5.2.2 Networking Nodes	305
24.5.2.3 Select New Firmware	306
24.5.2.4 Get Version	306
24.5.2.5 OTA Start	306
24.5.2.6 OTA Finish	308
24.6 QA	309
24.6.1 What's the Best Way to Distinguish Between Different Equipment Types for OTA? $\dots$	309
24.6.2 Ways to Differentiate between Different Devices?	310
24.6.3 Is it Possible to Confirm the Version before OTA?	310
24.6.4 Can I Revert to a Previous Version?	310
24.6.5 What Needs to Be Done in FW in order to Differentiate between Device Types for	
Separate OTAs?	311
24.6.6 What Needs to Be Done in FW in order to Distinguish FW Version Information for OTA	311
	~ 4 4
	311
<b>25 Subnet Bridge</b>	311 318
25.1 Function Introduction	311 318 318
<b>25 Subnet Bridge</b> 25.1 Function Introduction         25.2 Subnet Bridging Principles	311 318 318 319
<b>25.</b> Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration	<b>311</b> <b>318</b> 318 319 319
25.2 Subnet Bridging Principles	311 318 318 319 319 320
24.7 Appendix Log         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding	<b>311</b> <b>318</b> 318 319 319 320 <b>322</b>
24.7 Appendix Log         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>322</li> <li>322</li> </ul>
24.7 Appendix Log         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles         26.2 Routing Table Types	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>322</li> <li>323</li> </ul>
<ul> <li>25.1 Function Introduction</li> <li>25.2 Subnet Bridging Principles</li> <li>25.3 Configuration</li> <li>25.4 Function Display</li> <li>26 Direct Forwarding</li> <li>26.1 Routing Principles</li> <li>26.2 Routing Table Types</li> <li>26.2.1 Test Firmware Configuration</li> </ul>	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>322</li> <li>323</li> <li>323</li> <li>323</li> </ul>
<ul> <li>24.7 Appendix Log</li> <li>25 Subnet Bridge</li> <li>25.1 Function Introduction</li> <li>25.2 Subnet Bridging Principles</li> <li>25.3 Configuration</li> <li>25.4 Function Display</li> <li>26 Direct Forwarding</li> <li>26.1 Routing Principles</li> <li>26.2 Routing Table Types</li> <li>26.2.1 Test Firmware Configuration</li> <li>26.2.2 Fixed Routing</li> </ul>	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>323</li> <li>325</li> </ul>
25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles         26.2 Routing Table Types         26.2.1 Test Firmware Configuration         26.2.2 Fixed Routing         26.2.3 Non-fixed Routing	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> </ul>
24.7 Appendix Log         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles         26.2 Routing Table Types         26.2.1 Test Firmware Configuration         26.2.2 Fixed Routing         26.2.3 Non-fixed Routing         27 Private-beacon	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> </ul>
<ul> <li>24.7 Appendix Log</li> <li>25 Subnet Bridge</li> <li>25.1 Function Introduction</li> <li>25.2 Subnet Bridging Principles</li> <li>25.3 Configuration</li> <li>25.4 Function Display</li> <li>26 Direct Forwarding</li> <li>26.1 Routing Principles</li> <li>26.2 Routing Table Types</li> <li>26.2.1 Test Firmware Configuration</li> <li>26.2.2 Fixed Routing</li> <li>26.2.3 Non-fixed Routing</li> <li>27 Private-beacon</li> <li>27.1 Application Background</li> <li>27.2 Function Introduction</li> </ul>	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> <li>327</li> <li>327</li> </ul>
24.7 Appendix Edg         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles         26.2 Routing Table Types         26.2.1 Test Firmware Configuration         26.2.2 Fixed Routing         26.2.3 Non-fixed Routing         27 Private-beacon         27.1 Application Background         27.2 1 Moch Drivate Reacon	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> <li>327</li> <li>327</li> <li>327</li> <li>327</li> <li>327</li> </ul>
24.7 Appendix Log         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles         26.2 Routing Table Types         26.2.1 Test Firmware Configuration         26.2.2 Fixed Routing         26.2.3 Non-fixed Routing         27 Private-beacon         27.1 Application Background         27.2 Function Introductions         27.2.1 Mesh Private Beacon         27.2 Private Network Identity and Private Nede Identity	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> </ul>
24.7 Appendix Edg         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles         26.2 Routing Table Types         26.2.1 Test Firmware Configuration         26.2.2 Fixed Routing         26.2.3 Non-fixed Routing         27 Private-beacon         27.1 Application Background         27.2 Function Introductions         27.2.1 Mesh Private Beacon         27.2.2 Private Network Identity and Private Node Identity         27.2.3 Introduction to Opcode	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> <li>327</li> <li>327</li> <li>328</li> <li>329</li> </ul>
24.7 Appendix Log         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles         26.2 Routing Table Types         26.2.1 Test Firmware Configuration         26.2.2 Fixed Routing         26.2.3 Non-fixed Routing         27 Private-beacon         27.1 Application Background         27.2 Function Introductions         27.2.1 Mesh Private Beacon         27.2.2 Private Network Identity and Private Node Identity         27.2.3 Introduction to Opcode	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> <li>327</li> <li>327</li> <li>327</li> <li>327</li> <li>328</li> <li>329</li> <li>330</li> </ul>
24.7 Appendix Eug         25 Subnet Bridge         25.1 Function Introduction         25.2 Subnet Bridging Principles         25.3 Configuration         25.4 Function Display         26 Direct Forwarding         26.1 Routing Principles         26.2 Routing Table Types         26.2.1 Test Firmware Configuration         26.2.2 Fixed Routing         26.2.3 Non-fixed Routing         27 Private-beacon         27.1 Application Background         27.2 Function Introductions         27.2.1 Mesh Private Beacon         27.2.2 Private Network Identity and Private Node Identity         27.2.3 Introduction to Opcode         27.3 Test Steps	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> <li>327</li></ul>
<ul> <li>25.7 Appendix Edg</li> <li>25.1 Function Introduction</li> <li>25.2 Subnet Bridging Principles</li> <li>25.3 Configuration</li> <li>25.4 Function Display</li> <li>26 Direct Forwarding</li> <li>26.1 Routing Principles</li> <li>26.2 Routing Table Types</li> <li>26.2.1 Test Firmware Configuration</li> <li>26.2.2 Fixed Routing</li> <li>26.2.3 Non-fixed Routing</li> <li>27 Private-beacon</li> <li>27.1 Application Background</li> <li>27.2.1 Mesh Private Beacon</li> <li>27.2.2 Private Network Identity and Private Node Identity</li> <li>27.2.3 Introduction to Opcode</li> <li>27.3 Test Steps</li> <li>28 Minor Mesh Enhancements</li> <li>28 I Oncodes Anoneontor Server Model</li> </ul>	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> <li>327</li></ul>
<ul> <li>25.7 Appendix Edg</li> <li>25.1 Function Introduction</li> <li>25.2 Subnet Bridging Principles</li> <li>25.3 Configuration</li> <li>25.4 Function Display</li> <li>26 Direct Forwarding</li> <li>26.1 Routing Principles</li> <li>26.2 Routing Table Types</li> <li>26.2.1 Test Firmware Configuration</li> <li>26.2.2 Fixed Routing</li> <li>26.2.3 Non-fixed Routing</li> <li>26.2.3 Non-fixed Routing</li> <li>27 Private-beacon</li> <li>27.1 Application Background</li> <li>27.2.1 Mesh Private Beacon</li> <li>27.2.2 Private Network Identity and Private Node Identity</li> <li>27.3 Test Steps</li> <li>28 Minor Mesh Enhancements</li> <li>28 11 Application Background</li> <li>28 11 Application Background</li> </ul>	<ul> <li>311</li> <li>318</li> <li>319</li> <li>319</li> <li>320</li> <li>322</li> <li>323</li> <li>323</li> <li>325</li> <li>327</li> <li>321</li> <li>331</li> <li>331</li> </ul>

28.1.2 Function	331
28.1.3 Test Steps	331
28.2 Large Composition Data Models	332
28.2.1 Application Background	332
28.2.2 Function	332
28.2.3 Test Steps	332
28.3 SAR Configuration Models	332
28.3.1 Application Background	332
28.3.2 Function Description	333
28.3.3 Test Steps	333
28.4 EPA(Enhanced Provisioning Authentication)	333
28.4.1 Application Background	333
28.4.2 Function Description	333
28.4.3 Test Steps	335
28.5 On-Demand Proxy Model	335
28.5.1 Application Background	335
28.5.2 Function Description	335
28.5.3 Test Steps	336
28.5.3.1 Testing with APP	336
28.6 Solicitation PDU RPL CFG Models	341
29 Networked Lighting Control(NLC)	343
29.1 Application Background	
29.2 All NLC Profiles	343
29.2.1 NLC Profiles list	343
29.2.2 User Experience when Lights and Sensors work together	343
29.3 Publish_adress Configuration Methods	344
29.4 DICNLCP	345
29.4.1 Function	345
29.4.2 nlc_switch Button	
29.4.3 Element Address	346
29.4.4 nlc_switch Button Functions	
29.4.4.1 nlc_switch button onoff Command Mode	
29.4.4.2 nlc_switch button delta_level Command Mode	
29.4.4.3 nlc_switch button move_level Command Mode	
29.4.4.4 nic_switch button to Switch to on/off Command Mode	
29.4.5 lest Steps	
29.4.5.1 SDK Settings	
29.4.5.2 Add to Network	
29.4.5.4 Key Default Function Test	350
29.5 DSINEEF	
29.5.2 Hardware Introduction	352
29.5.3 Button Functions	
29.5.4 Test Steps	
29.5.4.1 SDK Settings	

	29.5.4.2 Add to Network	. 353
	29.5.4.3 Button Test	. 353
	29.6 BLCNLCP	. 355
	29.6.1 Function Description	. 355
	29.6.2 Hardware Introduction	. 355
	29.6.3 Test Steps	. 355
	29.6.3.1 SDK Settinas	. 355
	29.7 ocssnlcp	. 357
	29.7.1 Function Description	. 357
	29.7.2 Test Steps	. 357
	29.7.2.1 SDK Settings	. 357
	29.7.2.2 Function	. 357
	29.8 ALSNLCP	. 359
	29.8.1 Function Description	. 359
	29.8.2 Test Steps	. 360
	29.8.2.1 SDK Settings	. 360
	29.8.2.2 Function Test ALSNLCP	. 360
	29.9 ENMNLCP	. 361
	29.9.1 Function Description	. 361
	29.9.2 Test Steps	. 361
	29.9.2.1 SDK Settings	. 361
	29.9.2.2 Function Test	. 362
30	) Ellisys Decrypts Mesh Packets	. 363
	30.1 Click Record to Grab the Packet	. 363
	30.2 Fill in Mesh Information for Decryption	. 363
	30.3 Other Methods to Get the Key	. 366
	30.3.1 Provision UART Log of provision flow Via Firmware	. 366
	30.3.2 Via Android App	. 366
	30.3.3 Via iOS App	. 368
	30.3.4 Via JSON File	. 368
31	Operating Instructions for Telink-developed Bluetooth Mesh Decryption and Analysis Tool	. 370
	31.1 Application Background	. 370
	31.2 Operation Procedure	. 370
	31.2.1 Configure Monitor serial port	. 370
	31.2.2 Connect the serial hardware	. 371
	31.2.3 Add Monitor to a Mesh Network	. 371
	31.2.4 Log Parsing	. 372
	31.2.5 Extended Functions	. 373
32	2 Spirit LPN	. 375
	32.1 Function Description	. 375
	32.2 Configuration	. 375
	32.2.1 Set Gateway to Continuous Packet Sending Mode	. 375
	32.2.1.1 Enable Key Detection	. 375
	32.2.1.2 Configure the Numbers of Gateway Sending Packets Continuously	. 376
	32.2.2 Setting the Wake-up Period and Scan Window for LPN	. 376
	32.2.2.1 Setting the Wake-up Period	. 377
	32.2.2.2 Setting the Scanning Window after Wake-up	. 377

	32.3 Function Demonstration	377
	32.4 Platform Access Setting	378
33	Android and iOS APP User Guide	3 <b>79</b>
	33.1 App download	379
	33.2 Device Network	379
	33.2.1 Manual Provision Networking	379
	33.2.1.1 Add Device in Manual Mode	379
	33.2.1.2 Status During Manually Adding Devices	380
	33.2.2 Auto Provision Networking	381
	33.2.3 Rescan Peripheral Devices	382
	33.3 Device Interface	383
	33.3.1 Refresh Device	384
	33.3.2 All on/off	384
	33.3.3 Single Device on/off	384
	33.3.4 CMD Command	384
	33.3.5 Log	385
	33.3.6 Device Setting (Light device)	386
	33.3.6.1 Light Device Control	387
	33.3.6.2 Single Device Group	389
	33.3.6.3 Light Device Settings	390
	33.3.7 Device Setting (Switch Device)	397
	33.3.7.1 Switch Device Control	398
	33.3.7.2 Switch Device Setting $\ldots$	398
	33.4 Group Interface	398
	33.4.1 On/Off Group	399
	33.4.2 Group Setting	399
	33.4.2.1 On/Off Group Devices Individually	400
	33.4.2.2 Lum & Temp	400
	33.4.2.3 Extend Address Control	400
	33.4.2.4 HSL	400
	33.5 Network Interface	401
	33.5.1 Mesh info	402
	33.5.2 Scenes	404
	33.5.2.1 Create Scene	404
	33.5.2.2 Edit Scene	405
	33.5.3 Direct Forwarding	406
	33.5.3.1 Fixed Routing	407
	33.5.3.2 Non-fixed Routing	409
	33.5.4 Mesh OTA	410
	33.5.4.1 Distributor: Phone mode upgrade (App as distributor mode)	410
	33.5.4.2 Distributor: Verify and Apply Mode Upgrade (Directly Connected Nodes as	
	Distributor Mode)	412
	33.5.4.3 Distributor: Verify Only Mode Upgrade (Directly Connected Nodes as Distrib-	
	utor Mode)	414
	33.5.5 Private beacon	415
	33.5.5.1 Config GATT Proxy	415
	33.5.5.2 Private GATT Proxy	416
	-	

	33.5.5.3 Config Node Identity	416
	33.5.5.4 Private Node Identity	416
	33.5.5.5 Config GATT Proxy + Config Node Identity	417
	33.5.5.6 Config GATT Proxy + Private Node Identity	417
	33.5.5.7 Private GATT Proxy + Config Node Identity	418
	33.5.5.8 Private GATT Proxy + Private Node Identity	418
	33.5.5.9 Config Beacon	419
	33.5.5.10Private Beacon	420
	33.5.5.11Beacon + Private Beacon	421
33.6 Setting	g Interface	423
33.6.1	Manage Network	424
	33.6.1.1 Show Detail	425
	33.6.1.2 Share Export	426
	33.6.1.3 Switch To This Network	428
	33.6.1.4 Import mesh	429
	33.6.1.5 Delete Network	433
	33.6.1.6 Clear All Network	433
33.6.2	OOB Database	434
	33.6.2.1 Add an OOB Database Manually	434
	33.6.2.2 Import OOB Database via Txt File	435
	33.6.2.3 Delete OOB Database	435
	33.6.2.4 Use No-OOB Automatically	435
33.6.3	Root Cert	436
	33.6.3.1 Networking by Default Certificate	436
	33.6.3.2 Generate and Import New Certificate for Networking	438
	33.6.3.3 Switch Certify Base Certificates	442
	33.6.3.4 Delete Certify Base Certificate	443
33.6.4	Settings	443
	33.6.4.1 Enable Log	444
	33.6.4.2 Enable Privare Mode (Default Bound)	444
	33.6.4.3 Provision Mode	444
	33.6.4.4 Enable Subscription Level Service model ID	445
	33.6.4.5 Enable DLE Mode Extend Bearer	445
	33.6.4.6 Online Status	445
	33.6.4.7 Reset Settings	445
34 Common A	PI	446
34.1 Provis	ioning Callbacks	446
34.1.1	Provision Event Callback	446
	34.1.1.1 void mesh_node_prov_event_callback(u8 evt_code)	446
	34.1.1.2 u8 is_provision_success()	446
	34.1.1.3 rf_link_light_event_callback (u8 status)	446
34.1.2	Provisioning Message Handle	446
	34.1.2.1 PB_ADV	446
	34.1.2.2 PB_GATT	446
34.2 Proxy	Server API	447
34.2.1	Provision Service	447
	34.2.1.1 Int pb_gatt_Write (void *p)	447

34.2.2 Proxy Service	47
34.2.2.1 Int proxy_gatt_Write(void *p)	47
34.3 Configuration Callbacks API	47
34.3.1 Int mesh_cmd_sig_cfg_appkey_set()	47
34.4 model_enable	47
34.4.1 MD_SAR_EN	47
34.4.2 MD_ON_DEMAND_PROXY_EN	48
34.4.3 MD_OP_AGG_EN	48
34.4.4 MD_LARGE_CPS_EN	48
34.4.5 MD_SOLI_PDU_RPL_EN	48
34.4.6 MD_DF_CFG_SERVER_EN and MD_DF_CFG_CLIENT_EN	48
34.4.7 MD_SBR_CFG_SERVER_EN and MD_SBR_CFG_CLIENT_EN 4	48
34.4.8 MD_REMOTE_PROV	48
34.4.9 MD_PRIVACY_BEA	48
34.4.10MD_BATTERY_EN	48
34.4.11MD_LOCATION_EN	48
34.4.12MD_LEVEL_EN	49
34.4.13MD_DEF_TRANSIT_TIME_EN	49
34.4.14MD_POWER_ONOFF_EN	49
34.4.15MD_SCENE_EN	49
34.4.16MD_TIME_EN	49
34.4.17MD_SCHEDULE_EN	49
34.4.18MD_SENSOR_EN	49
34.4.19MD_MESH_OTA_EN	49
34.4.20MD_LIGHTNESS_EN	49
34.4.21MD_LIGHT_CONTROL_EN	49
34.4.22_IGHT_TYPE_CT_EN	-50
34.4.23_IGHT_TYPE_HSL_EN	50
34.4.24_IGHT_TYPE_XYL	50
34.4.29_IGHT_TYPE_POWER	50
34.4.26MD_PROPERTY_EN	50
34.5 Light CT and RGB PWM Output API	50
34.5.1 Void light_dim_refresh(int idx)	50
34.6 Vendor Model Client and Server API	50
34.7 Firmware Update and Blob Transfer API	-51
35 QA	52

• Telink

## List of Figures

Figure 1.1	File Architecture	29
Figure 1.2	Mesh SDK demo code	32
Figure 1.3	Mesh SDK compiling options	32
Figure 1.4	PID and VID	35
Figure 1.5	ATT user interface	36
Figure 1.6	Packet Transmission Flow	39
Figure 1.7	Packet Reception Flow	40
Figure 1.8	Check global variable via Tdebug	41
Figure 1.9	Tdebug overview	42
Figure 1.10	Read structure variables or arrays	42
Figure 1.11	Read.bin file	43
Figure 1.12	Print level	43
Figure 1.13	Print module	44
Figure 1.14	Set print pin	45
Figure 1.15	Set baud rate	45
Figure 1.16	Choose log module	46
Figure 2.1	FlashMapB85m512K	47
Figure 2.2	FlashMapB91m1M	48
Figure 2.3	RAM Map	49
Figure 2.4	stack_debug_mode	50
Figure 2.5	address_no_retention_bss_end	51
Figure 2.6	lst file	52
Figure 2.7	lst file	53
Figure 2.8	RAM_sort	54
Figure 2.9	RAM_read	55
Figure 2.10	RAM_result	56
Figure 2.11	IRQ_stack_check	57
Figure 2.12	retention_ram_size_overflow	57
Figure 2.13	retention_ram_list	58
Figure 2.14	System Clock & System Timer	59
Figure 3.1	Layered Architecture	63
Figure 3.2	Mesh Topology	68
Figure 3.3	16 bit Address Allocation	68
Figure 3.4	Network PDU Format	69
Figure 3.5	Network PDU Field Definitions	69
Figure 3.6	Access Payload Field	70
Figure 3.7	Opcode Format	70
Figure 3.8	Unprovisioned device beacon PDU	71
Figure 4.1	Hardware connection	76
Figure 4.2	BDT interface	77
- Figure 4.3	Erase Flash	78
- Figure 4.4	Bin file	78
- Figure 4.5	bin file burned into flash	79
- Figure 4.6	Input information	79
5		

🐮 Telink

Figure 4.7	SIG_MESH_TOOL interface	80
Figure 4.8	ScanDev window	80
Figure 4.9	provision window	81
Figure 4.10	Set internal provision success window	81
Figure 4.11	Provision enabled	82
Figure 4.12	Gateway mode log	82
Figure 4.13	GATT Master dongle log	83
Figure 4.14	Click bind_all	84
Figure 4.15	mesh UI	84
Figure 4.16	Configure UART port	85
Figure 4.17	SIG_MESH_TOOL interface	86
Figure 4.18	mesh window	87
Figure 4.19	Node status	87
Figure 4.20	Single node control	88
Figure 4.21	All node control	88
Figure 4.22	Obtain node address	88
Figure 4.23	Allocate one light to multiple groups	89
Figure 4.24		89
Figure 4.25	Configure node parameter with UI $\ldots$ $\ldots$ $\ldots$ $\ldots$ $\ldots$ $\ldots$	90
Figure 4.26	"GetPub_S"	91
Figure 4.27	Command sending log	91
Figure 4.28	"SecNwBc"	91
Figure 4.29	"TTL"	91
Figure 4.30	SDK default value	92
Figure 4.31	"transmit"	92
Figure 4.32	"Relay"	92
Figure 4.33	"Friend"	92
Figure 4.34	"Proxy"	93
Figure 4.35	"Lightness"	93
Figure 4.36	"C/T"	93
Figure 4.37	Return value	93
Figure 4.38	Double click to choose the node	94
Figure 4.39	"set time"	94
Figure 4.40	time set parameters	95
Figure 4.41	Switch between TAI and local time	95
Figure 4.42	PC firmware operate	96
Figure 4.43	Input scene number	97
Figure 4.44	Recall scene	98
Figure 4.45	Delete scene	99
Figure 4.46	Action Set	100
Figure 4.47	Click "id"	101
Figure 4.48	schedule parameter	102
Figure 5.1	Default testable commands	103
Figure 6.1	SDK initialization	104
Figure 6.2	Enable/disable the configuration	104
Figure 6.3	OnPowerUpType.png	105
Figure 6.4	Configure group index	107

🐮 Telink

Figure 6.5 Global variable to get group number	109
Figure 6.6 Heartbeat packet	110
Figure 6.7 Receiving and filtering connectable packet	113
Figure 6.8 RAM_Cost_for_each_node	114
Figure 6.9 extend_ADV format	115
Figure 6.10 Extend_Adv_Option	116
Figure 6.11 Long sleep 40s test	120
Figure 7.1 command callback	126
Figure 7.2 publish function	128
Figure 8.1 Composition data	141
Figure 10.1    adv Provisioner Timing Sequence Chart	155
Figure 10.2 Function Invoking Relationship Chart for the packet Tx Part of Adv-provision $$ .	156
Figure 10.3 Function Invoking Relationship Chart for the Packet Rx Part of Adv-provision .	157
Figure 10.4    gatt provisioner Timing Sequence	158
Figure 10.5 Packet Tx Function Entry of gatt_provision	159
Figure 10.6 Packet Rx Function Entry of gatt_provision	159
Figure 10.7 SIG_MESH_TOOL	161
Figure 10.8 g_all_on	161
Figure 10.9 CMD vender on	162
Figure 10.10 Add light via provisioner	163
Figure 10.11 unprovision beacon	164
Figure 10.12 SetPro Internal	166
Figure 10.13 Provision	167
Figure 10.14 bind_all	168
Figure 10.15 Light on/off control	169
Figure 10.16 Provisioner Control Flow Chart	170
Figure 11.1 Timing Sequence of ReceiveDelay and ReceiveWindow	174
Figure 11.2 Establish friendship	175
Figure 11.3    Friendship Message Exchange	175
Figure 11.4 LPN_get_level	188
Figure 11.5 ONOFF operation on LPN	189
Figure 11.6 mesh_lpn_sleep_prepare	191
Figure 12.1 Switch Burning Connection	199
Figure 12.2 Switch button	200
Figure 12.3 Flow chart for switch RC	201
Figure 12.4 Flow chart for sleep processing	202
Figure 12.5 Switch button	203
Figure 12.6 publication_set_parameters	204
Figure 13.1 Device uuid	205
Figure 13.2 unprovision broadcast package	206
Figure 13.3 Connection successful and print device uuid	206
Figure 13.4 Print device uuid at a scan node	206
Figure 13.5 static oob provision success	207
Figure 13.6 capability data	208
Figure 13.7 Configuration	208
Figure 13.8 Apply tri-truple	209
Figure 13.9 Apply tri-truple	209



Figure 13.10	firmware file $\ldots$	210
Figure 13.11	Xiaomi defined mesh provision mode	210
Figure 13.12	Apply certificate	211
Figure 13.13	Provision method	212
Figure 14.1	gateway reset	216
Figure 15.1	Write PID to device uuid	218
Figure 15.2	Firmware Configuration	218
Figure 16.1	Normal connection	221
Figure 16.2	Click in order	222
Figure 17.1	Packet format	223
Figure 17.2	reference details	224
Figure 17.3	device_status_update	224
Figure 17.4	SIG_MESH_TOOL firmware demo	225
Figure 18.1	BDT tool	226
Figure 18.2	wtcdb tool	227
Figure 18.3	Start OTA	227
Figure 18.4	Select bin file	228
Figure 18.5	Click Gate_ota	229
Figure 19.1	Click Setting	230
Figure 19.2	Click Share	231
Figure 19.3	Click IMPORT	231
Figure 19.4	Click mesh.json	232
Figure 19.5	Click Setting	232
Figure 19.6	Click Share	233
Figure 19.7	Click IMPORT	233
Figure 19.8	Click mesh.json	234
Figure 19.9	JSON file path	235
Figure 19.10	Complete sharing $\sim$	236
Figure 20.1	Connect device	237
Figure 20.2	log info	238
Figure 20.3	Provision parameter setting and device provision	239
Figure 20.4	The data interaction of Provision	240
Figure 20.5	Get device composition data	240
Figure 20.6	Bind	241
Figure 20.7	Bind	241
Figure 20.8	APPKey add command	241
Figure 20.9	Filling data	242
Figure 20.10	D Transmission parameter format reference	243
Figure 20.11	Relay	243
Figure 20.12	2 Friend	243
Figure 20.13	<sup>3</sup> Proxy	244
Figure 20.14	Firmware send successfully	245
Figure 20.15	Broadcast address	245
Figure 20.16	Firmware send successfully	246
Figure 20.17	7 Firmware send successfully	247
Figure 20.18	3 cb_vd_light_onoff_set	249
Figure 20.19	9 tdebug tool	250

🐮 Telink

Figure 22.1	Certificate mode	53
Figure 22.2	Open git bash	54
Figure 22.3	Generate root certificates 2	54
Figure 22.4	Go to the certificates page	55
Figure 22.5	Importing a new certificate	55
Figure 22.6	Importing a new certificate	55
Figure 22.7	Set the newly imported certificate as root certificate	56
Figure 22.8	Creating an intermediate certificate	56
Figure 22.9	Change the device UUID CID PID in gen-device.config	57
Figure 22.10	Generate device certificate	58
Figure 22.11	Burning certificates to flash	58
Figure 22.12	CRC comparison	59
Figure 22.13	CRC comparison	59
Figure 23.1	The Architecture Of Remote Provisioning	261
Figure 23.2	MD_REMOTE_PROV_opcode	62
Figure 23.3	MD_REMOTE_PROV_scan_flow	63
Figure 23.4	MD_REMOTE_PROV_provision_flow 2	63
Figure 23.5	打开 MD_REMOTE_PROV_App	64
Figure 23.6	MD_REMOTE_PROV_App_setting	65
Figure 23.7	MD_REMOTE_PROV_App_provision	66
Figure 23.8	MD_REMOTE_PROV_App_provision_success	67
Figure 23.9	Open MD_REMOTE_PROV	67
Figure 23.10	Open EXTENDED_ADV_ENABLE	68
Figure 23.11	EXTENDED_ADV_ENABLE	68
Figure 23.12	Gateway settings extended broadcast packets	69
Figure 23.13	Setting the gateway mode	70
Figure 23.14	Remote_provision add lights	271
Figure 23.15	No nodes that support the remote_provision feature are selected	271
Figure 23.16	Remote provision nodes	72
Figure 23.17	Double click nodes	73
Figure 23.18	Click prov	74
Figure 23.19	Trigger adding light	75
Figure 23.20	) Provisioning status	76
Figure 23.21	Bind app_key	77
Figure 24.1	Open MD_MESH_OTA_EN	80
Figure 24.2	Open EXTENDED_ADV_ENABLE	281
Figure 24.3	The host computer opens the extend_adv 2	82
Figure 24.4	Network node	83
Figure 24.5	Select new firmware	83
Figure 24.6	Download new firmware to local cache	84
Figure 24.7	Get node information	85
Figure 24.8	OTA progress reporting	86
Figure 24.9	Mesh OTA completed	87
Figure 24.10	ota reboot	87
Figure 24.11	Gateway ota flowchart	89
Figure 24.12	Open MD_MESH_OTA_EN	94
Figure 24.13	Enable server distributor	94



Figure 24.14	mesh_UI	295
Figure 24.15	Select new firmware	296
Figure 24.16	Get version	297
Figure 24.17	start mesh OTA	298
Figure 24.18	verify apply	299
Figure 24.19	verify only	300
Figure 24.20	kma ota finish	301
Figure 24.21	Recover log	301
Figure 24.22	Check for success	302
Figure 24.23	Switch off the friend function	302
Figure 24.24	LPN mesh ota start	304
Figure 24.25	ota success	305
Figure 24.26	LPN mesh ota start	306
Figure 24.27	LPN mesh ota verify apply mode	307
Figure 24.28	LPN mesh ota verify only mode	308
Figure 24.29	ota success	309
Figure 24.30	fw_distribution_start	309
Figure 24.31	fw_distribution_start_all	310
Figure 24.32	start_all_UI	310
Figure 25.1	Hotel subnet map $\ldots$	318
Figure 25.2	Bridging tables	319
Figure 25.3	Open MD_SBR_CFG_SERVER_EN	320
Figure 25.4	Scene display	320
Figure 26.1	Directed Forwarding & Managed Flooding schema	322
Figure 26.2	Fixed routing directangle toggle list interface	324
Figure 26.3	Adding a fixed routing $\sim 0.0000000000000000000000000000000000$	325
Figure 26.4	Non-fixed routing establishment rules	326
Figure 27.1	Value of different beacon packages	327
Figure 27.2	Segment of private_ivi	328
Figure 27.3	Identification type values	328
Figure 27.4	Type value of private	329
Figure 27.5	Type value of private	329
Figure 27.6	Private beacon opcode	330
Figure 28.1	Bind compression	332
Figure 28.2	epa description	334
Figure 28.3	On-Demand Proxy descriptions	335
Figure 28.4	set private beacon	336
Figure 28.5	Disconnection effects	337
Figure 28.6	Send solicitation PDU	338
Figure 28.7	Send solicitation PDU results	339
Figure 28.8	set on demand private	340
Figure 28.9	get on demand private value	341
Figure 28.10	PDU_RPL configuration mods	342
Figure 29.1	pub result	345
Figure 29.2	Switch button introduction	346
Figure 29.3	app	351
Figure 29.4	scene_recall	353

Figure 29.5	app	354
Figure 29.6	Operation_of_a_Light_Lightness_Controller	356
Figure 29.7	pub ocs	359
Figure 29.8	pub als	361
Figure 29.9	pub enm	362
Figure 30.1	Mesh security	363
Figure 30.2	Get mesh key	364
Figure 30.3	Get mesh iv	364
Figure 30.4	Mesh security set	365
Figure 30.5	Provision UART log	366
Figure 30.6	Android key	367
Figure 30.7	iOS key	368
Figure 30.8	json key	369
Figure 30.9	json iv	369
Figure 31.1	mesh_monitor_uart_io_setting	370
Figure 31.2	mesh_monitor_hardware_connection	371
Figure 31.3	mesh_monitor_baudrate_setting	371
Figure 31.4	mesh_monitor_report_format	372
Figure 31.5	mesh_monitor_test_demo	373
Figure 32.1	Open key	375
Figure 32.2	Open demo	375
Figure 32.3	Setting the number of packets sent	376
Figure 32.4	Setting the wake-up period $\ldots$ $\ldots$ $\ldots$	377
Figure 32.5	Setting the scan time	377
Figure 33.1	Manual mode adding devices	380
Figure 33.2	Status display for android manual mode add device process	381
Figure 33.3	iOS device status	381
Figure 33.4	Android auto provision	382
Figure 33.5	iOS auto provision	382
Figure 33.6	Android device reloads device list & rescan and auto-networking	383
Figure 33.7	iOS device reloads device list & rescan & auto-networking	383
Figure 33.8	Android & iOS app device interface	384
Figure 33.9	Android & iOS CMD interface	385
Figure 33.10	) Android & iOS log interface	386
Figure 33.11	The light device setting interface	386
Figure 33.12	2 Android & iOS control interface	387
Figure 33.13	B Light Device Lighting Control interface	388
Figure 33.14	Sensor Device Sensor Control interface	389
Figure 33.15	5 Android & iOS add group interface $\ldots$	390
Figure 33.16	5 Android & iOS settings interface	391
Figure 33.17	7 Android & iOS Device Config	392
Figure 33.18	3 Android & iOS Composition Data	393
Figure 33.19	Android & iOS Scheduler	395
Figure 33.20	O Android & iOS edit Scheduler	395
Figure 33.21	Android device OTA interface	396
Figure 33.22	2 iOS device OTA interface	397
Figure 33.23	Android & iOS switch device setting interface	398

Figure 33.24	Android & iOS Group interface	. 399
Figure 33.25	Android & iOS Group setting	. 400
Figure 33.26	Android & iOS HSL interface	. 401
Figure 33.27	Android & iOS network interface	. 402
Figure 33.28	Android Mesh info interface	. 403
Figure 33.29	iOS Mesh info interface & Steps of adding Netkey / APPkey	. 404
Figure 33.30	Android create Scene interface	. 405
Figure 33.31	iOS create Scene interface	. 405
Figure 33.32	Android scene edit interface	. 406
Figure 33.33	iOS scene edit interface	. 406
Figure 33.34	Directed Forwarding & Managed flooding	. 407
Figure 33.35	Fixed Routing Directangle Toggle list interface	. 408
Figure 33.36	Adding a fixed route	. 409
Figure 33.37	Non-fixed route establishment rules	. 409
Figure 33.38	Introduction to loading methods	. 410
Figure 33.39	Android phone method upgrade steps and upgrade completion interface	. 411
Figure 33.40	iOS phone method upgrade steps & upgrade completion interface	. 412
Figure 33.41	Android verify and apply upgrade steps & upgrade completion interface	. 413
Figure 33.42	iOS verify and apply upgrade steps & upgrade completion interface	. 413
Figure 33.43	Android verify only method upgrade steps & upgrade completion interface	. 414
Figure 33.44	Android verify only method upgrade steps & upgrade completion interface	. 415
Figure 33.45	Open config GATT proxy & broadcast type separately	. 415
Figure 33.46	Open config GATT proxy & broadcast type separately	. 416
Figure 33.47	Open config node identity & broadcast types individually	. 416
Figure 33.48	Open private node identity & broadcast types individually	. 417
Figure 33.49	Config GATT proxy + Config node identity & broadcast type	. 417
Figure 33.50	Config GATT proxy + Private node identity & broadcast type	. 418
Figure 33.51	Private GATT proxy + Config node identity & broadcast type	. 418
Figure 33.52	Private GATT proxy + Private node identity & broadcast state	. 419
Figure 33.53	Light blue APP & ellisys receive beacon packets after opening beacon	. 420
Figure 33.54	Open private beacon after light blue APP & ellisys receive beacon packets	. 421
Figure 33.55	Beacon packets received by light blue APP after opening beacon + private beacon	. 422
Figure 33.56	Beacon packets received by ellisys after opening beacon + private beacon	. 423
Figure 33.57	Android & iOS setting interface	. 424
Figure 33.58	Android & iOS manage network interface	. 425
Figure 33.59	Show detail interface	. 426
Figure 33.60	Android & iOS export json file	. 427
Figure 33.61	Android QR code export	. 427
Figure 33.62	iOS QR code export	. 428
Figure 33.63	Switch to this network	. 429
Figure 33.64	Network list interface import button	. 430
Figure 33.65	Android json file import	. 431
Figure 33.66	iOS json file import	. 431
Figure 33.67	Android QR code import network interface	. 432
Figure 33.68	iOS QR code import network interface	. 432
Figure 33.69	Delete the network interface	. 433
Figure 33.70	Clear all network interface	. 434

Figure 33.71	Add OOB database manually
Figure 33.72	OOB data in TXT format
Figure 33.73	OOB List, delete OOB data
Figure 33.74	Android & iOS cert list interface
Figure 33.75	Open CERTIFY_BASE_ENABLE
Figure 33.76	CERT_TYPE set to CERTIFY_OOB_BY_DEFAULT_CERT
Figure 33.77	Android & iOS manual networking certify tips
Figure 33.78	Android & iOS auto-networking certify tips
Figure 33.79	Generated "root.der" certificate
Figure 33.80	Import root.der certificate for Android
Figure 33.81	Import root.der certificate for iOS
Figure 33.82	Generat intermediate certificates
Figure 33.83	Change the UUID and the corresponding CID and PID
Figure 33.84	Generate device certificate bin file
Figure 33.85	Switch the certify base certificate interface
Figure 33.86	Android & iOS Setting/Settings interface

relink Semiconductor



## **List of Tables**

Table 1.1	Mesh SDK Project Example	32
Table 3.1	Health model related messages	72

Telink Semiconductor



### **1 SDK Overview**

Telink SIG Mesh SDK supplies demonstration code for SIG\_mesh protocol application development, based on which users can develop their own application program.

SDK download method:

Visit Telink Wiki to download the latest Bluetooth Mesh SDK, or click this link to download directly: sig\_mesh\_sdk.zip.

Current projects of the SDK apply to Telink IC 825x/8278/b91m/8269, Telink provides PC tool (sig\_mesh\_tool.exe) connecting master dongle through USB to realize provisioner function, please note, master dongle supports only 8269 SoC series.

For a quick overview and demonstration of basic functionality, please refer to:

BLE SIG Mesh Quick Start

### **1.1 SDK File Architecture**

The file architecture for Telink SIG Mesh SDK includes APP (application) layer and BLE&SIG\_mesh protocol layer. After the SDK project is imported in Telink IDE (please refer to AN\_IDEUG-E1\_Telink IDE User Guide.pdf for project importing, AN\_16063000-E1\_Guide for Adding New Project on Existing SDK.pdf for adding new project), the file structure is shown in figure below, containing the following top-layer folders. There are several main top-level folders: boot, common, drivers, homekit\_src, proj\_lib, stack, and vendor.

### Note:

In the B91m project, enable this macro \_\_\_TLSR\_RISCV\_EN\_\_ in the compiler's pre-compile macro.



### Figure 1.1: File Architecture

- boot: This folder contains bootloader of the SoC, i.e, the compiling process after MCU boot or awake from deep sleep, this is the environment base for later C programs.
- drivers: This folder contains configuration files and driver programs for hardware peripherals related with MCU, like clock, flash, i2c, usb, gpio, uart and etc.
- proj: This folder contains MCU related peripheral driver, such as flash, I2C, USB, GPIO, UART driver, and etc.
- proj\_lib: This folder contains library files necessary for MCU running, for example, BLE stack, RF driver, PM driver. Since this folder is supplied in the form of library files, the source files are not open to users, for example, BT stack library file "liblt\_8269\_mesh.a", library file for SIG\_mesh common node "libsig\_mesh.a", library file for SIG\_mesh low power node "libsig\_mesh\_LPN.a", library file for SIG\_mesh prov.a".
- stack: This folder contains BLE protocol related header files. The sorce files are compiled into library files, and are not open to users.
- vendor: This folder contains user APP-layer code, including:
  - 8267\_master\_kma\_dongle:Firmware used for host test. In combination with host tool(sig\_mesh\_tool.exe)
    of GATT mode, it can act as a provisioner and it's used for demonstration and debugging.
  - common: It mainly contains common modules in mesh/mesh\_lpn/mesh\_provision/mesh\_switch, for example, SIG mesh model processing, LED module, factory initialization module, test command module.
  - mesh/mesh\_gw\_node\_homekit/mesh\_lpn/mesh\_provision/mesh\_switch/ spirit\_lpn have the same structure, which all include "app.c", "app.h", "app\_att.c", "app\_config.h" and "main.c".

Telink

тł



"app.c/app.h" contains initialization and bottom-layer callback function; "app\_att.c" is description of BT ATT table and interface functions; "app\_config.h" defines corresponding macros and declarations in projects; main.c is main function and interrupt function entry.

### 1.1.1 main.c

This includes the main function entry, system initialization related functions, and the infinite loop while(1), it is not recommended to verify this file, please follow the fixed codes.

```
int main (void) {
    FLASH_ADDRESS_CONFIG;
#if PINGPONG_OTA_DISABLE
    ota_fw_check_over_write(); // Copying firmware for non-Pinkpong OTAs
#endif
    blc_pm_select_internal_32k_crystal(); //Select internal 32k rc as 32k counter clock source
    cpu_wakeup_init();//The most basic hardware initialization of the MCU
    int deepRetWakeUp = pm_is_MCU_deepRetentionWakeup(); //Determine whether to wake up from
    \leftrightarrow deep retention.
    rf_drv_init(RF_MODE_BLE_1M); //RF Initialization
    gpio_init(!deepRetWakeUp); //gpio initialization, user configure relevant parameters in
→ app_config.h
    clock_init(SYS_CLK_16M_Crystal);
    if( deepRetWakeUp ){
        user_init_deepRetn ();//Quick initialization of deep retention waking up
    }else{
        user_init_normal ();//ble initialization, whole system initialization, user to set up
    }
    irq_enable();
                         //globalize interruptions
    while (1) {
#if (MODULE_WATCHDOG_ENABLE)
        wd_clear(); //clear watch dog
#endif
        main_loop (); //Includes tasks for ble transceiver processing, low power management,
   mesh and user
\hookrightarrow
    }
}
```

### 1.1.2 app\_config.h

Users configure file to configure all system related parameters, including BLE parameters, GPIO parameters, PM low power management configure parameters and etc.

The definition of each parameter in app\_config.h in later parts of this document when each module is introduced.

### 1.1.3 BLE stack entry

There are 2 entry functions of BLE stack code in Telink BLE SDK.

a) BLE related interrupt handler entry of irq\_handler in main.c file irq\_blt\_sdk\_handler.

```
_attribute_ram_code_ void irq_handler(void)
{
    .....
    irq_blt_sdk_handler ();
    .....
}
```

b) BLE logic and data processing function entry in application file main\_loop blt\_sdk\_main\_loop.

### 1.2 Demo Project

Telink SIG Mesh SDK provides multiple BLE demonstrations.

Users can observe the intuitive effect by running the hardware and software demos. Users can also modify the demo code to complete their own application development.

1	🗁 vendor
	👂 🔁 8267_master_kma_dongle
	🗅 🗁 common
	🖻 🗁 mesh
	b 🔁 mesh_gw_node_homekit
	🖻 🗁 mesh_lpn
	b 🔁 mesh_provision
	b 🗁 mesh_switch
	🖻 🗁 spirit_lpn

4

Figure 1.2: Mesh SDK demo code



### Figure 1.3: Mesh SDK compiling options

The differences of each mesh demo are listed in the following table.

#### Table 1.1: Mesh SDK Project Example

Demo	Vendor folder	Application	Mesh Feature
8258_mesh/ 8269_mesh	.\mesh	CT/HSL light and etc	Relay, friend, proxy
8258_mesh_LPN\ 8269_mesh_LPN	.\mesh_lpn	LPN	LPN, proxy

Demo	Vendor folder	Application	Mesh Feature
8258_mesh_gw\ 8269_mesh_gw	.\mesh_provision	Gateway provisioner	adv provisioner, Relay, friend
8258_mesh _gw_node	.\mesh_provision	Gateway+light node	adv provisioner, Relay, friend, proxy
8258_mesh_switch\ 8269_mesh_switch	.\mesh_switch	Remote control applications	ргоху
8258_spirit_LPN	.\spirit_lpn	Tmall Genie Customized LPN Mode	ргоху
8269_mesh_ master_dongle	.\8267_master \_kma_dongle	ТооІ	GATT provisioner

- 8269\_mesh\_master\_dongle compiling option: supports GATT provisioner function, no rely nor friend function. Firmware uses this for host test, together with GATT mode, the firmware can act as a provisioner to demonstrate and debug.
- 8258\_mesh, 8269\_mesh compiling option: compiling project of normal SIG Mesh nodes, can be configured by provisioner, supports relay, friend, proxy function, no provision function.
- 8258\_mesh\_LPN, 8269\_mesh\_LPN compiling option: compiling project of LPN MESH nodes, receive message via friendship, does not support relay, friend, provision functions. It supports proxy function, but only communicates between the GATT master and LPN, does not forward commands sent by the app to other nodes.
- 8258\_mesh\_gw, 8269\_mesh\_gw compiling option: compiling project of gateway provisioner nodes, supports adv provisioner, can configure other nodes, supports relay, friend functions.
- 8258\_mesh\_switch, 8269\_mesh\_switch compiling option: compiling project of (switch) MESH nodes. To lower power consumption, after the switch is provisioned, it send message with other receiving. Does not support relay, friend function.
- 8258\_gw\_node compiling option: supports functions of both gateway adv provisioner and mesh node. Like a gateway, it can establish its own network and add other unprovision nodes. It can also be provisioned by other provisioners.
- 8258\_spirit\_LPN compiling option: self-defined LPN mode of TMALL Genies.

### 1.3 LIGHT\_TYPE\_SEL Introduction

This macro is used to choose the pre-configured light types.

#define	LIGHT_TYPE_NONE
#define	LIGHT_TYPE_CT
#define	LIGHT_TYPE_HSL
#define	LIGHT_TYPE_XYL
#define	LIGHT_TYPE_POWER

#define LIGHT_TYPE_CT_HSL	5
#define LIGHT_TYPE_DIM	6 // only single PWM
#define LIGHT_TYPE_PANEL	7 // only ON/OFF model
<pre>#define LIGHT_TYPE_LPN_ON_OFF_LEVEL</pre>	8 // only ON/OFF, LEVEL model
#define TYPE_TOOTH_BRUSH	9
<pre>#define LIGHT_TYPE_NLC_CTRL_CLIENT</pre>	10
//#define LIGHT_TYPE_NLC_BLC	<pre>// set NLCP_BLC_EN to 1 to enable Basic Lightness Controller</pre>
↔ Mesh Profile.	
#define LIGHT_TYPE_NLC_SENSOR	11

### LIGHT\_TYPE\_CT:

CT is the abbreviation of color temperature light, and the corresponding product is color temperature light, contains color temperature related model, for example, Light CTL Server, Light CTL Setup Server, Light CTL Temperature Server, and corresponding extend model, such as Generic On/off Server, Generic Level Server, Light Lightness Server and etc.

### LIGHT\_TYPE\_HSL:

The corresponding product is RGB light, contains Light HSL Server, Light HSL Hue Server, Light HSL Saturation Server, Light HSL Setup Server and corresponding extend model, for example, Generic On/off Server, Generic Level Server, Light Lightness Server and etc.

### LIGHT\_TYPE\_XYL:

The corresponding product is XYL light, contains Light xyL Server, Light xyL Setup Server and corresponding extend model, for example, Generic On/off Server, Generic Level Server, Light Lightness Server and etc.

### LIGHT\_TYPE\_POWER:

The corresponding product is power adapter, contains generic Power Level Server, Generic Power Level Setup Server and corresponding extend model, for example, Generic On/off Server, Generic Level Server and etc.

### LIGHT\_TYPE\_CT\_HSL:

The corresponding is CT light + HSL Light, contains CT and HSL related model, and Generic On/off Server, Generic Level Server, Light Lightness Server. CT light use the same lightness and on/off parameter with HSL light. Only one light is lighted at one time.

### LIGHT\_TYPE\_DIM:

The corresponding light is dimming light, contains Light Lightness Server, Light Lightness Setup Server and the corresponding extend model, for example, Generic On/off Server, Generic Level Server.

### LIGHT\_TYPE\_PANEL:

The corresponding product is switch panel, which is the server, controlled by instruments like app and executing on/off switch. The default switch number is 3(defined by LIGHT\_CNT).

### LIGHT\_TYPE\_LPN\_ONOFF\_LEVEL:

The corresponding product is LPN equipment, contains Generic On/off Server model by default, and mesh OTA model is disabled. This is mainly for demo LPN function.

### Note:

- When use LPN equipment, 825x retention RAM size should not exceed 32K.
- All models in node will appear in composition data (global variable model\_sig\_cfg\_s\_cps).

### TYPE\_TOOTH\_BRUSH:

The corresponding product type is a customized product.

### LIGHT\_TYPE\_NLC\_CTRL\_CLIENT:

The corresponding product type is the NLC feature for Mesh V1.1, as detailed in section DICNLCP.

#### LIGHT\_TYPE\_NLC\_SENSOR:

The corresponding product type is the NLC feature for Mesh V1.1, as detailed in section OCSSNLCP, ALSNLCP and ENMNLCP.

### 1.4 Version ID(VID) and Product ID(PID) Configuration

Configure file: vendor -> common -> version.h, this file will be also used in compiling codes.

For example:

#define	MESH_PID_SEL	(LIGHT_TYPE_SEL)	
#define	MESH_VID	(VERSION_GET(0x33, 0x30))	
#define	FW_VERSION_TELINK_RELEASE	(VERSION_GET(0x33, 0x30))	

1) PID and VID in Composition data are defined by MESH\_PID\_SEL, MESH\_VID

2) The 3rd to 6th bits in firmware file is the PID and VID here.

8258_mesh.bin × PID VID																		
						•												
	Q	1	2	3	4	Ş	6	7	Ş	9	ą	þ	ç	þ	ę	f		
0000000h:	26	80	01	00	33	30	5D	01	4B	4E	4C	54	B0	02	88	00	;	&€30].KNLT??
0000010h:	AE	80	00	00	00	00	00	00	34	6B	02	00	00	00	00	00	;	畝4k

Figure 1.4: PID and VID

3) It showed in ATT UI of general APP in the following way:

CONNECTED NOT BONDED	CLIENT	SERVER	:
Generic Access UUID: 0x1800 PRIMARY SERVICE			
Device Information UUID: 0x180A PRIMARY SERVICE			
PnP ID UUID: 0x2A50 Properties: READ			+
Firmware Revision S UUID: 0x2A26 Properties: READ Value: 3030	String		+
PID+VID+FW_VERSION Unknown Service	_TELINK_REL	EASE	

Figure 1.5: ATT user interface

MESH\_PID\_SEL(PID): general APP will show information in ASCII, so "0x00 0x01" is not visible. Users need to verify it to their own PID.

MESH\_VID(VID): "0x33, 0x30" is shown as "30" in ASCII. Users need to verify it to their own PID.

FW\_VERSION\_TELINK\_RELEASE: "0x33, 0x30" is shown as "30" in ASCII. This is the version ID when Telink release the SDK, users should not verify this.

4) Users can verify MESH\_PID\_SEL and MESH\_VID according to their own requirement.

### 1.5 Mobile App Introduction

### 1.5.1 App Installation

### 1.5.1.1 Android App

Get the installation package in the SDK development kit:

 $\label{eq:linkbleMesh} telinkbleMesh\telin$ 

Or download it through the app "Telink Apps".


# 1.5.1.2 iOS App

Get it in the App Store by searching telinksigmesh. Or download it through the app "Telink Apps".

## 1.5.1.3 App Operating Instructions

Refer to the chapter Android and iOS APP User Guide.

# 1.6 Mesh Application Packet Tx/Rx Processing

## 1.6.1 Packet Transmission Function

#### Packet Transmission between Nodes

Data transmitted by invoking the function "mesh\_tx\_cmd2normal\_primary()" follows SIG mesh protocol. Commands such as "access\_cmd\_on/off ()" are derived by assembling the "mesh\_tx\_cmd2normal\_primary()".

Developers can enable the function "sim\_tx\_cmd\_node2node()" (it's masked by default) to demonstrate command transmission. The effect is: After power on, "ON" command and "OFF" command are automatically and alternately sent with the interval of three seconds. This function will be introduced in detail in subsequent section.

Be careful that whether the sending is successful:

(1) After executing the packet sending function, you need to judge the return value of the function, if it is 0, it means success, otherwise it fails, the corresponding error code, see tx\_errno\_e in the SDK for details:

enu	<pre>n tx_errno_e{</pre>	
	TX_ERRNO_SUCCESS	= 0,
	TX_ERRNO_DEV_OR_APP_KEY_NOT_FOUND	= 1,/* device key or app key not found */
	TX_ERRNO_GET_UT_TX_BUF_FAIL	<pre>= 2,/* get the upper layer tx buffer fail */</pre>
	TX_ERRNO_ADDRESS_INVALID	= 3,/* source address or destination address invalid */
	TX_ERRNO_PAR_LEN_OVER_FLOW	= 4,/* parameters length > 378 */
	TX_ERRNO_TX_BUSY	= 5,/* segment busy, reliable busy, */
	TX_ERRNO_TX_FIFO_FULL	<pre>= 6,/* tx fifo full: mesh_adv_cmd_fifo_(normal message)</pre>
$\hookrightarrow$	or mesh_adv_fifo_fn2lpn_(message fro	om friend to LPN)*/
	TX_ERRNO_PAR_LEN_LPN_CTL	= 7,/* All transport control messages originated by a
$\hookrightarrow$	Low Power node shall be sent as Unse	egmented */
	TX_ERRNO_IV_INVALID	= 8,/* have not get iv index after import JSON */
	TX_ERRNO_ALL_OTHER_ERR	= -1,/* default error */
٦.		

(2) In the case of sending segment packet, if it fails in the middle of sending the packet, the callback for sending failure is "mesh\_seg\_block\_ack\_cb()", see "st\_block\_ack\_t" for the error type. For example, "ST\_BLOCK\_ACK\_BUSY" indicates that the receiver is receiving a segment packet from another node and has not finished. At this time, the client can delay for a period of time according to the situation, and then the application layer will do the retransmission processing.

<sup>};</sup> 

#### typedef enum{

```
ST_BLOCK_ACK_RX_ALL = 0, // RX node has received all segments.
ST_BLOCK_ACK_MISSING = 1, // RX node only received some segments.
ST_BLOCK_ACK_BUSY = 2, // RX node is receiving another segment flow, so current tx
segment flow should be stopped and retry later.
ST_BLOCK_ACK_TIMEOUT = 3, // tx segment flow timeout.
ST_BLOCK_ACK_UNKNOW = 4, //
}st_block_ack_t;
```

(3) Before sending a packet, you can determine if it is currently in the tx busy state. Use is\_busy\_mesh\_tx\_cmd() to determine this. You can also call the packet sending interface directly first, and then look at the return value.

#### Master Packet Transmission from Directly-Connected Node to Master

Call bls\_att\_pushNotifyData(), please refer to <AN\_17092701\_Telink 826x BLE SDK Developer Handbook> section 3.4.3.10 for more details, this method is used to send the data in any customised format by the customer. However, there is no mesh function, so it is not recommended to use.

#### Note:

 New UUID should be introduced when employ this method, otherwise it may conflict with current UUID protocol. Users can define new UUID in my\_Attributes\_provision[]/my\_Attributes\_proxy[]/ my\_Attributes[] to define BLE service.

## 1.6.2 Packet Transmission Flow

Note: red font shows library functions.





Figure 1.6: Packet Transmission Flow



# 1.6.3 Packet Reception Flow



Figure 1.7: Packet Reception Flow

# 1.6.4 Packet Reception Callback Function Introduction

# generic model:

The interface of generic model is in the file "vendor/common/generic\_model.c". For callback function, please see the structure "mesh\_cmd\_sig\_func[]". For example, generic on message command. After this command is received, as specified in the packet reception flow, procedure will finally flow to the "mesh\_cmd\_sig\_g\_on/off \_set()", in which user implements the effect of turning on/off light or setting



gradient parameter. The gradient effect is processed in the "light\_transition\_proc" and the processing interval is LIGHT\_ADJUST\_INTERVAL(20ms).

#### vendor model:

The interface of vendor model is in the file "vendor/common/vendor\_model.c". Refer to "mesh\_cmd\_vd\_func[]". User can add vendor command as needed. If such command is received, as specified in the SIG mesh spec, procedure will flow to corresponding callback function, for example, "cb\_vd\_key\_report()".

## 1.6.5 SIG\_mesh Channel

The SIG\_mesh supports two types of communication channels:

adv-bearer: Implement mutual communication based on advertising mechanism, no need to establish BLE connection, the communication channel is the standard 37/38/39.

gatt-bearer: Implement communication based BLE connection, the communication channel is 1-36.

# 1.7 Telink Debug Method Introduction

## 1.7.1 Tdebug Tool Debugging

This is a stable/reliable debugging method with no effect of MCU performance, it can check/verify global variables on real time; it can also check running status of functions. To check if a function is executed or not, user can define a global variable, then count, and by checking the global variable via Tdebug, user can know if the function is executed or not. For example:

Арр.с	00287: void main_loop ()
🗮 include "//m .	00288: {
🗱 include "//pi	00289: static u32 tick loop;
# include "//pi # include "//pi	00290: tick_loop ++;
<pre># include "//pi # include "//pi</pre>	00291:

Figure 1.8: Check global variable via Tdebug

Tdebug is described as following, please check "Help" -> "User guide" in BDT tool for detail.

#### Note:

Right click in the 8th step to get this manual, as shown in figure below.

Sort the 9th step by name, then find tick\_loop (tick\_loop is static variable, the name maybe duplicate in the codes, so the compiler add a suffix of .12397 to distinguish), right click, then click Refresh, it will read all global variables and refresh, the value of tick\_loop will refresh, too.

If you want to change global variables, change in value chart, then press enter. To read back, right click, then press Refresh.

Telink Burning and De	ebugging To	ool (BDT)		
File View Tool Help			F	
■ 8258 • 1 EVK •	Setting	🔎 Erase	O <u>D</u> ownload <u>↓</u> <u>A</u> ctiv	tivate 🕨 Run 📕 Pause 🏶 Step 🔍 PC 🥂 Single step 🗸 🥂 🧟 Reset 😨 manual mode 🗸 🐇 Clear
ьо 1 10	ь0	10	2 sws	602 06 7 Stall 602 88 Start
Ŧ	Download			Tdebug
Variable Name	Addr	Len	Value	۸ ۸
tick light save	44514	4	00000000	TC32 EVK: Swire ok 6
tick_loop.12397	43b40	4	00201da3	
tick_proxy_hash.1173	44494	4	Refresh 8	F3
tick_pub_period_chec	45150	4	<u>S</u> ort by address	
tick_rc24mCal	4300c	4	Sort by name	9
tick_scan.5641	452c0	4	00000000	
tick_scan.6527	452d0	4	d6ff1db0	
tid_cache_idx.14128	45280	4	00000000	
tl_24mrc_cal	434d4	1	00000080	
txPower_index	452f4	1	00000bf	
ui_ota_is_working	4454f	1	00000000	
update_err_cb	452cc	4	00000000	
uri_dat	43554	64		
uri_dat_len	43594	1	0000030	
uri_hash	43550	4	b37874d9	
use_mesh_adv_fifo_fn	4527f	1	0000000	
vd_onoff_state	451e8	2	00000000	
x.12346	43998	4	000002a	· · · · · · · · · · · · · · · · · · ·
evk device: ok	Fi	le Patifice	\ble_lt_mesh\ble_lt_me	nesh\ble_lt_mesh\8258_mesh\bin>3 verion : 5.4.1

Figure 1.9: Tdebug overview

To read structure variables/arrays longer than 4 bytes but shorter than 1K bytes, click "..." in 1, and the read value will show in 2.

Variable Name	Addr	Len	Value	⊴
ll_module_pm_cb	430c4	4	000012a9	TC32 EVK: Swire ok!
ll_push_tx_fifo_handle	43154	4	0001c8e9	
mesh_adv_cmd_fifo	43914	9		Q43914: 20 00 08 2f 2f 4c 47 84 00 2
mesh_adv_cmd_fifo_t	4474c	256	1	Total Time: 22 ms
mesh_adv_fifo_fn2lpn	4398c	9		

Figure 1.10: Read structure variables or arrays

If it is longer than 1K bytes, a file will be generated and saved to Telink Burning and Debugging Tool -> config -> user -> Read.bin in BDT tool.



Variable Name	Addr	Len	Value	^
blt_notify_fifo_b	4484c	2304	$\bigcirc$ 1	1024 bytes have finished!
blt_ota_finished_flag	4449b	1	00000000	2048 bytes have finished!
blt_ota_finished_time	444c0	4	0000000	2001 Sytes have finished.
blt_ota_start_tick	45490	4	0000000	All 2304 bytes have been saved!
blt_ota_terminate_flaç	444a0	1	0000000	lotal lime: 152 ms
blt ota timeout us	43630	4	01c9c380	

Figure 1.11: Read.bin file

# 1.7.2 Log Print Debugging

The SDK versions after 2.9 support log print via uart output port simulated by GPIO, the default speed is 1Mbps. Please be noted, to guarantee the correctness of log output, when run log output, irq\_disable() is executed by default, so too much log data will slow down MCU performance and RF packet processing, as well as make the mesh unstable, thus harm the debugging procedure. So please use log output as little as possible, and shut as many log as possible after the debug is finished.

For log, you can configure print level(TL\_LOG\_LEVEL) and print module(TL\_LOG\_SEL\_VAL).

To rule out unnecessary logs in firmware by default, TL\_LOG\_LEVEL is set to TL\_LOG\_LEVEL\_ERROR, only print level less than or equal to TL\_LOG\_LEVEL\_ERROR will be printed. TL\_LOG\_LEVEL\_LIB is for printing library codes, or important non-library-code log. TL\_LOG\_LEVEL\_USER is for user, and is not in library. It is recommended to use LOG\_USER\_MSG\_INFO() for printing. To use LOG\_MSG\_INFO(), please verify TL\_LOG\_LEVEL.

<pre>#define #define #define #define #define #define #define #define #define #define</pre>	TL TL TL TL TL TL TL	LOG LOG LOG LOG LOG LOG LOG	LEVEL LEVEL LEVEL LEVEL LEVEL LEVEL LEVEL	DISABLE USER LIB ERROR WARNING INFO DEBUG MAX	0 1U 2U 3U 4U 5U 6U TL	// never use in // it will not be
		_100_				

#define TL\_LOG\_LEVEL

# TL\_LOG\_LEVEL\_ERROR

#### Figure 1.12: Print level

Print module (i.e., TL\_LOG\_SEL\_VAL): to print this module, the corresponding module like TL\_LOG\_USER should be included.

typedef enum{	
TL_LOG_MESH	= 0,
TL LOG PROVISION	= 1,
TL LOG LOWPOWER	= 2,
TL_LOG_FRIEND	= 3,
TL_LOG_PROXY	= 4,
TL_LOG_GATT_PROVISIO	$\mathbf{N} = 5,$
TL_LOG_WIN32	=6,
TL_LOG_GATEWAY	= 7,
TL_LOG_KEY_BIND	= 8,
TL_LOG_NODE_SDK	= 9,
TL_LOG_NODE_BASIC	= 10,
TL_LOG_REMOTE_PROV	= 11,
TL_LOG_CMD_RSP	,
TL_LOG_COMMON	,
TL_LOG_CMD_NAME	1
TL_LOG_NODE_SDK_NW_U	JT ,
TL_LOG_IV_UPDATE	,
TL_LOG_USER	, // never use in library.
TL_LOG_MAX,	
} ? end printf_module_enum	? printf module enum;

## Figure 1.13: Print module

The log will be printed only when the corresponding print level and print module meet the requirements.

Note:

There is a maximum length limit on the length of the data to be printed, when exceeded, the data will be truncated. If you want to avoid truncation, you can change the size of the log\_dst[] array.

#### Log Printing Setup

Step 1 Define HCI\_LOG\_FW\_EN as 1

PRINT\_DEBUG\_INFO means to use GPIO to simulate UART, and it can only support TX UART, but not RX UART.

Step 2 Set print pin



🗅 差 🖪 🖁 🖳	😂 🛛 🗶	🖻 🛍 🕹	2 🖸 🛍	<b>1</b> 99 99 68	🛯 🕬		🔶 🔎 🗐	🕘 🗘 🗓	] 🎗 🗎 🖽 🛛	I 2 G    1
App_config_8258.h	000 000 000	6: <b>#if</b> WIN 7: #define 8: <b>#else</b>	HCI_ACCES	s HC	I_USE_USB					
<pre>pragma once pif defined(_cplu </pre>	000 000 000	9: #define 0: <b>#endif</b> 1:	HCI_ACCES	S HC	I_USE_USB					
<pre># endif include "//v) PCBA_8258_DONGLE PCBA_8258_C1T139;</pre>	000	2: <b>#if (HC</b> 3: #define 4: <b>#endif</b>	I_ACCESS== UART_GPIO	HCI_USE_U _SEL	ART) UART	_GPI0_82	58_PB0_PB1			
PCBA_8258_C1T139, PCBA_8258_SEL ≡ VSER CONFIG DEF:	0000 0000 0000	5: 6: #define 7: <b>#if HC</b>	HCI LOG F	WEN 1		_		-		
<pre>#LOG_RT_ENABLE_ # ID_VENDOR # ID_PRODUCT_BASE</pre>	0000	8: #define 9: #define 0: <b>#endif</b>	DEBUG_INF PRINT_DEB	O_TX_PIN UG_INFO			GPIO_PB2 1			
STRING_VENDOR	000	1: 2: #define		FØ						

Figure 1.14: Set print pin

Step 3 Set the baud rate. Default we use 1 Mbps.

When simulate UART output by IO, the interrupt is disabled (SIMU\_UART\_IRQ\_EN=1), so the speed of log is the fast the better under this mode, do not reduce band rate.

#### Note:

Some USB adapter board may have serious mis-alarming when the speed is under 1Mbps rate, to avoid this, users can reduce UART speed (which will reduce log printing speed), or change USB adapter, such as model CH340G.

Myprintf.h	00020: *
	00021: ************************************
🗱 ifndef MYPRINTF_]	00022: <b>#ifndef</b> MYPRINTF H
SIMU_BAUD_1152	00023: #define MYPRINTF H
SIMU_BAUD_2304	00024:
BAUD_USE	00025: #define SIMU BAUD 115200 115200
uart_simu_send	00026: #define <b>SIMU BAUD 230400</b> 230400
∰ endit	00027: #define <b>SIMU BAUD 1M</b> 1000000
	00028:
	00029: #define BAUD USE SIMU BAUD 1M
	00030: #define SIMU UART IRQ EN 1
	00031:
	00032:

#### Figure 1.15: Set baud rate

#### **Step 4** Choose log module

In the grading log definition, besides the above print level, there is log module. To print log, print level and log module should both be set to the right value.

To simplify log in Demo SDK, for TL\_LOG\_SEL\_VAL, only TL\_LOG\_USER is enabled, other log are disabled.

TL\_LOG\_USER is not be called anywhere by default, to add print, user may run the following commands.

LOG\_USER\_MSG\_INFO(pbuf, len, format,...), where:

- pbuf: for transferring a buffer into character and printing. If there is no, set this to 0.
- len: length of pbuf.



If other API is needed, for example, LOG\_MSG\_INFO, check TL\_LOG\_LEVEL first.

	· V 바. 예너의 아프 44 8 8 8 8 V 프 10 10 4 4 4 7 2 12 11 11 2 12 1 1 1 1 1 12 12 1
App mesh.h	01524: IL LOG NODE SDR NW_01,
. defe	01525: IL LOG IV UPDATE ,
	01526: IL LOG USER , // never use in Library.
tf DEBUG PR	01527: IL_LOG_MAX,
TL LOG SI	01528; } } end printf_module_enum ? printf_module_enum;
🗰 🧱 else	01529: #define MAX_MODULE_STRING_CNT 20
TL_LOG_S1	01530:
🛱 endif	01531: #if WIN32
🛱 else	01532: #define MAX_STRCAT_BUF 1024
# MAX_STRUAT_	01533: <b>#if</b> DEBUG_PROXY_FRIEND_SHIP
	01534: #define TL_LOG_SEL_VAL (BIT(TL_LOG_COMMON) BIT(TL_LOG_LOWPOWER) BIT(TL_LOG_FRIEND) BIT(TL_LOG_CMD_RSP) BIT(TL_LOG_IV_UPDATE))
	01535: #else
erse mandif	01536: #define TL LOG SEL VAL (BIT(TL LOG PROVISION) BIT(TL LOG GATT PROVISION) BIT(TL LOG GATEWAY) \
endif	01537:  BIT(TL LOG COMMON) BIT(TL LOG KEY BIND) BIT(TL LOG CMD RSP) BIT(TL LOG CMD NAME) \
if (TL LOG LEV	01538: BIT(TL LOG WIN32) BIT(TL LOG IV UPDATE) BIT(TL LOG NODE BASIC) BIT(TL LOG REMOTE PROV))
LOG_MSG_ERR	01539: #endif
🗰 else	01540: #else
M LOG_MSG_ERR	01541: // just for node part
🗰 endif	01542: #define MAX STRCAT BUE 48
TE (TL_LDG_LEV	
M LUG_MSG_WAR	01544: #define TL LOG SEL VAL BIT(TL LOG USER)//(BIT(TL LOG NODE SDK)/BIT(TL LOG FRIEND)/BIT(TL LOG IV UPDATE)) // BIT(TL LOG NODE SDK NW UT)
M LOG MSG WAR	01545: #else
endif	01546: #define TL LOG SEL VAL (BIT(TL LOG PROVISION) BIT(TL LOG FRIEND) BIT(TL LOG NODE SDK) BIT(TL LOG NODE BASIC))
if (TL LOG LEV	01547: #endif
M LOG MSG INF	01548: <b>#endif</b>
🗱 else	01549:
LOG_MSG_INF	01550: #if (TL_LOG_LEVEL >= TL_LOG_LEVEL_ERROR)
if (TL_LOG_LEV LOG_MSG_INF selse LOG_MSG_INF	01547: #endif 01548: #endif 01549: #if (TL_LOG_LEVEL >= TL_LOG_LEVEL_ERROR)

Figure 1.16: Choose log module

**Step 5** To open pre-set debug log, set TL\_LOG\_SEL\_VAL, for example:



# 2 MCU Basic Modules

This part introduces mesh related information, for detail, please refer to MCU Basic Modules in AN\_19011501-C2\_Telink Kite BLE SDK Developer Handbook.

# 2.1 Flash and RAM map

# 2.1.1 Flash Map Introduction

Take the default FlashMapB85m512K and FlashMapB91m1M of Demo SDK for example:

Refer to sdk/doc/SIGMeshFlashmap\_yyyymmdd.xlsx for the corresponding flash map documentation.

SIG Mesh B85m 512K flash map							
Addr	content	size(K)	Comments				
00000							
	Firmware A (Pingpong OTA)	192					
2FFFF							
30000							
	Sig Mesh Parameters 1	64					
3FFFF							
40000		192					
	Firmware B (Pingpong OTA)						
6FFFF							
70000		24					
	Sig Mesh Parameters 2						
75FFF							
76000							
	Mac (0x76000-0x76005)	4					
76FFF							
77000	Fraguenes Offect (0x77000)						
		4	static OOB is optional				
77FFF							
78000							
	User Parameters	32	For User define				
7ffff							

#### Figure 2.1: FlashMapB85m512K

For more information about "Sig Mesh Parameters 1", please check the SDK's FLASH\_ADR\_MESH\_KEY, FLASH\_ADR\_MD\_CFG\_S ......



For more information about "Sig Mesh Parameters 2", please check the SDK's FLASH\_ADR\_MD\_VD\_LIGHT, FLASH\_ADR\_MD\_SCENE ......

SIG Mesh B91m 1M flash map								
Addr	content	size(K)	Comments					
00000								
	Firmware A (Pingpong OTA)	384						
5FFFF								
60000			should not be used to store					
	reserved for future use	128	parameters unless without flash					
7ffff			protection funtion					
80000								
	Firmware B (Pingpong OTA)	384						
DFFFF								
E0000								
	Sig Mesh Parameters 1	64	refer to table on the right for details					
EFFFF								
F0000								
	Sig Mesh Parameters 2	24	refer to table on the right for details					
F5FFF								
F6000								
	User Parameters	32	For User define					
FDFFF								
FE000	E 011 1 10 1 000							
		4	static OOB is optional					
FEFFF								
FF000								
	Mac (0xff000-0xff005)	4						
FFFFF								

#### Figure 2.2: FlashMapB91m1M

For more information about "Sig Mesh Parameters 1", please check the SDK's FLASH\_ADR\_MESH\_KEY, FLASH\_ADR\_MD\_CFG\_S ......

For more information about "Sig Mesh Parameters 2", please check the SDK's FLASH\_ADR\_MISC, FLASH\_ADR\_RESET\_CNT ......

## 2.1.2 RAM map (8258 64K)

Check ./boot.link for detailed configuration file.



B85m 64K RAM map

- data(retention): contains variables with attribute\_data\_retention prefix, and all global variables whose initial value is not 0, default attribute is retention data; (data is shown in the list file of the B85m SIG mesh SDK, not retention\_data).
- bss(retention): contains variables with attribute\_bss\_retention prefix, and all global variables whose initial value is 0, default attribute is retention bss; ; (bss is shown in the list file of the B85m SIG mesh SDK, not retention\_bss).
- data(no retention): contains global variables with attribute\_no\_retention\_data prefix. (Note that, for some reason, in the list file of the B85m SIG mesh SDK it shows retention\_data instead of no\_retention\_data).
- bss(no retention): contains global variables with attribute\_no\_retention\_bss. Also contains irq\_stack, whose size is defined by IRQ\_STK\_SIZE, default is 0x300, the size is smaller than 0x200 in demo SDK. (Note that, for some reason, in the list file of B85m SIG mesh SDK it shows retention\_bss instead of no\_retention\_bss).
- normal stack: after bss, before 64K RAM. The initial value is Oxfffffffff, to speed up RAM initialization, only 3K RAM will be initialized.

#### Note:

- The end address of "retention bss" needs to be less than 0x848000 when working in retention sleep mode. This is because the maximum retention RAM is 32KB.
- The prefixes attribute\_bss\_retention and attribute\_no\_retention\_bss cannot be used for variables that are not initialized to 0. Otherwise the initialization value is invalid and defaults to 0.

# 2.2 Checking of Stack Overflow and Retention RAM Overflow

Take B85m as example:

# 2.2.1 Checking Method of Stack Overflow

In order for the stack not to overflow, first of all, make sure that the end address of the no retention bss is less than the end address of the RAM, that is, leave space for a normal stack.

The end address of no retention bss can be used with the tdebug Tool to sort variables by address, and the address after the last variable, is the end address.

It can also be calculated by the \*.lst file generated by compiling.

Then, test all functions, check if normal stack and irq\_stack overflow.

# 2.2.1.1 Checking Method of Normal Stack Overflow

The initial value is Oxffffffff. The demo SDK need a stack of around 2.5KB, because all used stack is less than 3KB, to speed up RAM initialization, only 3KB RAM will be initialized. Check by reading RAM, if the 61KB address, i.e., 0x84F400–0x84F403 is not 0xFFFFFFFF, that means the stack is over 3KB, and there is a possibility that the Normal stack has overflowed. To further confirm whether the stack has overflowed, you can initialize the entire normal stack by modifying it as follows:



Figure 2.4: stack\_debug\_mode

Where no\_retention\_bss\_end is the same as the VMA address of the sdk\_version below. Check by reading RAM and if you find that the 4 bytes at the no\_retention\_bss\_end location is not 0xFFFFFFFF, it means that the Normal stack overflowed.

8258	_mesh.ls	:t 🔀 🛛					
19	Sect	ions:					
20	⊟Idx	Name	Size	VMA	LMA	File off	Algn
21	0	.vectors	00000210	00000000	00000000	0008000	2**4
22			CONTENTS,	ALLOC, LOP	AD, READONI	LY, CODE	
23	1	.ram_code	000021d4	00000210	00000210	00008210	2**2
24			CONTENTS,	ALLOC, LOP	AD, READONI	LY, CODE	
25	2	.text	0001a7c0	000023f0	000023f0	0000a3f0	2**4
26			CONTENTS,	ALLOC, LOP	AD, READONI	LY, CODE	
27	3	.rodata	00001d18	0001cbb0	0001cbb0	00024bb0	2**2
28			CONTENTS,	ALLOC, LOP	AD, READONI	LY, DATA	
29	4	.data	00000260	00842d00	0001e8c8	0002ad00	2**2
30			CONTENTS,	ALLOC, LOP	AD, DATA		
31	5	.bss	00003d94	00842f60	0001eb28	0002af60	2**2
32			ALLOC				
33	6	.retention_dat	ta 00000000	c 00846cf4	4 0001eb2	8 0002ecf	4 2**2
34			CONTENTS,	ALLOC, LOP	AD, DATA		
35	7	.retention bs:	s 00000154	00846d00	0001eb34	0002ed00	2**2
36			ALLOC	$\sim$			
37	8	.sdk_version	00000024 🤇	00846e54	0001eb34	0002ee54	2**2
38			CONTENTS,	ALLOC, LOP	AD, DATA		
39	9	.TC32.attribut	tes 000000	10 000000	000000 00	0002ee	78 2**0
40	L		CONTENTS,	READONLY			
41	<b>10</b>	.comment	0000001a	00000000	00000000	0002ee88	2**0
42	L		CONTENTS,	READONLY			

Figure 2.5: address\_no\_retention\_bss\_end

# 2.2.1.2 Checking Method of irq\_stack Overflow

Initialize the value to 0x0000000, check the irq\_stk[] variable by tdebug and observe the usage; if you find that the 4 bytes of irq\_stk[0-3] are non-zero, it means that it has overflowed.

# 2.2.2 RAM Remaining Size Analysis

According to the light\_8258.lst file generated by the compiler (in the same directory as the bin file) the analysis is as follows:

(Note that for some reason the retention\_data shown in the list file for the B85m SIG mesh SDK refers to no\_retention\_data and the retention\_bss refers to no\_retention\_bss)

🗉 🛛 Telink

😸 8258_	_mesh. 1st 🔀
---------	--------------

19	Secti	ions:					
20	⊡Idx 1	Vame	Size	VMA	LMA	File off	Algn
21	0.	vectors	00000210	00000000	00000000	0008000	2**4
22			CONTENTS,	ALLOC, LO	AD, READON	LY, CODE	
23	1.	ram_code	000021d4	00000210	00000210	00008210	2**2
24			CONTENTS,	ALLOC, LO	AD, READON	LY, CODE	
25	2 .	text	0001a7c0	000023f0	000023f0	0000a3f0	2**4
26			CONTENTS,	ALLOC, LO	AD, READON	LY, CODE	
27	3	rodata	00001d18	0001cbb0	0001cbb0	00024bb0	2**2
28			CONTENTS,	ALLOC, LO	AD, READON	LY, DATA	
29	4	data	00000260	00842d00	0001e8c8	0002ad00	2**2
30			CONTENTS,	ALLOC, LO	AD, DATA		
31	5	bss	00003d94	00842f60	0001eb28	0002af60	2**2
32			ALLOC				
33	6	retention_da	ta 0000000	c 00846cf	4 0001eb2	8 0002ecf	4 2**2
34			CONTENTS,	ALLOC, LO	AD, DATA		
35	7	retention_bs	<b>s</b> 00000154	00846d00	0001eb34	0002ed00	2**2
36			ALLOC				
37	8 .	sdk_version	00000024	00846e54	0001eb34	0002ee54	2**2
38			CONTENTS,	ALLOC, LO	AD, DATA		
39	9.	TC32.attribu	tes 000000	10 000000	00 000000	00 0002ee	78 2**0
40	L		CONTENTS,	READONLY			
41	E 10	comment	0000001a	00000000	00000000	0002ee88	2**0
42	L		CONTENTS,	READONLY			

Figure 2.6: lst file

Idx Name: Segment name

Size: Size of bytes occupied by this segment

VMA: Actual running address

LMA: Storage address in flash

Remaining RAM = end address of RAM - address of last byte of no retention bss. (If there is a retention\_data or retention\_bss segment, the end address of the retention\_bss is used)

The 825X RAM start address is 0x840000 and the 8258 RAM size is 64KB, so the RAM end address is 0x850000.

The way to get the address of the last byte of the bss:

For SDKs with an sdk\_version section, the last byte of the bss is equal to the first address of the sdk\_version. The sdk\_version does not take up any RAM space, because it is not loaded from flash to RAM in the cstartup.

The SDKs without an sdk\_version section can be obtained in one of two ways:

1			
	light	8258 let	$\sim$

	Q		2,0,		4,0,		
16	Idx	Name	Size	VMA	LMA	File off	Algn
17	0	.vectors	000001a0	00000000	00000000	0008000	2**4
18			CONTENTS,	ALLOC, LO	AD, READON	LY, CODE	
19	1	.ram code	000016d0	000001a0	000001a0	000081a0	2**2
20		_	CONTENTS,	ALLOC, LO	AD, READON	LY, CODE	
21	2	.text	0000c004	00001870	00001870	00009870	2**4
22			CONTENTS,	ALLOC, LO	AD, READON	LY, CODE	
23	3	.rodata	00000468	0000d874	0000d874	00015874	2**2
24			CONTENTS,	ALLOC, LO	AD, READON	LY, DATA	
25	4	.data	000006d0	00842200	0000dcdc	0001a200	2**2
26			CONTENTS,	ALLOC, LO	AD, DATA		
27	5	.bss	00001869	008428d0	0000e3ac	0001a8d0	2**4
28			ALLOC				
29	6	.TC32.attribu	tes 000000	10 000000	00 000000	00 0001a8	d0 2**0
30			CONTENTS,	READONLY			
31	7	.comment	0000001a	00000000	00000000	0001a8e0	2**0
32			CONTENTS,	READONLY			

#### Figure 2.7: Ist file

(1) bss VMA (i.e. bss start address) + bss size, for example, the above figure is: (0x8428d0 + 0x1869) = 0x844139.

If there is a retention\_data or retention\_bss segment, the end address of the retention\_bss is used, i.e. retention\_bss VMA (i.e., retention\_bss start address) + retention\_bss size.

Note: The retention\_data and retention\_bss here are actually no retention RAM zones, it is for some reason that boot.link needs to name the data and bss as retention zones, as well as the noretention\_data and noretention\_bss are named as retention\_data and retention\_bss zones.

(2) With the BDT tool the variables are sorted by address, the address of the last variable + the size of this variable, as follows. 0x844138 + 1 = 0x844139, (the high "8" has been omitted in the BDT tool)

BDT connect to 1:usb#vid_248a&pid_5320#5&34f5f5c&0&2#{28d78fad-5a12							
Device File View Tool Help							
	Setting (	Erase 🛓	Download + Activate				
b0 10	b0	10	<b>2</b> SWS 602				
Ŧ	Download						
Variable Name	Addr	Len	Value ^				
blt_tx_wptr	4400c	1	0000000				
p_vendor_mesh_node	44010	4	0000000				
master_link_tick	44014	4	0000000				
master_link_state	44018	4	0000000				
switch_rf_tx_once_tim	4401c	4	0000000				
ble_ll_channelTable	44020	40					
ble_mac_tbl_idx	44048	4	0000000				
ble_mac_tbl	4404c	64					
T_CH	4408c	5					
ble_ll_chn	44091	1	0000000				
ble_ll_lastUnmappedC	44094	4	0000000				
ble_mac_flash_blank_i	44098	4	0000000				
ble_ll_channelNum	4409c	4	Refresh F3				
att_buff	440a0	48	Sort by address				
rf_packet_att_rsp	440d0	33	Sort by name				
flash_dat_swap	440f4	64					
pair_ac 🛛 🔿	44134	4	fa6e04c7				
flash_capacity	44138	1	0000013				



According to the above chart, the remaining RAM of this firmware = 0x850000 - (0x8428d0 + 0x1869) = 0xBEC7 = 47.7K.

It should be noted that not all of the remaining RAM can be used directly, for this remaining RAM we need to allocate a section to the normal stack (for the B85m SIG mesh stack it requires a reserved amount of not less than 2.5k, for the B91m SIG mesh stack requires a reserved amount of not less than 4k, for the private mesh stack it requires a reserved amount of not less than 512). Taking the B85m SIG mesh in the above figure as an example, it means that the real RAM that can be used in this firmware is about 47.7-2.5 = 45.2K.

## 2.2.3 Checking Whether The Stack Overflows Using 8258 as An Example

#### RAM Overflow Check:

RAM overflow occurs when the stack usage is beyond the remaining RAM area, and out-of-bounds use of the BSS area or other RAM areas causes the system to crash.

In this system, there are two stack areas, one is the normal stack and the other is the irq stack. The use of the stack is indeterminate and depends mainly on the depth of the function calls that have actually been run and the size of the local variables used in the function.

## 2.2.3.1 Checking Whether The Normal Stack Overflows

- (1) A simple way is: the remaining RAM size (please refer to 1.1 for calculation), the SIG mesh SDK should be larger than 2.5k, and the private mesh should be larger than 512, this is an empirical judgment, no actual read confirmation.
- (2) Another way is: test all the functions once, (the function with trigger reboot cannot be included), and then check if there is any overflow in the normal stack, i.e., after checking the address at the end of the bss if there is still a consecutive area of 0xFF (the initial value of the normal stack is 0xFF), and then check if there is any overflow, if yes, then the RAM is not overflowed. Check the address after the end address of the bss, for example, check the last 4K. (The unused RAM before 3K, that is, before 0x84F400, is not initialized to speed up initialization, so it will be a random number). However, none of our stacks should exceed 3K, so only focus on 3K. The reason for reading 4K is mainly for the address 4K alignment, that is, in the read to the document inside the address of the lower 12bit and the actual address is the same, so it is more convenient to view. View as follows: (press the "Tab" key in the "84f000" control inside), the generated file in the BDT tool is located in the directory ". /BDT/release\_v5.4.4/config/user/", e.g. ". /BDT/release\_v5.4.4/config/user/ log\_17\_17\_48\_addr0x0084f000.bin". Because the BDT tool has to do judgment, when the length of the read data is greater than or equal to 1K, it will be automatically saved as a file, the format of the file name is "log + timestamp + start address".

i놵 Memory Access				emplate-CN.docx - word
8258 👻	EVK	•	CORE -	• (4096) • (A = D
add: 84f000	▼ data:	0	$\smile$	AaBbCcDc AaB AaBbCcD AaBbCcD AaBb
		-		Change Code Text Contents F-Title Note Text Note Tex
BDT connect to 1:usb	#vid_248a&	oid_5320#5	5&34f5f5c&0&2#{28c	28d78fad-5a12-11d1-ae5b-0000f803a8c2}
Device File View Tool	Help			
i□ <u>8</u> 258 ▼ <sup>\</sup> EVK ▼	Setting	🗲 Erase 🔄	Lownload + Activa	ttivate 🕨 Ryn III Pause ັ Step Q PC 🖋 Single step י 🤇 🤆 Reset 😨 manual mode י 🚽 🖉
b0 10	b0	10	2 SWS	S 602 06 ■ Stall 602 88 > Start
<u>+</u>	Download			값입 Tdebug 표 Log windows
Variable Name	Addr	Len	Value	
rnd_m_w	44698	4	42266b9e	
rnd_m_z	4469c	4	14025077	1024 bytes have finished! 2048 bytes have finished!
rng	446a0	4	00013ecc	3072 bytes have finished!
rssi_pkt	446b0	1	00000aa	4096 bytes have finished!
sdk_mainLoop_run_fla	44714	4	00000001	All 4096 bytes have been saved!
sim_onoff.15899	43acb	1	00000001	Iotal Time: 295 ms
sim ty last time 1589	43f2c	4	19cd51f0	

Figure 2.9: RAM\_read

The read file, for example as below, where 84F000+0xF0C is the maximum stack usage for the current run. The stack usage is about 0x100. We see that there are many more consecutive "0xFFFFFFF", so the stack is not overflowing.

log\_17\_17\_48\_addr0x0084f000.bin X

	Q	1	2	3	4	Ş	6	7	ş	9	ą	þ	ç	þ	ę	f		
00000e30h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000e40h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000e50h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000e60h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000e70h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000e80h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000e90h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000ea0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000eb0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000ec0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000ed0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000ee0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000ef0h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	;	
00000f00h:	FF	FF	FF	$\mathbf{FF}$	$\mathbf{FF}$	$\mathbf{F}\mathbf{F}$	$\mathbf{FF}$	FF	$\mathbf{FF}$	FF	$\mathbf{FF}$	FF	04	00	00	00	;	
00000f10h:	3C	2C	84	00	10	04	00	00	00	00	00	00	3C	FF	84	00	;	<,? </td
00000f20h:	00	00	00	00	60	28	84	00	00	00	00	00	D6	7B	00	00	;	`(?謠
00000f30h:	AC	22	84	00	58	FF	84	00	00	00	00	00	00	00	00	00	;	??X ?
00000f40h:	5C	FF	84	00	64	28	84	00	64	28	84	00	D7	25	84	00	;	<pre>\ ?d(?d(???</pre>
00000f50h:	0C	D7	00	00	12	00	00	00	F4	7F	00	00	30	22	84	00	;	.??0"?
00000f60h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;	
00000f70h:	64	28	84	00	78	28	84	00	DO	ЗA	84	00	04	00	00	00	;	d(?x(???
00000f80h:	B8	FF	84	00	10	04	00	00	13	00	00	00	Β1	07	00	00	;	???
00000f90h:	74	25	84	00	00	00	00	00	00	D4	6E	49	01	00	00	00	;	t%?詎I
00000fa0h:	82	40	00	00	01	00	00	00	74	25	84	00	00	00	00	00	;	侤t%?
00000fb0h:	00	D4	6E	49	00	00	00	00	FO	41	00	00	01	00	00	00	;	.詎ェ餉
00000fc0h:	00	00	00	00	00	D4	6E	49	00	00	00	00	DC	42	00	00	;	龍ェ蹷
00000fd0h:	00	00	00	00	20	06	80	00	20	00	80	00	00	00	00	00	;	€€
00000fe0h:	20	00	80	00	00	00	00	00	86	06	00	00	EA	28	00	00	;	.€??
00000ff0h:	5C	02	00	00	00	00	00	00	00	00	00	00	0A	01	00	00	;	\

#### Figure 2.10: RAM\_result

## 2.2.3.2 Checking Whether The Irq Stack Overflows

The irq stack is used by interrupt function, the current configured size is IRQ\_STK\_SIZE, 0x300, (this size is not recommended for customer to change smaller), all interrupt callback function will also use this stack. So customer should avoid defining very large local variables in the interrupt callback function.

Checking irq stack is similar to the second method of checking normal stack, test all the functions once, (the function with trigger reboot cannot be included), then check whether there is still a continuous 0x00 area in this buf of irq\_stk in BDT (the initial value of irq stack is 0x00), and if yes, it means that there is no overflow of RAM. It shows in the following figure:

Ŧ	Download			many "00"00 00 00" so not over 100
Variable Name	Addr	Len	Value	^ 0429d0; 00 00 00 00 00 00 00 00 00 00 00 00 0
group_address	431c0	16		
hwRevision_charUUIE	42280	2	00002a27	
hwRevision_value	422bc	4	22222222	042a10: 00 00 00 00 00 00 00 00 00 00 00 00 0
iBeaconCnt.6288	431d8	1	00000000	042a20: 00 00 00 00 00 00 00 00 00 00 00 00 0
iBeaconInterval	42e95	1	00000000	$\begin{array}{cccccccccccccccccccccccccccccccccccc$
interval_th	42654	1	00000010	042a50; 00 00 00 00 00 00 00 00 00 00 00 00 00
irq_mask_save	43fb8	ouple	00102002	042a60: 00 00 00 00 00 00 00 00 00 00 00 00 0
irq_stk	428d0	768		$- 42a_70: 00 00 00 00 00 00 00 00 00 00 00 00 0$
rq_timer1_cb_time	42c20	4	00000000	042a80: 00 00 00 00 00 00 00 00 00 00 00 00 00
keep_new_cmd_alarm	43004	1	00000000	042aa0: c8 2a 84 00 b8 b1 00 00 10 00 00 a8 b2 00
last_alarm_time.6289	42ffc	4	b3250e98	042ab0: 04 2b 84 00 2d 2b 84 00 02 25 21 00 ff ff f3
last_alarm_time_bridg	431b8	4	00000000	
ast_conn_ibeacon_tin	4315c	4	00000000	042ae0: 00 00 00 00 00 00 00 00 40 07 80 00 a8 53 00
last_ibeacon_time.628	430a4	4	00000000	042af0: 28 2b 00 00 01 00 00 01 00 00 01 00 00 01 00 00
last_music_tick	42bd8	4	00000000	- 042b00: 00 00 00 00 00 00 00 00 00 00 00 00
ed_count	42c2c	4	00000000	042b10: 00 00 00 00 00 21 04 00 23 25 04 00 05 21 04 042b20: 00 7c 00 00 5a 97 00 00 27 00 00 00 82 25 21
led_dbg	43348	4	00000000	042b30: ff ff f3 07 00 6f ab 8e 69 da 32 97 ea a4 00
ed_event_pending	43344	4	00000000	042b40: cf 5b df d6 40 2a 93 d2 00 00 00 00 00 00 00 00 00 00 00 00 00
led_is_on.5084	42be8	4	00000000	042b60: b0 76 00 00 92 00 00 00 00 9b 27 84 00 00 00 00 00
led_lum	4220c	1	00000064	042b70: 20 2c 84 00 26 23 00 00 02 00 09 00 02 00 00
led_lum_tmp	42c0c	1	00000000	042b80: 68 16 00 00 00 00 00 00 ff 7f 00 00 36 05 00
ed_no.5083	42c10	4	00000000	$042b_{2}$ 00 00 00 00 00 00 00 00 00 00 00 00 00
led_onoff.5116	4252d	1	00000001	042bb0: 00 6a 18 00 00 00 00 00 14 2c 00 00 20 06 80
led_onoff.5124	4255c	1	00000001	042bc0: 00 00 00 08 00 00 00 00 00 00 00 ee 28 00
led sel.5081	42c28	4	00000000	Iotal lime: 58 ms

Figure 2.11: IRQ\_stack\_check

# 2.2.4 Size Calculation of Retention RAM

For SIG mesh SDK, if the retention function is turned on, i.e. the macro PM\_DEEPSLEEP\_RETENTION\_ENABLE is enabled, when compiling, it will be checked by boot.link document: if the retention use is more than 32KB, then an error will be reported. It is shown in the following figure:

./drivers/8258/lpc.o ./drivers/8258/qdec.o ./drivers/8258/random.o ./drivers/8258/s7816.o ./drivers/8258/spi.o ./drivers/8258/timer.o ./drivers/8258/uart.o ./drivers/8258/watchdog.o ./drivers/8258/flash/flash mid01460c8.o ./drivers/8258/flash/flash mid130c8.o ./drivers/8258/flash/flash mid1325e.o ./drivers/8258/flash/flash mid160c8.o ./drivers/8258/flash/flash mid1325e.o ./drivers/8258/flash/flash mid134051.o ./drivers/8258/flash/flash mid160c8.o ./drivers/8258/flash/flash mid1360c8.o ./drivers/8258/flash/flash mid134051.o ./drivers/8258/flash/flash mid1360c8.o ./drivers/8258/flash/flash mid1360eb.o ./drivers/8258/flash/flash mid1360c8.o ./drivers/8258/flash/flash mid1360eb.o ./drivers/8258/flash/flash mid14325e.o ./drivers/8258/flash/flash mid1360c8.o ./boot/8258/cstartup 8258\_RET\_16K.o \_\_llt\_8258\_mesh \_lsoft-fp \_lsig\_mesh\_825x C:\TelinkSDK\opt\tc32\bin\tc32=lf-ld.exe: \*\*Retention Area Overflow, Check your data/bss usages\*\* C:\TelinkSDK\opt\tc32\bin\tc32=lf-ld.exe: \*\*Retention Area Overflow, Check your data/bss usages\*\* make: \*\*\* [firmware.elf] Error 1

Figure 2.12: retention\_ram\_size\_overflow

Calculate the size for Retention RAM according to the list documentation:

Sect	ions:					
⊟Idx	Name	Size	VMA	LMA	File off	Algn
0	.vectors	00000210	00000000	00000000	0008000	2**4
		CONTENTS,	ALLOC, LOP	AD, READONI	LY, CODE	
1	.ram_code	00002bb4	00000210	00000210	00008210	2**2
		CONTENTS,	ALLOC, LOA	AD, READONI	LY, CODE	
2	.text	0001a0dc	00002dd0	00002dd0	0000add0	2**4
		CONTENTS,	ALLOC, LOA	AD, READONI	LY, CODE	
3	.rodata	00003050	0001ceac	0001ceac	00024eac	2**2
		CONTENTS,	ALLOC, LOA	AD, READONI	LY, DATA	
4	.data	00000240	00843700	0001fefc	0002b700	2**2
		CONTENTS,	ALLOC, LOP	AD, DATA		
5	.bss	0000243a	00843940	0002013c	0002b940	2**4
		ALLOC				
6	.retention_da	<b>ta</b> 00000004	4 00845d70	0002013	c 0002dd7	c 2**0
		CONTENTS,	ALLOC, LOP	AD, DATA		
7	.retention_bs	s 00000114	00845d80	00020140	0002dd80	2**2
		ALLOC				
8	.TC32.attribu	tes 0000003	10 000000	000000 00	0002dd	80 2**(
		CONTENTS,	READONLY			
9	.comment	0000001a	00000000	00000000	0002dd90	2**0
L		CONTENTS,	READONLY			
				$\sim$		
		Figure	2.13: retentio	n_ram_list		

The retention area contains vectors, ramcode, data, bss segments.

Note that in the mesh SDK (mesh SDK only), data and bss are placed in the retention area by default, i.e., they can be defined without the attribute\_data\_retention or attribute\_bss\_retention prefixes, and for some other reasons, the retention data and retention bss shown above are actually non-retention, only using the name.

So the end address of the retention area is calculated as: .bss VMA (i.e. bss start address) + bss size, for example, the above figure is: (0x843940 + 0x243a) = 0x845d7a; Retention size is: 0x845d7a - 0x84000 = 0x5d7a = 23.4k;

# 2.3 Startup File cstartup.s and Link File boot.link

Unlike the BLE base SDK, the mesh SDK uses only one copy of cstartup.S and boot.link. Take B85m as an example, B85m only uses cstartup\_8258\_RET\_16K.S, which is applicable to the configuration of 16K retention and 32K retention. Because it has already done the automatic identification of the size, the space after the end address of the retention data is used as an ordinary no retention RAM for storing the no retention data/bss, etc.

# 2.4 Clock

Telink

The MCU clock is defined by CLOCK\_SYS\_CLOCK\_HZ. The default value is 32MHz for B85m mesh project and 48MHz for B85m gateway project; the default is 48MHz for both B91m mesh and gateway projects.

# 2.4.1 System clock & System Timer

The system clock is the clock of MCU programs.

The system timer is a read-only timer, providing timing for BLE time sequence, as well as for users.

For Telink 826x IC, the time source of system timer is system clock; for Telink 8x5x IC, system timer is separated with system clock. As shown in figure below, system timer is 16M, generated by 2/3 divider from the external 24M Crystal Oscillator.



As can be seen in above figure, the external 24M Crystal Oscillator will be double to 48M, then divided by the dividers and generate 16M/24M/32M/48M as system clock, such clocks are called crystal clock (e.g., 16M crystal system clock, 24M crystal system clock); internal 24M RC Oscillator can also generate 24M RC clock, 32M RC clock and 48M RC clock, which we call RC clock(BLE SDK does not support RC clock).

For BLE SDK, we recommend crystal clock.

Call the following API configuration system clock when initialization, choose corresponding clock in the definition of enum variable SYS\_CLK\_TYPEDEF.

void clock\_init(SYS\_CLK\_TYPEDEF SYS\_CLK)

8x5x System Timer is different from system clock, user should know the clock source of each hardware module in MCU, whether it is system clock or system timer. The following case is an example, where the system clock is 24M crystal, system clock is 24M, and system timer is 16M.

In app\_config.h, the definitions of system clock and the corresponding S, mS, uS are shown as below:

#define CLOCK\_SYS\_CLOCK\_HZ 24000000
enum{

```
CLOCK_SYS_CLOCK_1S = CLOCK_SYS_CLOCK_HZ,
CLOCK_SYS_CLOCK_1MS = (CLOCK_SYS_CLOCK_1S / 1000),
CLOCK_SYS_CLOCK_1US = (CLOCK_SYS_CLOCK_1S / 1000000),
};
```

All hardware modules whose clock source is system clock must use only CLOCK\_SYS\_CLOCK\_HZ, CLOCK\_SYS\_CLOCK\_1S when set module clock; in other words, if the module's clock set is the clock defined above, then the clock source of the module is system clock.

For example, in PWM driver, PWM cycle and interval are set as following, means the clock source of PWM is system clock.

System Timer is the fixed 16M, so for this timer, the s, ms, us are defined as following in SDK code:

```
//system timer clock source is constant 16M, never change
enum{
    CLOCK_16M_SYS_TIMER_CLK_1S = 16000000,
    CLOCK_16M_SYS_TIMER_CLK_1MS = 16000,
    CLOCK_16M_SYS_TIMER_CLK_1US = 16,
};
```

The following system timer related API should use similar CLOCK\_16M\_SYS\_TIMER\_CLK\_xxx to define time, as shown below.

```
void sleep_us (unsigned long microsec);
unsigned int clock_time(void);
int clock_time_exceed(unsigned int ref, unsigned int span_us);
#define ClockTime clock_time
#define WaitUs sleep_us
#define WaitMs(t) sleep_us((t)*1000)
```

System Timer is BLE timing standard, so, all BLE timing related parameters and variables should use CLOCK\_16M\_SYS\_TIMER\_CLK\_xxx to define time.

## 2.4.2 System Timer Usage

System Timer will start working after cpu\_wakeup\_init in Main function finishes initialization, user can read the value of System Timer tick.

System Timer tick is 32bit long, it will increase by 1 for each time cycle, i.e., 1/16 us, the value is range from 0x00000000 to 0xffffffff. The value of tick is 0 when system boot, it takes about (1/16) us \* ( $2^32$ ) = 268s to reach the maximum value, i.e., System Timer tick repeats the cycle every 268s.

System tick will not stop when MCU is running.

Users can read System Timer tick via clock\_time() function.

u32 current\_tick = clock\_time();

The whole BLE timing sequence is designed based on System Timer tick, and System Timer tick is widely used in the program for timing and over timing record, it is highly recommended to use System Timer tick for timing and over timing determination.

For example, a simple software timing, it is based on inquiry, with moderate real-time character and accuracy, it is for application with lower requirement for time error. Method:

1) start timing: set a u32 variable, read and record current System Timer tick.

u32 start\_tick = clock\_time(); // clock\_time() return System Timer tick value

2) Check the difference between current System Timer tick and start\_tick, see if it is surpass the timing value, if it is, then the timer is triggered, the program will take corresponding action, and the timer will be cleared or start a new timing cycle depends on requirement.

Suppose the timing time is 100 ms, to inquire if the time is up in the following way:

if( (u32) ( clock\_time() - start\_tick) > 100 \* CLOCK\_16M\_SYS\_TIMER\_CLK\_1MS)

The limiting case of the system clock tick going from Oxffffffff to O is solved by converting the difference value to a u32 type.

To deal with the u32 issue caused by different system clock, the SDK provides a unified function, users can use the following function to inquire for any system clock source.

if( clock\_time\_exceed(start\_tick, 100 \* 1000)) //the unit for the second parameter is us

Please be noted, one cycle of 16M clock is 268s, so this function is only applicable to timing no more than 268s. Timing longer than 268s need an extra software counter.

For example, after 2s A is triggered (for only once), the program will take B action.

```
u32 a_trig_tick;
int a_trig_flg = 0;
while(1)
{
    if(A){
        a_trig_tick = clock_time();
        a_trig_flg = 1;
    }
    if(a_trig_flg &&clock_time_exceed(a_trig_tick,2 *1000 * 1000)){
        a_trig_flg = 0;
```

т	Tel	link	

B(); } }

Teint Semiconductor

# 3 Mesh Spec Introduction

See Summary of mesh\_1.1\_feature for mesh-related spec downloads.

This section follows the chapter order in MshPRFv1.0.1.pdf, this is only a brief, for detail, please refer to respective chapters in SIG MESH spec.

# 3.1 Layered architecture



Figure 3.1: Layered Architecture

# 3.1.1 Model layer

The Model defines a node supporting function, each model defines its own op code and status. E.g., generic on/off model defines Generic ON/OFF/GET/STATUS.

When provision, provisioner will get all model id that the node supports via get composition data, and get to know what functions the node supports. Only when the node supports a certain model, the corresponding op code defined by the same model will be sent to the node.

There are 2 kinds of models, server model and client model.

Server model: it is a model which can be controlled. It has its own status, can be changed/obtained by other nodes, e.g., on/off server model can receive on/off set/get command, can response to on/off status command, but it cannot send out on/off set/get command, nor handle on/off status command.

Client model: it can control server node, it has no status of its own. E.g., on/off client model, it can send out on/off set/ get command, it can also handle received on/off status command, but it cannot send out on/off status command, nor handle on/off set/get command.

# 3.1.2 Foundation Model layer

Foundation Model is similar to model, it is basic model, contains Configuration Server model, Configuration Client model, Health Server model, Health Client model.

All configured nodes must contain Configuration Server model, all provisioner must contain Configuration Client model. The 2 models contain subscription add/delete op code, and both of the models' access layers are encrypt with device key, so only provisioner node can send out set/get command of configuration model.

## 3.1.3 Access layer

Combine op code with parameter in prescribed format.

## 3.1.4 Transport layer

Decrypt/encrypt with app key or device key (for configuration model). Determine if it need segmentation or reassembly.

To compliant with protocols that does not support long packet like BLE4.2, the maximum payload is set to 31byte.

## 3.1.5 Network layer

For transmit: contains sequence number for packet, encrypt data with network key and iv index. Sequence number will increase by 1 after transmission.

For reception: decrypt data with network key and iv index, then determine if sequence number is valid (if it is bigger than received value), waive if invalid.



## 3.1.6 Bearer layer

Send out encrypted packet to mesh network via LL\_TYPE\_ADV\_NONCONN\_IND(0x02).

# 3.2 Architectural concepts

# 3.2.1 States

Node states, e.g., on/off States, lightness States.

# 3.2.2 Bound states

2 bound states, like on/off and lightness. When on/off switch from 1 to 0, lightness will switch to 0, too; when on/off switch from 0 to 1, lightness will switch from 0 to the value before turn off. Similarly, when lightness switch from 0 to non-0 values, on/off value will switch from 0 to 1.

# 3.2.3 Messages

The encrypted packet sent to mesh network. Also called as mesh packet/mesh command.

# 3.2.4 Node & Elements

Node is a complete node or Bluetooth module, while element is an addressable entity within a node.

Node has only 1 address, while element can have 1 or multiple continuous addresses. The first element address is called primary address, which is the same with Node address.

Multiple element addresses are needed when a node has multiple same type states. E.g., a switch with 3 sockets need to control on/off state by Generic ON/OFF command, the task cannot be completed if there is only 1 address, in this case, and multiple element addresses are needed.

Although CT light has only one on/off states, it needs 2 generic level models, one is for lightness, the other is for temp; thus it needs 2 elements.

Similarly, HSL light needs 3 elements because it needs 3 generic level models for lightness, Hue and Sat, respectively.

When build network, node will report element number in provision flow interaction flow. E.g., if the number is 2, provisioner will assign an address to node, for example, 0x0002, node will then assign 0x0002 to element 1 and 0x0003 to element 2. Provisioner will assign from 0x0004 for the next node when provisioning.

# 3.2.5 Models

Please refer to 3.3.1 Model layer.

## 3.2.6 Publish & subscribe

- Publish: element send out status spontaneously, configure publish address and publish cycle parameter with command Config Model Publication Set. When publish address is configured, each time when status changes, node will execute publish status action spontaneously. Cycle publish parameter will determine whether it need to send it by a certain cycle.
- Subscribe: when a node receives published status message (e.g., generic on/off status) or control message (e.g. generic on/off), it will determine whether to handle this message based on the model's Subscribe list[].Subscribe list[] contains group address or virtual address, unicast address and Oxffff is invalid. Add new elements with Config Model Subscription Add, Config Model Subscription Virtual Address Add.

Determination rule

- 1) When received destination address is not unicast address, check if it can find a matching address in corresponding model's Subscribe list.
- 2) When destination address is unicast address, check if it matches its own element address.
- 3) When destination address is 0xffff, then means the message should be received/handled.

## 3.2.7 Security

Encrypt/decrypt need Network key, IV index, App key or device key.

A message need to be encrypted twice, encrypt the whole access layer(including op code, parameters) with app key or device key, and encrypt network layer with network key + iv index, network layer is the packet sent to mesh network, contains source address, destination address and sequence number.

When encrypt access layer, if the corresponding model of op code is config model, use device key, otherwise use app key.

For segment message, because the access layer was encrypted by the whole payload, so the decryption will be done only when all the segment packets are received.

SDK supports 2 network keys (NET\_KEY\_MAX) and 2 app keys (APP\_KEY\_MAX) by default.

Multiple network key manage multiple network.

Multiple app keys manages products of different security levels. For example, there are lights and lock in the same mesh network, and lock has higher security level, in this case, user can assign an independent app key to the lock, and the app key is only open to certain mobile app(provisioner), and will not share when sharing network, thus guarantee a higher security level.

## 3.2.8 Sequence Number Storage

As described in 3.1.5 Network layer, Sequence Number(SNO) of mesh message increases by 1 each time it sends out command, when reception determines SNO, it should be bigger than received value, otherwise the value is invalid. This requires store SNO to flash every time it sends out commands, which is too often. To avoid this, we defined the following: store SNO only when it increase by MESH\_CMD\_SAVE\_DELTA(default value is 0x80). To guarantee SNO is bigger than



used value, the reading SNO will be added by MESH\_CMD\_SNO\_SAVE\_DELTA when boot up. Check MESH\_CMD\_SNO\_SAVE\_DELTA mesh\_flash\_save\_check() and mesh\_misc\_retrieve for reference.

## 3.2.9 Friendship

Friendship is the relationship built between Friend Node (FN) and Low Power Node(LPN) according to prescribed establish friend ship flow. After friendship is established, when LPN sleep, FN will store any message sent to LPN by other nodes. When LPN wakes up, it will send POLL inquiry command to FN, and FN will answer with stored message. This method can lower power consumption, but it need friend node in the network, and will cause delay in command receiving and answering.

For a more details, please refer to the mesh spec and the subsequent LPN chapters.

## 3.2.10 Features

Mesh features:

- Relay feature: node will reduce message's TLL value by 1 after it receive the message, and then send out relay, when the received TLL value is less than or equal to 1, then no relay. With relay, the mesh network can obtain longer transmission distance. TLL is to control the delay time of the last node that receives the message. The TLL default value is TTL\_DEFAULT(OxOA) in SDK, this macro is verify-able, provisioner can also configure this with Config Default TTL Set, maximum value is 127.
- Proxy feature: proxy is the protocol for mobile app connect to mesh network. In mesh network, app is an independent node, with its own Node address. Most app can not define transmitting packet discretionarily, neither monitoring mesh network all the time (switch to wifi for some time), so mobile app need to connect to a node with BLE GATT, the node will send out the data it receives from the app, and when it receives the answer message from the mesh network, it will sends back to mobile app with GATT according to proxy protocol.
- App sends message to node with ATT\_OP\_WRITE\_CMD(0x52), node reply app with notify, i.e. ATT\_OP\_HANDLE\_VALUE\_NOTI(0x1B).
- Low Power feature: please refer to Friendship introduction in 3.2.9.
- Friend feature: please refer to Friendship introduction in 3.2.9.

# 3.2.11 Mesh Topology



Figure 3.2: Mesh Topology

All nodes without low power support Relay, Friend. All nodes support ADV provisioning or GATT provisioning by default in this SDK.

# link sen 3.3 Mesh networking

#### 3.3.1 Network layer

Address:

Values	Address Type
060000000000000	Unassigned Address
0b0xxxxxxxxxxxxx (excluding 0b00000000000000)	Unicast Address
0b10xxxxxxxxxxxxxxx	Virtual Address
0b11xxxxxxxxxxxxxxx	Group Address

#### Figure 3.3: 16 bit Address Allocation

- Unassigned address: O for Unassigned address
- Unicast address for element address
- Group address: for group control and publish—-subscribe scheme.
- Virtual address: use together with 16BYTE label UUID, Virtual address is the value that calculate UUID with hash algorithm. When group address (total 16384) is not enough, this can be used to expand.



#### **Network PDU:**





Field Name	Bits	Notes
IVI	1	Least significant bit of IV Index
NID	7	Value derived from the NetKey used to identify the Encryption Key and Privacy Key used to secure this PDU
CTL	1	Network Control
TTL	7	Time To Live
SEQ	24	Sequence Number
SRC	16	Source Address
DST	16	Destination Address
TransportPDU	8 to 128	Transport Protocol Data Unit
NetMIC	32 or 64	Message Integrity Check for Network

#### Figure 3.5: Network PDU Field Definitions

- IVI: iv index (i.e., the lowest bit of iv\_idx\_st.tx[3], stored as big endian)
- NID: network key related
- CTL: flag for control message

#### Network transmit count/interval

Network transmit count is how many times it need to repeat to send out a command. The rf packet is exactly the same, including SNO. The purpose of send command repeatedly is to increase reception success rate. For example, for a 2-node mesh network, if the success rate of each transmit is 80%, that means the packet losing rate is 20%, so, theoretically, the packet losing rate is 6th power of 20%, which is 0.0064%, i.e., the success rate is 99.993%. This of course also depends on RF environment.

Our SDK's default re-transmit count is 5, i.e., TRANSMIT\_CNT\_DEF(5), the total transmit time = n+1 = 6.



Network transmit interval is the interval of 2 adjacent re-transmitted packets, the default value in SDK is 30-40ms, defined by TRANSMIT\_INVL\_STEPS\_DEF, calculate in the following way: ((TRANS-MIT\_INVL\_STEPS\_DEF + 1)\*10 + (0-10))ms.

Network transmit count and transmit interval can also be configured by SIG config command CFG\_NW\_TRANSMIT\_SET.

In conclusion, to send a network packet, e.g., generic ON/OFF no ack (this command need no de-packing), SDK need about 40  $^{*}$  6 = 240ms.

#### Reliable retry

Reliable retry is the retry on app layer, used for commands with status answer, e.g., generic ON/OFF. When send out a network packet (including network transmit), the program will check if the status is received or not, if not, then it will retry, and the sequence number in the network packet will thus change. The program will retry twice at the maximum by default.

#### 3.3.2 Access layer

Field Name	Size (octets)	Notes
Opcode	1, 2, or 3	Operation Code
Parameters	0 to 379	Application Parameters

#### Figure 3.6: Access Payload Field

Opcode Format	Notes	
0xxxxxxx (excluding 01111111)	1-octet Opcodes	
0111111	Reserved for Future Use	
10xxxxxx xxxxxxxx	2-octet Opcodes	
11xxxxxx zzzzzzz	3-octet Opcodes	

#### Figure 3.7: Opcode Format

There are 3 kinds of op code, 1byte, 2byte and 3byte. 1byte and 2byte are defined by SIG, 3byte is defined by vendor, in which 2 bytes are vendor ID (CID), a vendor id supports at most 64 vendor opcode in the whole mesh network.

Access layer contains op code and parameter, supports 380 byte at the maximum.

# 3.3.3 Transport layer

To compliant with protocols that does not support long packet like BLE4.2, the adv's maximum payload is set to 31byte. The effective payload of a single packet is 11bytes, others are occupied by communication protocols, so when Access layer is longer than 11byte, it need to be segmented, thus, for vendor op code, if the parameters is longer than 8 byte(8=11-3), the mesh protocol stack will segment the message spontaneously.

# 3.3.4 Mesh beacon

The following figure shows unprovisioned device beacon PDU.

Unprovisioned Device Beacon

Len (1B)	Type = < <mesh beacon="">&gt; (1B)</mesh>	Beacon Type = 0x00 (1B)	Device UUID (16B)	OOB Info (2B)	URI Hash (4B)	ì
-------------	---	-------------------------------	----------------------	------------------	------------------	---

#### Figure 3.8: Unprovisioned device beacon PDU

Node can be identified by Device UUID. For some mobile phone, for example, IOS cannot get mac, nor can get mac in future remote provision, so in SIG mesh, node is identified by Device UUID instead of by mac.

Unprovisioned beacon is transmit via non-connectable ADV packet, used for PB-ADV provision mode.

Please refer to spec 3.9.2 Unprovisioned Device beacon for Oob info and URI Hash.

Before provision, unprovisioned beacon will be transmit via unprov\_beacon\_send(), the transmit interval is defined by beacon\_send.inter = MAX\_BEACON\_SEND\_INTERVAL, the default value is 2s.

After provision, security beacon will be sent out via mesh\_tx\_sec\_nw\_beacon(). User can also enabledisable this transmit command via CFG\_BEACON\_SET. Please refer to SecNwBc operation in 4.4, the transmit interval is defined by SEC\_NW\_BC\_INV\_DEF\_100MS, the default value is 10s.

## 3.3.5 IV update flow

This is IV index update flow. Both network layer and access layer decryption/encryption need IV index. As described before, mesh network requires network PDU's sequence number accumulate all the time, and the length of sequence number is 3 bytes. So when sequence number approach its maximum, IV index needs to be updated, otherwise the sequence number will be reset to 0, and will be invalid in reception end. So IV index is the expand of sequence number.

Nodes will start and execute IV index update flow spontaneously. When a node is noted that its sequence number is bigger than IV\_UPDATE\_START\_SNO (0xC00000), it will start IV update flow. The IV index increases by 1 for each IV update.

Check SPEC V1.0 3.10.5 IV Update procedure for details.

## 3.3.6 Heartbeat

Mesh heartbeat, sent out periodically, can be used for on/off line check (periodical publish can do the same thing), and hops calculation, i.e., to calculate how many times heartbeat message hopped before it get received.

Count received heartbeat, calculate the hops of each heartbeat, get the min hops and max hops, thus know the structure of the whole network and the message transmit ion reliability of each node. One heartbeat subscription configuration can only monitor/calculate 1 node.

hops is calculated in the following way:

hops = InitTTL - RxTTL +1

- InitTTL: TLL in heartbeat publish set
- RxTTL: TLL in received message network PDU

Nodes do not send heartbeat by default, check "Heartbeat demonstration" section for detailed configuration.

## 3.3.7 Health

Health model related message is used for node's warning/error status, e.g., battery warning/error messages. Check spec 4.2.15.1 Current Fault for detail, as shown in table below.

Value	Description	
0x00	No Fault	
0x01	Battery Low Warning	
0x02	Battery Low Error	
0x03	Supply Voltage Too Low Warning	
0x04	Supply Voltage Too Low Error	
0x05	Supply Voltage Too High Warning	
0x06	Supply Voltage Too High Error	
0x07	Power Supply Interrupted Warning	
0x08	Power Supply Interrupted Error	
0x09	No Load Warning	
0x0A	No Load Error	
OxOB	Overload Warning	
0x0C	Overload Error	

#### Table 3.1: Health model related messages
Value	Description
OxOD	Overheat Warning
OxOE	Overheat Error
OxOF	Condensation Warning
0x10	Condensation Error
Ox11	Vibration Warning
0x12	Vibration Error
0x13	Configuration Warning
0x14	Configuration Error
0x15	Element Not Calibrated Warning
0x16	Element Not Calibrated Error
0x17	Memory Warning
0x18	Memory Error
0x19	Self-Test Warning
0x1A	Self-Test Error
Ox1B	Input Too Low Warning
Ox1C	Input Too Low Error
Ox1D	Input Too High Warning
Ox1E	Input Too High Error
Ox1F	Input No Change Warning
0x20	Input No Change Error
0x21	Actuator Blocked Warning
0x22	Actuator Blocked Error
0x23	Housing Opened Warning
0x24	Housing Opened Error
0x25	Tamper Warning
0x26	Tamper Error
0x27	Device Moved Warning
0x28	Device Moved Error
0x29	Device Dropped Warning

Value	Description
0x2A	Device Dropped Error
Ox2B	Overflow Warning
0x2C	Overflow Error
0x2D	Empty Warning
0x2E	Empty Error
0x2F	Internal Bus Warning
0x30	Internal Bus Error
0x31	Mechanism Jammed Warning
0x32	Mechanism Jammed Error
0x33-0x7F	Reserved for Future Use
0x80-0xFF	Vendor Specific Warning / Error

# 4 Debugging Tool Instructions

# 4.1 Download Firmware

Please refer to "Help" -> "User guide" for BDT tool detailed instruction. The following is instruction for frequent used operations.

Before start, download 8258\_mesh.bin into each node (8258 Dongle), then burn provisioner nodes, there are 2 provisioner modes, master dongle mode connected via GATT and gateway mode(ADV mode).

GATT mode: download 8269\_mesh\_master\_dongle.bin in to 8269 Master Dongle (8269 Dongle).

Gateway mode: download 8258\_mesh\_gw.bin to 8258 gateway Dongle.

For example, user can download firmware to 8258 light node wit the following steps:

1) Hardware connection: connect miniUSB port on EVK board with PC USB port with USB cable, the light on EVK board will flash if the connection succeed. 8258 Dongle connect with EVK board USB port via USB port.

evk device: ok will show in the left lower corner of the tool.



Figure 4.1: Hardware connection

2) Download 8258\_mesh.bin to 8258 Dongle flash with Telink BDT tool.

Telink

т



🐼 BDT connect to 1:usb#vid 248a&pid 5320#5&946e9eb&0&2#{28d78fad-5a12-11d1-ae5b-0000f803a8c2}

Device File View Tool Help			
🖾 8258 • '~ EVK • 🛞 Se <u>t</u> ting 🔗 Er	rase <u>I</u> Download + <u>A</u> ctivate	I▶ Run II <u>P</u> ause ▶ <u>S</u> tep	O Q PC 🥵 Single
b0 10 b0	10 2 SWS	602 06	■ Stall
↓ Download		號건 Tdebug	
[17:25:24]: TC32 EVK: Swire ok! [17:25:28]: Activate OK!			
	Figure 4.2: BDT interfac	e	
		~~~	

Step 1 Open BDT, click to choose the right part no, then click sws to check if EVK and 8258 dongle can communicate normally, if yes, Swire ok will show. If not, it may because the SoC is in sleep mode, click **Activate** to awake the SoC, this is especially important for low power equipment, Activate OK will show when succeed. Active contains MCU restart.

	Erase					
Step 2 Click		to erase	8258	Dongle	flash.	

.

Note: click Setting to set start address and size for the erase action.

🗉 Telink

🗟 BDT connect to 1:usb#vid_248a&pid_5320	)#5&946e9eb&0&2#{28d78fad-5a12	2-11d1-ae5b-0000f803a8c2}		- 🗆 X
Device File View Tool Help				
🛄 8258 • 🍾 EVK • 🐵 Se <u>t</u> ting 🕖 Erase	<u>↓</u> Download + Activate	II Pause 🏶 Step 🔍 PC 💉	Single step 🔹 🥂 🤁 Reset 🖷 manual m	iode 🕶 📕 <u>C</u> lear
b0 10 b0 10	2 SWS 602	06 Stall	602 88	► Start
<u>↓</u> Download	8111 1012 012	Tdebug	E Log windows	
Flash Sector (4K) Erase a Flash Sector (4K) Erase a	t address: 6c000 t address: 6d000 t address: 6e000 t address: 6e000 t address: 70000 t address: 71000 t address: 72000 t address: 72000 t address: 74000 t address: 76000 t address: 76000 t address: 78000 t address: 78000 t address: 78000 t address: 70000 t address: 70000 t address: 70000 t address: 70000 t address: 70000 t address: 76000 t address: 76000 t address: 76000 t address: 76000			^

Figure 4.3: Erase Flash

**Step 3** Click "File", then click "open", choose corresponding "8258\_mesh.bin" file, click to open, then the BDT corresponding file:

evk device: ok	File Path: D:\3.0.2\SIG_MESH_Release_V3.0.2_20191111\sdk\8269_mesh_master_dongle\8269_mesh_master_dongle.bin
	Figure 4.4: Bin file
Step 4 Click	▶ Download , burn the chosen 8258_mesh.bin in flash address start from 0.
Note: click	Setting to set start address and size for the download action, default vaule is 0.



😵 BDT connect to 1:usb#vid_248a&pid_5320#5&946e9eb&0&9#{28d78fad-5a12-11d1-ae5b-0000f803a8c2}	- 🗆	$\times$
Device File View Tool Help		
i 🗓 8258 • 'v EVK • 🛞 Setting 🕐 Erase 🛓 Download + Activate 🕨 Run II Pause ♥ Step Q PC 💉 Single step • C Reset 😨 manual	mode - 🖁 🚽 Clear	
b0 10 b0 10 C SWS 602 06 Stall 602 88	Start	
Log windows ELC windows		
Flash Page Program at address 20c00 Flash Sector (4K) Erase at address 21000 Flash Page Program at address 21400 Flash Page Program at address 21400 Flash Page Program at address 21c00 Flash Page Program at address 22000 Flash Page Program at address 22000 Flash Page Program at address 22400 Flash Page Program at address 22800 Flash Page Program at address 22000 Flash Page Program at address 23000 Flash Page Program at address 24000 Flash Page Program at address 24800 File Download to Flash at address 0x0000000: 150148 bytes Total Time: 8687 ms		~
evk device: ok File Path: D:\3.0.2\SIG_MESH_Release_V3.0.2_20191111\sdk\8258_mesh\8258_mesh.bin V	ersion : 5.4.3	

Figure 4.5: bin file burned into flash

#### Firmware Burning Steps

**Step 1** Click "Tool"—"Memory Access", dialogue box pops up.

#### Note:

This action only applicable when burning light nodes and gateway nodes, GATT master dongle does not need this.

Memory Access			_	×
8258 ~	EVK	- FLASH	~ 6	~
addr 76000	✓ data:	20 19 11 22 FF 11		~

Figure 4.6: Input information

Input the 6-byte MAC as shown above, click enter in data field to write. Click Tab in Addr field to read, this is the read-back confirmation.

If the mac field keeps empty, when 8258 dongle reboot, it will detect 0x76000 has no mac, then it will assign a random mac and save it in 0x76000 in flash.

**Step 2** After reboot, 8258 Dongle can work as a light node.



Above is the BDT frequent used operations. Users can also click "Help" -> "User guide" for details of other functions and operations.

### 4.2 BLE Connection and Adding Light in Gateway USB Mode

- 1) Open sig\_mesh\_tool.exe, plug the gateway dongle with burned 8258\_mesh\_gw.bin in PC USB port.
- As shown below, "Found" means 8258 gateway Dongle connected correctly with PC tool, and the communication works. The tool will choose tl\_node\_gateway.ini automatically based on connected hardware.

P Telink sig_mesh Found	X
CHD tl_node_gateway.ini INI BULKOUT ASCII	Log [fastbind 2 retry Clear Save Save ] Hex [ Adv Stop Scan rp_scan OTA 2x test]
LPN get_lightness LPN get_onoff lightness get_Real fv_info_get fv_info_get_all fv_distribution_start_all fv_distribution_start_0002 fv_distribution_start_002 fv_distribution_start_02_03	<pre>&lt;0001+11:35:00:056 [INFO]: (common)System start</pre>
<pre>'istribution_stop fv_distribution_stop fv_distribution_dtop fv_update_get fv_update_get fv_update_start fv_update_abort fv_update_apply obj_transfor_get obj_transfor_start</pre>	<pre>&lt;0008&gt;11:25:20:059 [INFO]:(GATEMAX)the gateway mac adr is : 27 12 2c d8 cd ab</pre>
<pre>obj_transfer_abott obj_block_transfer_statt obj_block_get obj_block_get </pre>	
sched_action_set sched_action_set_off sched_action_set_on sched_action_set_scenel 	
<pre>time_get time_cone_set time_cone_get time_delta_set time_delta_get time_delta_get</pre>	~
blmg_loam_geo           scene_store           e0 ff 00 00 00 00 02 00 02 00 b6 01	ALL   chn_set USB connect Path: search_file Hesh UART UART GATE_RESET Hesh_ota Gate_ota Frov Close

Figure 4.7: SIG\_MESH\_TOOL interface

- 3) Boot 8258mesh node.
- 4) Click "Scan" to open "ScanDev" window, showing corresponding mac address, including rssi and frequency offset.

	$\times$
BULKOUT ASCII 🔽 Log 🗆 fastbind 2 retry Clear Save Save 🔽 Hex Adv Stop Scan rp	scan OTA Rx test
ScanDev	×
	41 31 cf d5
20 19 aa bb cc 20 -54 dBm 78 K ()	2 0a 0c 0a ff
20 19 aa bb cc 16 -54 dBm 94 K ()	a ff 4c 00 10
el el e2 e3 cd ab -70 dBm 40 K ()	
bf d5 51 63 a7 f8 -70 dBm 72 K ()	69 69 39 b7
20 19 aa bb cc 17 -54 dBm 83 K ()	
21 22 33 44 ff ff -70 dBm 79 K ()	ee 2b 94 ec



5) Click corresponding item in "ScanDev" to choose node.

Gateway mode: double click to choose the node, no connection or command transmission.

8258 gateway Dongle mode: double click will build BLE connection, if the red light on 8258 gateway Dongle turns on, means BLE connection is built successfully, "Stop" is to stop current BLE connection, when the white light on 8258 gateway Dongle turns on, means BLE connection is stopped. Currently supports only single node BLE GATT connection.

6) Click "Prov" to open "provision" window.

#### Note:

"Provision" and "bind\_all" is forbidden after initialization, users can not use the 2 button at the same time.

The "network\_key" is generated randomly when first open "provision", it can be modified before click "Set-Pro\_internal".

provision	×
☐ Fast prov mode SetPro_internal network_key b3 12 4d c8 43 bb 8b a6 1f 03 5a 7d 09 38 25 1f	Static         apk_idx       00 00         app_key       60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48         bind_all
key_index     00 00     iv_index     11 22 33 44       iv_update_flag     0     unicast_adr     01 00       Provision	
filter_operation filter_type white_list v filter_data 01 00 ff ff SetFilter Add_mac RM_mac	

Figure 4.9: provision window

7) Click "SetPro\_internal" to set network initial parameters, print "Set internal provision success" in log window to show that the parameters are set successfully.

<0313>18:30:08:760 [INFO]:(gatt_provision)Set internal provision success			~
<			>
ALL chn_set USB connect Path:	search_file		Mesh
UART GATE_RESET Mesh_o	ta Gate_ota	Prov	Close

#### Figure 4.10: Set internal provision success window



Once click "SetPro\_internal", the corresponding parameters like netkey cannot be modified, so "Set-Pro\_internal" will turn to grey. Parameters will be saved in mesh\_database.json and will be read automatically next time open the tool. If network\_key need to be modified, then the whole network should be dismissed, and reset to Factory settings.

Now "Provision" is enabled. unicast\_addr is the primary address to be assigned to provision, user can change it manually, but it highly recommended not to.

provision	×
Fast prov mode         SetPro_internal         network_key         b3 12 4d c8 43 bb 8b a6 1f 03 5a 7d 09 38 25 1f	Static         OD 00           app_key         60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48           bind_all
key_index 00 00 iv_index 11 22 33 44	
iv_update_flag 0 unicast_adr 02 00	
Provision	
filter_operation	
filter_type white_list - filter_data 01 00 ff ff	
SetFilter Add_mac	
RM_mac	

Figure 4.11: Provision enabled

8) Click "Provision" to execute SIG provision flow to add corresponding node into network. The red LED will flash 4 times to show the connection success. Log information is shown below:

#### Gateway mode log:

: 91 8d 02 00 03 ff 89 df 3e 39 31 dc df dl ba 06 24 ff 2c 82 93 d2 <0084>14:06:55:791 [INFO]:(GATEWAY)HCI_GATEWAY_CMD_PROVISION_EVT : 91 89 01 02 00 97 21 al 78 cd ab 6e 8f 6e 9b 87 el 51 38 af 6a 97 21 al 78 cd ab		
		~
<	>	,
ALL chn_set USB connect Path: search_file	Mesh	
UART GATE_RESET Mesh_ota Gate_ota	Prov Close	2

Figure 4.12: Gateway mode log

#### GATT Master dongle log:



A REAL PROPERTY AND A REAL
Log fastbind 2 retry Clear Save Save V Hex Adv Stop Scan rp_scan OTA Rx test
<0004>17:11:24:113 [INFO]:(gatt_provision)SEND:the provision start is
: 00 00 00 00 00 00
:
82 ce 4b 14 49 18 25 13 f7 da a1 4b 84 20 de a4 71 17 a6 0f 78 cc 14 30 34 49 a7 b1 99 81 73 c3
e9 4b 0d 3c 9a 0c 40 1d c4 48 dd 39 0a cc 95 1a bf 95 6b 5d 68 0a 41 92 bf 35 75 62 d0 ea 37 0c
<pre>&lt;0006&gt;17:11:24:231 [INFO]:(gatt_provision)RCV:the pubkey of the device is .</pre>
23 eb 84 06 8d 23 eb e6 34 8f c4 5a ef 3d 64 4c 26 d4 6f 71 d5 ca 28 04 84 8f 29 48 8b 36 31 d7
06 3c 7a al 7c 5e b8 46 35 bd 6b 6l d8 16 4b dl f2 50 fd 72 8e e5 53 6l 1b c3 ab f5 1d f5 cf 8f
<0007>17:11:24:251 [INFO]: (gatt_provision) SEND: the provisioner's comfirm is
: a5 e5 a9 1a 52 b4 bd 34 fe 29 11 2e c5 da 84 a9
: 80 75 6d ab 9d 3f 6b 39 2f f4 27 34 45 e8 b0 a0
<0009>17:11:25:876 [INFO]:(gatt_provision)SEND:the provisioner's random is
: b3 a6 db 3c 87 0c 3e 99 24 5e 0d 1c 06 b7 47 de
<0010>17:11:25:949 [INFO]: (gatt_provision) RCV: the device's random is
: ca 35 46 d5 19 41 19 80 eC /0 C2 91 a6 95 80 0d
<pre>&lt;0012&gt;17:11:25:966 [INFO]: (gatt provision) the device commitme check is success &lt;0012&gt;17:11:25:966 [INFO]: (gatt provision) SEND: the provisioner's device info is</pre>
: b3 12 4d c8 43 bb 8b a6 1f 03 5a 7d 09 38 25 1f 00 00 00 11 22 33 44 06 00
<0013>17:11:25:975 [INFO]:(gatt_provision)the node's dev key:
: c7 fb 7f c1 7e b6 b3 2d 3c cf 28 7e 89 47 20 62
<0014>1/:11:26:109 [INFO]: (gatt_provision) RCV:rcV the provision completet cmd provision success
<0016>17:11:26:149 [INFO]: (Basic) filter send cmd is 1: 00 01
<0017>17:11:26:170 [INFO]:(Basic)filter send cmd is 1: ff ff
<0018>17:11:26:270 [INFO]: (Basic) the filter rsp is 0: 00 00 05 e5
<0019>17:11:26:280 [INFO]: (Basic)mesh_rc_data_cfg_gatt dec suc
<0021>17:11:26:294 [INFO]:(log_win32) white fist <0021>17:11:26:306 [INFO]:(log_win32)GATT addr 0x0006. filter list status. ListSize is: 0
<pre>&lt;0022&gt;17:11:26:318 [INFO]:(Basic) the filter rsp is 0: 00 01 f3 b6</pre>
<0023>17:11:26:331 [INFO]:(Basic)mesh_rc_data_cfg_gatt dec suc
<0024>17:11:26:344 [INFO]:(log_win32) white list
<0025>17:11:26:356 [INFO]:(log_win32)GATT addr 0x0006, filter list status, ListSize is: 1
<0027>17:11:26:382 [INFO]: (Basic) the fifter rsp is 0. 00 02 57 Se
<0028>17:11:26:393 [INFO]:(log win32] white list
<0029>17:11:26:404 [INFO]:(log_win32)GA11 auur 0x0006, filter list status, ListSize is: 2
ALL T she ast servest CATE RECET Bath:
Cnn_set Connect GAIL_KESLI Fath
UART         USB         JS_UPDATE         Mesh_ota         Gate_ota         Prov         Close

Figure 4.13: GATT Master dongle log

9) Click "bind\_all" after configure app\_key, first get composition data will be sent to get all model ids, then bind app\_key to all models.

After Bind\_all, unicast\_adr will automatically accumulate based on the elements number the current node contains, and calculate primary address for next node provision.(e.g., CT light has 2 elements, then unicast\_adr will increase by 2 each time a CT light is added).

💱 Telink master Found	×
CHD sig_me provision	× can OTA Rx test
LBN get_light LBN get_ondf  Fast prov mode lightness get fv_info_get_ v_info_get_ v_distributi v_distributi fv_distributi fv_distributi fv_distributi fv_distributi	Static         00 00 00 00 13           apk_idx         00 00           app_key         60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48           bind_all         30
fv_distributi         fv_distributi           fv_distributi         fv_uotate_get           fv_uotate_get         00 00           fv_uotate_get         11 22 33 44           fv_uotate_get         fu_uotate_get           fv_uotate_get         00 00	30 : 80 3e 00 02 0 1 00 00 00 03 13 cnt 1 14 00 00 04 13
by_chose_boy fv_update_app cbj_transfer obj_transfer obj_transfer obj_transfer obj_tock_tra obj_chosk_tra	30 : 80 38 00 02 0 1 00 00 00 04 13 cmt 1 000000211 00 00 11 02 00 00 1 00 00 1 00 00
obj_block_get     filter_type     white_list     filter_data     01 00 ff ff       scheduler_get     SetFilter     Add_mac       sched_action	1 00 00 00 10 02 0   00   00   00 00 02 10   00 00 02 10   00 00 02 00
cched_action	i 00 00 00 02 10 (KITHIND)SEND: appkey bind addr: 0x0003,sig model 1d: 0x1056 (common)IxecCmd: a3 ff 00 00 00 00 00 02 00 80 3d 03 00 00 00 61 3 [Basic)the mesh access tx cmd is 0x3430 : 03 00 00 00 06 13
Line_cone_get         <0471>10:41:30:534 [INFO]           Line_cone_get         <0472>10:41:30:536 [INFO]           Line_dalta_set         <0472>10:41:30:561 [INFO]           Line_cone_get         <0472>10:41:30:561 [INFO]           Line_cone_set         <0472>10:41:30:561 [INFO]           Line_cone_set         <0473>10:41:30:561 [INFO]	:[Basio]adf_src:0x0003_adf_ds:[0x0001_access rx cmd is 0x3e00 : 80 3e 00 03 0 (cmd_spS)fatus Rap 20 00 10 08 03 e0 03 00 00 00 06 13 :[log win32]mesh tx reliable stop: op 0x3d80 rsp_max 1, rsp_cnt 1 (KEYBIND)SEND: mesh Reybind event success
clamtoamwe   <   <   <   <   <   <   <   <   <	set USB connect Path: search_file Mesh
	UART GATE_RESET Mesh_ota Gate_ota Prov Close

Figure 4.14: Click bind\_all

10) After binding App\_key, click mesh to enter mesh UI, user can turn on/off light here.



Figure 4.15: mesh UI

#### 11) Dismiss network

Telink

T

Both GATT master dongle and Gateway mode can follow the following way:

Choose a node, then click "DelNode" to delete this node. Refer to 4.5.3 "DelNode" instruction for detail.



#### Note:

In GATT master dongle mode, please delete no-GATT-direct-connected nodes first, then delete GATT direct connected nodes. Delete GATT directly connected node will disconnect current GATT.

# 4.3 BLE Connection and Adding Light in Gateway UART Mode

Configure UART ports:

- 1) Choose HCI\_USE\_UART in HCI\_ACCESS in gateway firmware, re-compile.
- 2) Insert port tool to PC, connect tx/rx with gateway rx/tx.
- 3) Open "sig\_mesh\_tool.exe".
- 4) Click UART then choose the pop-up COM port.
- 5) Click "Connect", if the connection succeed, the button will change to Disconnect. Now UART can execute gateway functions.
- 6) All other operations are similar with that of USB mode, please refer to section 4.2 for detail.



Figure 4.16: Configure UART port

# 4.4 BLE Connection and Adding Light in GATT master dongle Mode

1) Open sig\_mesh\_tool.exe", plug 8269 Master Dongle with burned program in PC USB port.



2) As shown below, "Found" means8269 Master Dongle connected correctly with PC tool, and the communication works. The tool will choose sig\_mesh\_master.ini automatically based on connected hardware.

Pelink master Found	×
CHD sig_mesh_master.ini INI BULKOUT ASCII LEN gee_lightness LEN gee_off lightness gee_Panel 	Log         fastbind         2         retry         Clear         Save         W Hex         Adv         Stop         Scan         rp_scan         OTA         Rx test           <0000-18:12:12:462
<pre>Fu_distribution_top fv_distribution_top fv_distribution_top fv_distribution_top fv_distribution_top fv_update_get fv_update_get fv_update_start fv_update_apply obj_transfer_get obj_transfer_start obj_transfer_start</pre>	Log
<pre>obj_chink_transer obj_lock_get obj_lock_get sched_action_get sched_action_get sched_action_set_off sched_acti</pre>	
<pre>time_set time_come_set time_come_set time_come_get time_delts_set time_delts_get time_cole_set time_cole_get</pre>	<
scene_store v a3 ff 00 00 00 00 02 00 02 00 b7 02 11 22 33 44 55 66 77 88	ALL     Image: charge definition     USB     connect     Path:     search_file     Mesh       12 00 C     UART     Image: charge definition     GATE_RESET     Mesh_ota     Gate_ota     Prov     Close

Figure 4.17: SIG\_MESH\_TOOL interface

3) Please refer to section 4.2 for further steps.

# 4.5 Control Corresponding Nodes

GATT master dongle and gateway have the same operation and UI.

### 4.5.1 UI Display and on/off Control of Single/All Node(s)

1) Click "Mesh". A "Mesh" window will pop up.



Figure 4.18: mesh window

2) By clicking "Nodes" in "Mesh" window, user can refresh all light status.

If click "Nodes" when choose reliable in the right drop-down box, it will send out lightness get command. The UI will be refreshed according to lightness status.

If click "Nodes" when choose unreliable in the right drop-down box, it will send out lightness get command. But on/off command is no ack. Node will not reply status in this case, so the UI will not be refreshed, use publish to refresh UI.

If click "Nodes" when choose online status in the right drop-down box, it will not send out command, only initialize UI to null, then refresh UI according to returned online status data.

Note: online status is a private mode, the node's firmware should enable ONLINE\_STATUS\_EN.

The lightness display in 0-100 scale, which is switched from SIG defined 0-65536 scale.

Μ	Mesh						
	Me	sh —					_
	001	0007	On	Off	$\bigcirc$	100	^
	002	0004	On	Off	$\bigcirc$	100	

Figure 4.19: Node status

3) Single node operation: click "On"/"Off", the corresponding light will control the switch status. The node status will be reported to the tool to refresh the corresponding status.





Figure 4.20: Single node control

5) All on all off control: Click "On"/"Off" besides "All", all nodes in the mesh network will switch to on/off.

Mesh		
Mesh	Nodes reliable V	Group
001 0007 On Off 🔾 100 📤	antipat	All On Off Svr Clnt
002 0004 On Off O 100	ffff	0 On Off



### 4.5.2 Group Control (Subscription Demo)

Click index number of the light node, e.g., 002, to get the node address and show in position in below figure. The default value is 0xffff, means no node is chosen.

Mesh							
- Me	sh —					_	Nodes reliable -
001	0007	On	Off	$\bigcirc$	100	^	neticet
002	0004	On	Off	0	100		0004
	75						



User can click/right click "Svr" box in Group Control, to add/delete this light node in/from corresponding group. The  $\checkmark$  in "Svr" means the node has been added to the group, the blank in "Svr" means the node is not in the group.

- "Svr" column is for generic on/off server model (0x1000)
- "Clnt" column is for generic on/off client model (0x1001)

Normally, node supports only server model, so only "Svr" column is operated.

Group index and group address's relation is described as: group address = group index + 0xC000.



Figure 4.23: Allocate one light to multiple groups

User can click the corresponding "On"/"Off" to control the group in Group controls.

Mesh Mesh 001 0007 On Off O 100 002 0004 On Off O 0 Group\_S Figure 4.24: Group control

### 4.5.3 Configure Node Parameter with UI

Telink

As shown in figure below, click position 1 to choose the node, then the tool will send get group (subscription list) command out automatically to get the node's group and show it in corresponding UI. Then determine if current node supports scene, time, scheduler function, if yes, send get commands out automatically to get scene, time and scheduler list.

Send SCENE\_REG\_GET to get all valid scene index, and display in UI list.

Send TIME\_GET to get time of current node, and display in UI. If the node is just booted, the time will be 0, which means the node is waiting for configuration, in this case, the time will keep 0, and do not do the timing action. User can configure time in 2 ways, 1, send time set command via app/gateway, 2, when the node boots, configure its time model's publish character to get time status other nodes published.

Send SCHD\_GET to get all valid scheduler index, then based on the returned index value, send SCHD\_ACTION\_GET respectively to get detail parameters, and display in UI list. As shown in below figure, the provision is just completed, no scheduler is added, so no need to send SCHD\_ACTION\_GET.

For the same reason, group, scene, time and scheduler are all blank.

23

😵 Telink master -- Found

_									
	CMD sig_mesh_master.ini  INI BULKOU	ASCII 🔽 Log 🗌 fastbind	2 retry Clear Save V Hex Adv Stop Scan rp_scan OTA Rx test						
	mesh_bulk_cmd_debug LPN_get_lightness	<pre>&lt;0000&gt;16:10:14: &lt;0001&gt;16:10:14:</pre>	107 [INFO]: (common Auto get subscription list to refresh UI 123 [INFO]: (common Execute: so ff ou 00 00 00 02 01 06 00 80 29 06 00 00 10						
1	LPN_get_onoff	<0002>16:10:14:	125 [INFO]: (Basic) the mesh access tx cmd is 0x2980 : 06 00 00 10						
	lightness_get_Panel	<0003>16:10:14:231 [INFO]: (Basic) adr_src:0x0006, adr_dst:0x0001, access rx cmd is 0x2a80 : 80 2a 00 06 0							
		< < <p>&lt; &lt; <p>&lt; &lt; &lt; <p>&lt; &lt; &lt; &lt; &lt; &lt; &lt;</p></p></p>							
	fw_info_get	<pre>&lt;0005&gt;16:10:14:261 [INFO]:(log_win32)mesh_bm_reliable_stop: op 0x2980 rsp_max 1, rsp_cnt 1</pre>							
	fw_info_get_all	<0006>16:10:14:	<0006>16:10:14:264 [INFO]:(common@auto get scene)to refresh UI						
	fw_distribution_get	<0007>16:10:14:	<0007>16:10:14:267 [INFO]: (common) Execond:						
	fw_distribution_start_all	<0008>16:10:14:	<0008>16:10:14:271 [INFO]: (Basic) the mesh access tx cmd is 0x4482 NULL						
	fw_distribution_start_0002	<0009>16:10:14:	392 [INFO]: (Basic) adr_src:0x0006, adr_dst:0x0001, access rx cmd is 0x4582 : 82 45 00 00 0						
	fw_distribution_start_02_03	<0010>16:10:14:	398 [INFO]:(cmd_rsp)Status Rsp 7 : 06 00 01 00 82 45 00 00 00						
	fw_distribution_start_0001	<0011>16:10:14:	412 [INFO]: (log_win32)mesh tx reliable_stop: op 0x4482 rsp_max 1, rsp_cnt 1						
	Iw_distribution_stop	<0012>16:10:14:420 [INFO]: (common auto get time to refresh UI							
	Tw_distribution_detail_get	<0013>16:10:14:	424 [INFO]: (common) Exected: as II 00 00 00 00 02 01 06 00 82 37						
	tw_update_get	<0014>16:10:14:	434 [INFO]: (Basic) the mesh access tx cmd is 0x3782 NULL						
	rw_update_prepare	<0015>16:10:14:	555 [INFO]:(Basic)adr_src:0x0006,adr_dst:0x0001,access rx cmd is 0x5d : 5d 00 00 00						
	IW_update_start	<0016>16:10:14:	561 [INFO]: (cmd_rsp)Status Rsp: 06 00 01 00 5d 00 00 00 00 00						
	rw_update_abort	<0017>16:10:14:	570 [ERR]:(common)time value is invalid, please set time						
	IW_update_appiy	<0018>16:10:14:	583 [INFO]: (log_win32)mesh_tx_reliable_stop: op 0x3782 rsp_max 1, rsp_cnt 1						
	obj_transfer_get	<0019>16:10:14:	593 [INFO]:(common auto get scheduler)to refresh UI						
	obj_transfer_start	<0020>16:10:14:	603 [INFO]: (common)Execting. as if 00 00 00 00 02 01 06 00 82 49						
м	lesh	Construction of the	4a 00 00						
	Mesh Nodes reliable v	Group	schedule						
		All On Off Sur Clast	Year Day Time						
	control of the settlest	All On On Svi Cint	any a family any a second seco						
			Custom Cu						
			Month						
	Group_S	1 On Off	set time						
	Craws C		□ Jan □ Feb □ Mar □ Apr □ Mav □ Jun □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □						
	Group_C	2 On Off	□ Jul □ Aug □ Sep □ Oct □ Nov □ Dec □						
	GroDelAll S	-	Scene						
	GripDelAll_C	3 On Of	Hour ⓒ any hour ⊂ once a day ⊂ cus 0 Store 1						
	GetPub_S	5 On Of	Minute Second id scene number						
	SecNwBc	6 On Off	C every 15 minute C every 15 minute C every 15 seco						
	Π	7 On Off	C once an hour C once a minute						
	transmit	8 On Off V	E File Mesh						
	Relay	9 On Off 4							
	Friend	10 On Off							
	Lightness	11 On Off	© On © Off © No action © Recall 0						
	СЛ	12 On Off	Action Set						
	RFU	14 On Off	id action						
	GetCPS								

Figure 4.25: Configure node parameter with UI

The following button is based on current chosen node.

#### "Group\_S"

By clicking this button, CFG\_SIG\_MODEL\_SUB\_GET will be sent to get subscription address list of on/off server model of current node, and display it in "Svr" column.

#### "Group\_C"

By clicking this button, CFG\_SIG\_MODEL\_SUB\_GET will be sent to get subscription address list of on/off client model of current node, and display it in "Clnt" column.

Most nodes do not support on/off client model, so this is not a frequent used button.

#### "GrpDelAll\_S"

By clicking this button, CFG\_MODEL\_SUB\_DEL\_ALL will be sent to delete subscription address list of on/off server model of current node, and clear "Svr" column.

#### "GrpDelAll\_C"

By clicking this button, CFG\_MODEL\_SUB\_DEL\_ALL will be sent to delete subscription address list of on/off client model of current node, and clear "Clnt" column.

#### "GetPub\_S"



By clicking this button, CFG\_MODEL\_PUB\_GET will be sent to get publish address of on/off server model of current node, and display the return value in later display box.

Modify publish address in input box, then press "Enter", CFG\_MODEL\_PUB\_SET will be sent, and the corresponding publish address is the value in input box.

GetPub_S 0
------------

Figure 4.26: "GetPub\_S"

Other parameters are default values. Check command sending log:

```
ExecCmd: a3 ff 00 00 00 00 02 01 06 00 03 06 00 00 00 00 00 ff 00 15 00 10
```

#### Figure 4.27: Command sending log

Other publish parameters can be modified by cfg\_pub\_set\_sig of INI command, modify the parameter to wanted value, then send.

#### **`SecNwBc**″

By clicking this button, CFG\_BEACON\_GET will be sent, the return value will display in right display box. This command determine whether to send security network beacon or not.

Modify value in input box, then press "Enter", CFG\_BEACON\_SET will be sent, and the corresponding parameter value is the value in the input box.



#### "TTL"

By clicking this button, CFG\_DEFAULT\_TTL\_GET will be sent, the return value will display in right display box. This command gets the default TTL value of the node. SDK default value is defined by TTL\_DEFAULT.

Modify TLL value in input box, then press "Enter", CFG\_BEACON\_SET will be sent, and the corresponding parameter value is the value in the input box.





#### "transmit"

By clicking this button, CFG\_NW\_TRANSMIT\_GET will be sent, the return value will display in right display box. This command gets the network transmit value of the node. The lower 3bit is network transmit count, the higher 5bit is network transmit interval.

SDK default values are defines as following:

### : #define TRANSMIT\_CNT\_DEF (5) : #define TRANSMIT\_INVL\_STEPS\_DEF (2)

#### Figure 4.30: SDK default value

#### Note:

transmit count(5) + network transmit interval(2) is 0x15.

Modify network transmit value in input box, then press "Enter", CFG\_NW\_TRANSMIT\_SET will be sent, and the corresponding parameter value is the value in the input box.

transmit	15
----------	----

Figure 4.31: "transmit"

#### "Relay"

By clicking this button, CFG\_RELAY\_GET will be sent, the return value will display in right display box. This command gets the relay enable value of the node.

Modify Relay value in input box, then press "Enter", CFG\_RELAY\_SET will be sent, and the corresponding parameter value is the value in the input box.



#### "Friend"

By clicking this button, CFG\_FRIEND\_GET will be sent, the return value will display in right display box. This command gets the friend feature enable value of the node.

Modify Friend value in input box, then press "Enter", CFG\_FRIEND\_SET will be sent, and the corresponding parameter value is the value in the input box.

Fri	end	1	

#### Figure 4.33: "Friend"

#### "Proxy"

By clicking this button, CFG\_GATT\_PROXY\_GET will be sent, the return value will display in right display box. This command gets the proxy feature enable value of the node.

Modify Proxy value in input box, then press "Enter", CFG\_GATT\_PROXY\_SET will be sent, and the corresponding parameter value is the value in the input box.

Proxy 1

#### Figure 4.34: "Proxy"

#### "Lightness"

By clicking this button, LIGHTNESS\_GET will be sent, the return value will first switch from 0-65535 scale to 0-0x64 and then display in right display box.

Modify Lightness value in input box, then press "Enter", LIGHTNESS\_SET will be sent, and the corresponding parameter value is the value in the input box.

Lightness	64
-----------	----

Figure 4.35: "Lightness"

#### "С∕Т″

By clicking this button, LIGHT\_CTL\_TEMP\_GET will be sent, the return value will first switch from 800-20000 scale to 0-0x64 and then display in right display box.

Modify C/T value in input box, then press "Enter", LIGHT\_CTL\_TEMP\_SET will be sent, and the corresponding parameter value is the value in the input box.



#### "RFU":

Reserve for future.

#### "GetCPS"

By clicking this button, COMPOSITION\_DATA\_GET will be sent, the return value will display in right display box.

1	log	ſ	f	ast	bin	d	2		ret	ry	lea	r S	Save	·		Save	• I	₩ н	ex∫	A	dv	St	op		Scar	1	rp	_sc	an		OTA	Rx	tes	st
ſ	<0	00	)>1	8:2	2:4	4:3	79	[IN]	FO] :	(cor	nmor	1) E:	xec	Cmd :	: a:	3 ff	: 00	00	00	00	02	01	06	00	80	08 (	00							
1	<0	00:	1>1	8:2	2:44	4:3	83	[IN]	FO]:	(Bas	sic)	the	e me	esh	ac	cess	tx t	cm	d i	в Ор	x088	30 :	: 00	0										
1	<0	00	2>1	8:2	2:44	4:5	32	[IN]	FO]:	(Bas	sic)	rx	seq	gmer	nt a	all	pac	kete	es :	rec	eive	ed												
1	<0	00:	3>1	8:2	2:44	4:5	39	[IN]	FO]:	(Bas	sic)	ad:	r_s:	re:(	0x0	006,	adr	_dst	t:0:	ĸ00	01,4	acce	299	rx	cmd	is	0x2	2 :						
1		02	00	11	02	01	00	33	30	69	00	07	00	00	00	17	01	00	00	02	00	03	00	04	00	00	fe	01	fe	00	ff	01	ff	
1		00	12	01	12	00	10	02	10	04	10	06	10	07	10	03	12	04	12	06	12	07	12	00	13	01	13	03	13	04	13	11	02	
1		00	00	00	00	02	00	02	10	06	13																							
1	<0	00	4>1	8:2	2:44	4:5	42	[IN]	FO]:	(cm	d_rs	p)	Stat	tus	Rsj					_:														
1		06	00	01	00	02	00	11	02	01	00	33	30	69	00	07	00	00	00	17	01	00	00	02	00	03	00	04	00	00	fe	01	fe	
1		00	ff	01	ff	00	12	01	12	00	10	02	10	04	10	06	10	07	10	03	12	04	12	06	12	07	12	00	13	01	13	03	13	
1		04	13	11	02	00	00	00	00	02	00	02	10	06	13																			
1	<0	00	5>1	8:2	2:44	4:5	64	[IN]	FO]:	(10	g_wi	.n3	2) me	esh_	tx	rel	iab	le_s	stoj	p: (	op (	0x08	380	rs	_max	к 1,	, rs	sp_o	ent	1				
1																																		
1																																		

Figure 4.37: Return value



#### "DelNode":

By clicking this button, NODE\_RESET will be sent to delete current node from the network. If the deletion succeed, the red LED light of the node will flash for 8 times, and the node will run reboot operation.

### 4.6 Time model operation

- The node's time model is disabled by default in Firmware, MD\_TIME\_EN needs to be set to 1. Gateway 8269 is disabled by default, need to be enabled, gateway 8258 is enabled by default. After setting MD\_TIME\_EN to 1 for the node and gateway, compile and burn, and regroup the network.
- 2) Double click to choose the node.

Mesh					Nodes reliable
001 0007	On	Off 🔾	100	^	notical citable
002 0004	On	Off 🔾	100	-	0004

Figure 4.38: Double click to choose the node

3) Click "set time", the tool will send current time of the PC to the node via "TIME\_SET". Time will display as following, and will refresh automatically.

Time	
get time	2019-11-25 01:01:39
set time	

Figure 4.39: "set time"

Note: the parameters of time set when sending is shown in detail below:

```
typedef struct{
    u32 TAI_sec; // 32bit is enough for 2000 ~ 2099 year
    u8 TAI_sec_rsv;
    u8 sub_sec;
    u8 uncertainty;
    u16 time_auth :1;
    u16 TAI_UTC_delta :15;
    u8 zone_offset;
}time_status_t;
```

Figure 4.40: time set parameters

- TAI\_sec: the value compares with time zone 0.
- zone\_offset: set current time zone, unit is 15 minutes.

```
e.g.: Beijing time2019/1/1 09:00:00(UTC+8) configuration:
```

```
void tx_cmd_time_set_local_sample()
{
    // beijing: 2019/1/1 09:00:00 (time zone: east 8)
    s8 zone hour = 8; // Positive numbers are eastwards
    mesh UTC t UTC = \{0\};
    UTC.year = 2019;
    UTC.month = 12;
    UTC.day = 4;
    UTC.hour = 10 - zone hour; // translate to 0 time zone.
    UTC.minute = 0;
    UTC.second = 0;
    u32 TAI_sec = get_TAI_sec(&UTC);
    time status t time set = {0};
    time set.TAI sec = TAI sec;
    time_set.zone_offset = get_time_zone_offset(zone hour*60);
    access cmd time set(0xffff, 1, &time set);
}
```

#### Figure 4.41: Switch between TAI and local time

The above function is to show how to switch local time to TAI, but normally it not in this way. Time set is set from Mobile APP or PC, and both have API to get current TAI\_sec and zone\_offset, e.g., PC firmware operate in the following way:

(OFFSET\_1970\_2000 is because PC's base time is 1970 while SIG MESH's base time is 2000)

```
void CTLMeshDlg::OnBnClickedSetTime()
ł
    if(0 != Sel Ele Check()) {
       return ;
    time status t settime={0};
   //mesh_UTC_t set_utc;
    CTime time = CTime::GetCurrentTime();
   u32 nTSeconds2 = (u32)time.GetTime();
    settime.TAI sec=nTSeconds2-OFFSET 1970 2000;
   TIME ZONE INFORMATION tz;
   u32 dwRet = GetTimeZoneInformation(&tz);
    settime.zone_offset = get_time_zone_offset(-tz.Bias);
    if(is support model dst(mesh sel,SIG MD TIME S,1)) {
        access cmd time set(mesh sel, 0, &settime);
    }else{
       LOG MSG INFO (TL LOG COMMON, 0, 0, "Node not support time model", 0);
}
```



#### Note:

When the node is powered off, the time will be lost, i.e. g\_TAl\_sec is equal to 0, so it will not start timing. You need to receive the timeset command from app or gateway, etc., or receive TIME\_STATUS information from other nodes that have not been powered off to publish. after that the clock will work properly. How to configure node publish TIME\_STATUS message: When Telink SIG mesh app is networking, it will check if there is a time model inside the model list of the composition data, and if yes, it will automatically send the publish command to the time model. If it is a gateway, or a master dongle, etc., you need to send the command manually.

# 4.7 Scene model operation

- 1) The node's scene model is disabled by default in Firmware, MD\_SCENE\_EN need to be set to 1. Gateway 8269 is disabled by default, need to be enabled, gateway 8258 is enabled by default.
- 2) Double click to choose the node.
- 3) Set the node's status to the scene wanted one via UI or INI. E.g., generic on/off set and lightness set.
- 4) Input scene number, then click "Store", and scene adding command (SCENE\_STORE) will be sent, to set node's current status to corresponding scene ID, and list all the configured scene ID in the list, as shown in figure below.

The processing function after the node receives the SCENE\_STORE is: mesh\_cmd\_sig\_scene\_set().

#### Note:

The SCENE\_STORE have only scene number, no light status information. Node will automatically save currently status information like on/off, lightness as scene status when receive scene adding command.





Ξ

Figure 4.43: Input scene number

5) Recall scene, i.e., set light status to status defined by scene. Click buttons in the following figure, then click "Recall".

The processing function after the node receives a SCENE\_RECALL is: mesh\_cmd\_sig\_scene\_recall().

#### Note:

Telink

T

Recall scene will change light status, but it is not reported because publish status is not configured, to refresh UI, configure publish parameter in corresponding model.



### Figure 4.44: Recall scene

- 6) Modify scene. No modify command, modify with scene store.
- 7) Delete scene. Click buttons in the following picture, then press "Delete".



Figure 4.45: Delete scene

# 4.8 Scheduler model operation

- 1) Parameters (refer spec for details)
- Year: any: each year, custom: a specific year, base: year 2000, ie, 0 means year 2000, 19 means year 2019
- Month: can choose 1/multiple/all. Note: both blank and all means choose all
- Day: any: each day, custom: a specific day
- Week: can choose 1/multiple/all. Note: both blank and all means choose all
- Hour: any: each hour, once a day: randomly respond once a day, and the random number is generated daily; custom: a specific year,
- Min: any: each minute, every 15 means responds on 0/15/30/45, every 20 means responds on 0/20/40, once an hour: randomly respond once an hour, and the random number is generated hourly, custom: a specific minute
- Second: similar with Min.
- 2) Double click to choose the node. If the action column is blank, that means the ID's schedule of the node is not configured yet.



Figure 4.46: Action Set

3) Click ID column, choose the to-be configured scheduler's ID (maximum 16 by definition in SIG, range from 0-15), the chosen ID will show in blue background, and the schedule parameter of the ID will be refreshed to the UI above.

celine sch

schedule	
Year Day	
Cany 0 1	-
MUIIUI	
🔽 Jan 🗹 Feb 🗹 Mar 🗹 Apr 🗹 Ma 🗸	Jun 📗
🗹 Jul 🗹 Aug 🗹 Sep 🗹 Oct 🗹 Nov 🗹	Dec
○ any hour ○ once a day ⓒ cus 8	_
Minute	
C any minute C any second	
every 15 minute every 15 sec	:0
C every 20 minute C every 20 sec	:0
Conce an nour Conce a minut	e
Week	
🗹 Mon 🗹 Tue 🗹 Wed _	
🗹 Thu 🗹 Fri 🗹 Sat 🗹 Sun	
Action	
• On <u>Off ONo action ORecall</u>	
Action Set	
id action	<u> </u>
	=
2	
3	
4	
5	Ŧ

Figure 4.47: Click "id"

4) User can modify schedule parameter, then click "Action Set" to send SCHD\_ACTION\_SET to configure.

Because of the UI display limit, only action parameters are shown in the list, i.e., "on", "off", "no action", "recall", if the field is blank, that means the schedule is not configured yet.

Click a specific value in ID column to check detail information of a schedule id.

_ schedule	
Year	Day
• any	• any
C custom	O custor 1
Month	
🗹 Jan 🗹 Feb 🗹 Mar	🗸 Apr 🗹 May 🗸 Jun
🗹 Jul 🗹 Aug 🗹 Sep	🗹 Oct 🗹 Nov🗹 Dec
C any hour C once	a day 💿 cus 8
Minute	Second
C any minute	O any second
C every 15 minute	• every 15 seco
• every 20 minute	C every 20 seco
• once an hour	O once a minute
• custom 0	• custom 0
Week	
🛛 🗹 Mon 🗹 Tue 🗹 We	ed
🛛 🗹 Thu 🗹 Fri 🗹 Sa	t 🗹 Sun
Action	
	ction C. Decall 1
Action Set	
id action	<u>^</u>
0 (on	=
	-
1	
2	
3	
·	
5	<b>T</b>

Figure 4.48: schedule parameter

#### 5) Delete Schedule

No specific delete command in SIG. Set action of the schedule to "No action" to delete the schedule.

# 5 Factory Test Mode

### 5.1 Purpose

Factory test mode is used to manufacture, to execute some common control tests without provision, e.g., on/off, lightness control, CT control and etc. Gateway and GATT master dongle support this mode while APP does not support for now.

### 5.2 Factory Test Mode Parameters

- unicast address: it is the lower 15bit of MAC by default, if the lower 15bit is 0, then take 1 as unicast address.
- The network key, app key, device key, IV index use the compiled default value.

### 5.3 Default Test-able Commands

The control-able models are defined by factory\_test\_model\_array[], while the useable commands of configure model are defined by factory\_test\_cfg\_op\_array[].

```
const u16 factory_test_model_array[] = {
    SIG_MD_G_ONOFF_S, SIG_MD_LIGHTNESS_S, SIG_MD_FW_UPDATE_S,
    SIG_MD_LIGHT_CTL_S, SIG_MD_LIGHT_CTL_TEMP_S, SIG_MD_LIGHT_HSL_S,
    SIG_MD_LIGHT_XYL_S
};
const u16 factory_test_cfg_op_array[] = {COMPOSITION_DATA_GET, NODE_RESET};
Figure 5.1: Default testable commands
```

Controls under factory mode do not need provision, please refer to section 4.5. Please be noted, the mode needs all nodes unprovisioned, including gateway and master dongle.

# 6 Important SDK Modules

# 6.1 Configure Mesh SDK Default Feature

1) Mesh nodes describe their supporting features in composition data (model\_sig\_cfg\_s\_cps.pageO.head.feature), SDK initialization is shown as below:



Composition data defines supports or not, enable/disable can also be defined under "support" status.
 Please refer to the configuration action model\_sig\_cfg\_s.frid in mesh\_global\_var\_init().

Mesh_common.c		01329:	#el	se		$\sim$
	_	01330:	mod	el sid	r cfq	s.frid = FEATURE FRIEND EN ? FRIEND SUPPORT DISABLE : FRIEND NOT SUPPORT;
🕃 endif	7	01331:	#en	lif		
enable	-	01222.	#endif			
set_random_enable		01332.	#enurr			
publish_when_powerup		01333:				
# if WIN32		01334:	mod	el_sig	_cfg_	s gatt_proxy = FEATURE_PROXY_EN ? GATT_PROXY_SUPPORT_ENABLE : GATT_PROXY_NOT_SUPPORT;
St and I		01335:	mod	el sig	cfg	s.node identity def = NODE IDENTITY SUBNET SUPPORT DISABLE;
set_unprov_beacon_par		01336:	mod	el sig	, cfg	s.nw transmit.count = TRANSMIT CNT DEF;
set_provision_adv_dat		01337:	mod	el sig	r cfg	s.nw transmit.invl steps = TRANSMIT INVL STEPS DEF;
set_naterial_tx_cnd		01338:	#if	0 /	7 TES	T_CASE_NODE_CFG_CFGR_BV01_EN in pts7_3_1.exe
nesh_tx_cmd2normal_pr		01339:	mod	el sid	r cfa	s.relav = RELAY NOT SUPPORT;
a nesh_tx_cmd2uuid		01340:	#el	se		
SendOpParaDebug_vendo:		01341:	mod	el sid	t cfa	s, relay retransmit.count = TRANSMIT CNT DEF RELAY:
nesh_tx_cmd2self_prim		01011.				
<pre>1s_need_response_to_s</pre>		01342:	moa	91_S19	_cig_	s.relay_retransmit.invi_steps = TRANSMIT_INVL_STEPS_DEF_RELAY;
<pre>mesh_rsp_handle_cb</pre>		01343:	mod	el sig	r cfg	s celay = feature_relay en ? relay_support_enable : relay not_support;
my_fifo_push_hci_tx_f hci_sepd_data_user		01344:	#en	lif	_	
C CLETCHLY PALTY P						

Figure 6.2: Enable/disable the configuration

During working procedure, user can enable/disable these features with the following commands: CFG\_FRIEND\_SET, CFG\_RELAY\_SET and CFG\_GATT\_PROXY\_SET.

During working procedure, user can enable/disable these features with the following commands: CFG\_FRIEND\_SET, CFG\_RELAY\_SET and CFG\_GATT\_PROXY\_SET.

3) For default features of each compiling project, please refer to Demo Project in SDK Instruction.

# 6.2 Common Macro Definitions

Some common macros and APIs can also be found in "common api".

### 6.2.1 LIGHT\_CNT and ELE\_CNT\_EVERY\_LIGHT

See "Definition of the number of elements of a node" for an introduction.

### 6.2.2 ONPOWER\_UP\_SELECT

ONPOWER\_UP\_SELECT defines the state setting of the lamp when the lamp node is powered off and powered back on, which can be set to one of the following states:

- ONPOWER\_UP\_OFF: After powering up, the lamp is set to the OFF state.
- ONPOWER\_UP\_DEFAULT: After powering up, the lamp is set to the ON state.
- ONPOWER\_UP\_STORE: After powering up, the light stays in the same state as it was before the power was off.

See chapter "3.1.4 Generic OnPowerUp" in the mesh model spec "MshMDL\_v1.1.pdf" for details:

### 3.1.4 Generic OnPowerUp

The Generic OnPowerUp state is an enumeration representing the behavior of an element when powered up. The values for the state are defined in Table 3.3.

Value	Description
0x00	Off. After being powered up, the element is in an off state.
0x01	Default. After being powered up, the element is in an On state and uses default state values.
0x02	Restore. If a transition was in progress when powered down, the element restores the target state when powered up. Otherwise the element restores the state it was in when powered down.
0x03–0xFF	Prohibited

Table 3.3: Generic OnPowerUp states

Figure 6.3: OnPowerUpType.png

### 6.2.3 MESH\_POWERUP\_BASE\_TIME

It is used to define how long after a node has been powered up, and then after a random time, it starts sending lightness status or onoff status to notify the gateway or cell phone, etc. that the node is currently online.

See mesh\_vd\_init() and system\_time\_run() for details:

```
void mesh_vd_init()
{
.....
    publish_powerup_random_ms = rand() % 1500; // 0--1500ms
    STATIC_ASSERT(MESH_POWERUP_BASE_TIME >=200);
    publish_powerup_random_ms += MESH_POWERUP_BASE_TIME; // 200ms: base time.
```

### 6.2.4 Checking Whether a Node has been Provisioned

is\_provision\_success();

# 6.3 Definition of the Number of Elements of a Node

The number of elements a node contains can be one or more. It is determined by the values of these two macros: ELE\_CNT\_EVERY\_LIGHT and LIGHT\_CNT to determine that each element occupies a unicast address.

```
#define ELE_CNT
```

(LIGHT\_CNT \* ELE\_CNT\_EVERY\_LIGHT)

- ELE\_CNT\_EVERY\_LIGHT means a product unit consists of several elements, for example, a color temperature lamp consists of two elements. The reason why we need two elements is that the color temperature lamp has two states, brightness value and color temperature value. The brightness can be controlled by commands such as level set, and the color temperature can also be controlled by commands such as level set. If there is only one element, when the color temperature node receives the level set command, there is no way to distinguish whether to control the brightness or the color temperature, so it needs two elements. Similarly, HSL (RGB) light needs three elements.
- LIGHT\_CNT: Indicates that a BLE mesh module has several product units.

For products supporting server model, for example, when one BLE module drives two color temperature lamps, LIGHT\_CNT needs to be set to 2.

For products that support the client model, such as remote control products, such as Switch project, the destination address of control commands sent by keys can be modified by modifying the publish address. Hence, for the keys with the same command, the number of keys need to be independently configured with a publish address needs to be consistent with the number of keys need to be configured with LIGHT\_CNT. For example, the Switch project of demo SDK has 4 pairs of keys that send group address, so LIGHT\_CNT is set to 4.



### 6.4 Grouping Features and Share-model

For a description of the spec counterpart, please refer to "4.2.4 Subscription List" in MshMDL\_v1.1.pdf, among others.

 SIG Mesh spec defines that we can configure the group number independently for each model, so currently each model has a copy of the independent group number data, the maximum number of storage is 8 (SUB\_LIST\_MAX), as shown in the following figure.

```
typedef struct{
    u16 ele adr;
                      // use as primary address for model_sig_cfg_s_t
    u8 no pub :1; // means not support publish function
                :1; // means not support subscription function; must before pub and sub par
    u8 no sub
    u8 pub_trans_flag :1; // transition process was ongoing flag.
    u8 pub 2nd state :1; // eg: lightness and lightness linear.
    u8 rsv2;
    bind key t bind key[BIND KEY MAX];
    u8 pub uuid[16];
    cb pub st t cb pub st;
                              // no need to save, fix later
    u32 cb tick ms;
                               // no need to save, fix later
                     // pub_adr and pub_par must existed if sub_list existed // offset:32
    u16 pub adr;
    mesh model pub par t pub par;
    u8 rfu3[1]
   [116 sub list[SUB LIST MAX];
                                        // pub_adr, pub_par, sub_list must follow com if existed
    u8 sub uuid[SUB
                       IST MAX [16];
}model common t;
```

Figure 6.4: Configure group index

(2) The commands for models that use device key encryption and decryption do not support multicast addresses because the device key is different for each node. The models that use device key are config model and so on, see MODEL\_ID\_DEV\_KEY[] for details:

<pre>const u32 MODEL_ID_DEV_KEY[] = {</pre>	
SIG_MD_CFG_SERVER,	SIG_MD_CFG_CLIENT,
<pre>SIG_MD_REMOTE_PROV_SERVER,</pre>	<pre>SIG_MD_REMOTE_PROV_CLIENT, // no para</pre>
<pre>SIG_MD_DF_CFG_S,</pre>	SIG_MD_DF_CFG_C,
<pre>SIG_MD_BRIDGE_CFG_SERVER,</pre>	SIG_MD_BRIDGE_CFG_CLIENT,
<pre>SIG_MD_PRIVATE_BEACON_SERVER,</pre>	SIG_MD_PRIVATE_BEACON_CLIENT,
<pre>SIG_MD_SAR_CFG_S,</pre>	<pre>SIG_MD_SAR_CFG_C, // save in model_sig_cfg_s_t now.</pre>
<pre>SIG_MD_ON_DEMAND_PROXY_S,</pre>	<pre>SIG_MD_ON_DEMAND_PROXY_C, // save in model_sig_cfg_s_t now.</pre>
<pre>SIG_MD_LARGE_CPS_S,</pre>	SIG_MD_LARGE_CPS_C, // no para to save

};

(3) The sig mesh spec also stipulates that within the same element, those with state binding relationship or model extension relationship should share the group number information. For example, after configuring a group number for the onoff model, the lightness model will be automatically bound to this group number. So SUBSCRIPTION\_SHARE\_EN needs to be turned on by default. For details, please refer to "Summary of xxx models" in the model spec "MshMDL\_v1.1.pdf", e.g., "6.7 Summary of lighting models" in the Figure 6.12: Relationships between lighting models - Part 1". You can also check the sub\_share\_model\_sig\_onoff\_server\_extend[] in the SDK, which contains all the models that have extensions to the onoff model. (4) Some models have no state binding relationship with each other, for example, onoff model, vendor model, sensor model, you need to send the command to configure the group number for these three models. In some applications, there are some proprietary requirements, i.e., when binding the group number to a model, you want to automatically bind the group number to the models that do not have extended model or state binding relationship, so as to reduce the time of group number configuration. At this time, you can enable SHARE\_ALL\_LIGHT\_STATE\_MODEL\_EN to realize. Note that this is a custom rule. After enabling this macro switch, put the model IDs of the group numbers that need to be auto-bound together in the specified array. If it is a SIG model, put it in the array sub\_share\_model\_sig\_onoff\_server\_extend[]. If it is a vendor model, put it in sub\_share\_model\_vendor\_server\_extend[].

For other details, please refer to the codes corresponding to the macros SUBSCRIPTION\_SHARE\_EN and SHARE\_ALL\_LIGHT\_STATE\_MODEL\_EN.

# 6.5 Method for a Node to Get the Group Number

• Getting it through global variables

Each model has a list of group numbers, in the case of the onoff model, obtained through the model\_sig\_g\_onoff\_level.onoff\_srv[i].com.sub\_list[] to get it.

Double click on the value of model\_sig\_g\_onoff\_level in BDT tool to get its information. After getting the information, refer to the structure definition of model\_sig\_g\_onoff\_level and find the position of sub\_list to see the group number.
🗉 🗧 Telink

									<b>_</b>	$\sim$												
Device File View Tool Help																						
	🔋 Se <u>t</u> ting 🚺	Erase 👤	Download 🕈 Ac	ctiva	te 🕨 R <u>u</u> n	ш	<u>P</u> ause	*	<u>S</u> tep	Q, P	c ,	Sing	jle step	•	୯	<u>R</u> eset	•	ma <u>n</u> ua	al mode	• •	₽ċ	lear
Unlock     Cadge																						
b0 10	Ь0	10	2 SV	WS	602		06			I	Sta	all		602			88			►	Start	
Ŧ	Download				1	i祝 Tde	ebug									Π	Log	window	NS			
Variable Name	Addr	Len	Value	^	156a0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	^
mi_service_change_pr	42d42	1	00000020		456b0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
misc_flag	44da0	1	00000000		156d0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
model_sig_cfg_s	454c8	232			156e0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
model_sig_cfg_s_cps	42e0c	50			456f0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
model sig g onoff le	455b0	552			15710:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
model_sig_g_power_c	457d8	554			15720:	71	00	00	00	00	80	00	00	00	00	00	00	00	00	00	00	
model_sig_health	45068	404			45730:	00	00	00	00	00	00	00	00	39	81	00	00	00	00	00	00	
model_sig_light_ctl	45a38	552			$\frac{15740}{15750}$	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
model_sig_lightness	45c60	368			15760:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
model_vd_light	4617c	184			45770:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
mtu_rx_fifo	4698c	260			15780:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
mtu_tx_fifo	46a90	36			157a0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
my_OtaData	43c70	1	00000000		457b0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
my_OtaProp	42d50	1	00000016		157c0:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
my_PnPCharacter	42d49	1	0000002		btal Ti	me:	33	ms	00	00	00	00	00									
my_appearance	43c60	2	00000000		, cut it																	
my_appearanceChara	42d2c	1	0000002		, «																	>

Figure 6.5: Global variable to get group number

• Get group number by model ID

The p\_model pointer returned by this function mesh\_find\_ele\_resource\_in\_model() is then available via p\_model->sub\_list[].

• Get group number of all model ID

Get the group number of all model\_id's by iterating over all model global variables in MeshSigModelResource[] and then using the first method "get by global variable".

## 6.6 Heartbeat demonstration

The heartbeat function is detailed in the "heartbeat" section.

No heartbeat message is sent by default, user can configure this by sending command HEART-BEAT\_PUB\_SET. After the command is sent, the node will send out heartbeat message. Below is an example: send heartbeat message every 2 seconds, and the corresponding INI command:

```
CMD-cfg_hb_pub_set_sig
=a3 ff 00 00 00 00 00 00 02 00 80 39 01 00 ff 02 05 07 00 00 00
```

The parameters are described as following:

• 80 39: op code

- 01 00: destination address of heartbeat is 0x0001
- ff: CountLog, Oxff means infinity
- 02: PeriodLog, period is 2 powers (02-1) i.e. 2 seconds
- 05: InitTTL, set TLL value for network layter when sending heartbeat message. This value can be customized and is not required to be equal to model\_sig\_cfg\_s.ttl\_def, as it depends on how many hops the user wants the nodes to be in range to receive the heartbeat message.
- 07 00: features, once any of relay, friend, proxy feature changes status(switches between enable and disable) the heartbeat message will be immediately reported.
- 00 00: NetKeyIndex。

Heartbeat packet can be seen in firmware tool.

c. cfg_hb_pub_set_sig
<pre>&lt;0005&gt;21.31.00.235 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 02 00 80 39 01 00 ff 02 01 07 00 00</pre>
<pre>&lt;0006&gt;21:31:36:244 [INFO]:(Basic)the mesh access tx cmd is 0x3980 : 01 00 ff 02 01 07 00 00 00</pre>
<pre>&lt;0007&gt;21:31:36:344 [INFO]:(Basic)adr_src:0x0002,adr_dst:0x0001,access rx cmd is 0x6 : 06 00 01 00 ff 0</pre>
<pre>&lt;0008&gt;21:31:36:352 [INFO]:(cmd_rsp)Status Rsp : 02 00 01 00 06 00 01 00 ff 02 01 07 00 00</pre>
<pre>&lt;0009&gt;21:31:36:360 [INFO]:(log_win32)mesh_tx_reliable_stop: op 0x3900 rsp_max 1, rsp_cnt 1</pre>
<0010>21:31:36-205 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 00
<0011>21-31:38:412 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 00
<pre>&lt;001221:31:40:537 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 05 00</pre>
<pre>\$613&gt;21:31:42:612 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 00</pre>
<0014>21:31:44:734 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 00
<pre>&lt;0015&gt;21:31:46:812 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 00</pre>
<0016>21:31:48:932 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 00
<pre>&lt;0017&gt;21:31:51:012 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 00</pre>
<pre>&lt;0018&gt;21:31:53:132 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 04</pre>
<0019 21:31:55:212 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0a 01 07 00
<0020>21:34-57:333 [INFO]:(common)heartbeat src adr is 0x0002,dst adr is 0x0001: 0= 01 07 00

Figure 6.6: Heartbeat packet

- Oa: heartbeat opcode, please note that heartbeat is control message.
- 05: InitTTL, same value as that of heartbeat set message parameter.
- 07: Features, same value as that of heartbeat set message parameter.

When the receiver receives a heartbeat message, it will execute the callback mesh\_process\_hb\_sub(), in which it can get the InitTTL value of the heartbeat message access layer parameter area and the ttl value of the network layer, and then subtract the two values, it will be able to know how many hops the heartbeat has gone through before it reaches the current node.

Taking the example that the ttl value of the network layer of the received heartbeat is equal to 2, the specific calculation is: hops =  $p_hb$ ->iniTTL- ( $p_bear$ ->nw.ttl) + 1 = 5 - 2 + 1 = 4; that is to say, the heartbeat has gone through 4 hops before it reaches the current node.

# 6.7 Mesh ADV Send Timing

The SDK user\_init initialization calls bls\_set\_advertise\_prepare (app\_advertise\_prepare\_handler) to register the broadcast packet send callback function, the user is allowed to access and modify the contents of the



broadcast packet before sending the broadcast. In app\_advertise\_prepare\_handler(rf\_packet\_adv\_t \* p), the p pointer points to the data area to be sent, and modifying the contents pointed to by p modifies the contents to be sent.

After registering app\_advertise\_prepare\_handler(), this function will be called to broadcast packets once every 10ms by default, which is defined by ADV\_INTERVAL\_MIN. The reason for defining 10ms is that the unit of network transmit interval is 10ms. For example, if the network transmit interval is equal to 2, the initial value of mesh\_tx\_cmd\_busy\_cnt is set to network\_tx\_cmd\_busy\_cnt when sending a network message. Whenever app\_advertise\_prepare\_handler() is called, mesh\_tx\_cmd\_busy\_cnt will be reduced by one, and then after reduced to 0, it will do the delay of 0~10ms, which means that it realizes the instantaneous transmission interval defined by the spec, and then the RF packet corresponding to the next transmit count can be sent.

# 6.8 API for Mesh ADV Payload Setting

## 6.8.1 Unprovisioned Device Beacon

The unconnectable broadcast packets sent by an unprovision device are for ADV provisioner discovery. The corresponding payload setting API is unprov\_beacon\_send(), which push the data into the mesh\_adv\_cmd\_fifo via mesh\_tx\_cmd\_add\_packet(), and then checks the mesh\_adv\_cmd\_fifo in app\_advertise\_prepare\_handler() and sends out the data when it sees data. For details of the data format, please refer to the section "3.10.2 Unprovisioned Device beacon" in the V1.1 spec. For details of the sample data, please refer to the section "8.4 Beacon sample data".

## 6.8.2 Mesh Provisioning Service Advertising

A connectable broadcast packet sent by an unprovision device are for discovery by the GATT provisioner. The corresponding payload setting API is set\_adv\_provision(). For details of the data format, see section "7.1.2.2.1 Advertising" of the V1.1 spec. For sample data, see "8.5 Provisioning Service sample data".

## 6.8.3 Mesh Secure Network Beacon

The unconnectable broadcast packet sent by an provisioned node is mainly used to broadcast the IV index, as well as IV update, and for the key refresh process. The corresponding payload setting API is mesh\_tx\_sec\_private\_beacon\_proc(). See "3.10.3 Secure Network beacon" in this section of the V1.1 spec for details on the data format. See "8.4 Beacon sample data" for the details of sample data.

## 6.8.4 Mesh Proxy ADV

The connectable broadcast packets sent after successful provisioning are for discovery and connection by the GATT proxy client. It contains network ID and node identity. The corresponding payload setting API is set\_adv\_proxy(). See "7.2.2.2.1 Advertising" in this section of the V1.1 spec for details on the data format. For sample data, see section "8.6 Mesh Proxy Service sample data".

# 6.9 Mesh Receiving Transmitting Self-defined Packet

#### Self-defined Packet Transmitting

When it is needed to send beacons that are not defined in the mesh spec, such as ibeacon, set BEA-CON\_ENABLE to 1. See BEACON\_ENABLE related code for details.

The SDK will call bls\_set\_advertise\_prepare (app\_advertise\_prepare\_handler) to register packet and send call back function when initialization, the packet can be visited and modified before sending out, SDK call the packet sending function once every 10ms by default. If user want to send self-defined packet, send it in a similar way of sending mesh-connectable packet in gatt\_adv\_prepare\_handler. Control packet sending interval by clock\_time\_exceed software timing, rf\_packet\_adv\_t \* p to packet to be sent, user can modify the contents of the packet pointed to by p according to packet format(please refer to set\_adv\_provision()), then set the return value ret to 1, means will send packet.

#### Receiving/Filtering Connectable Packet

The SDK call adv\_filter\_proc() during RF rx interrupt to filter received packets, return 0 to abandon received packet, return 1 to keep this packet, receive and compress into blt\_rxfifo without filtering. All connectable packet will be filtered by default. If user want to receive connectable packet, then open USER\_ADV\_FILTER\_EN, in user\_adv\_filter\_proc(), set the packet you want to return 1. It is not recommended to set all connectable packet to return 1, because this will do no filter to the packets, all packets will be pushed into blt\_rxfifo, including those packets sent by other no-mesh BLE products, this may be beyond the storage capability of our receiving buffer, thus result in losing mesh message as well as the mesh packet receiving.

blt\_sdk\_main\_loop () will check blt\_rxfifo, if there is data need to be processed, it will call app\_event\_handle(), use may process the received connectable packet in the if(LL\_TYPE\_ADV\_NONCONN\_IND ! = (pa->event\_type & OxOF)) branch of this callback function, as shown below:

AN-17120400-E7

```
int app_event_handler (u32 h, u8 *p, int n)
ł
    static u32 event cb num;
    event cb num++;
   int send to hci = 1;
   if (h == (HCI_FLAG_EVENT_BT_STD | HCI_EVT_LE_META)) //LE event
    {
       u8 subcode = p[0];
       #if MI API ENABLE
       telink_ble_mi_app_event(subcode, p, n);
       #endif
    //----- ADV packet -----
       if (subcode == HCI SUB EVT LE ADVERTISING REPORT) // ADV packet
            event_adv_report_t *pa = (event_adv_report_t *)p;
            if (LL TYPE ADV NONCONN IND != (pa->event type & 0x0F))
               return 0;
            #if DEBUG MESH DONGLE IN VC EN
            send to hci = mesh dongle adv report2vc(pa->data, MESH ADV PAYLOAD);
            #else
           send to hci = app event handler adv (pa->data, ADV FROM MESH, 1);
            #endif
        }
                   Figure 6.7: Receiving and filtering connectable packet
```

# 6.10 Method to Modify the Maximum Number of Nodes in a Mesh Network

The Mesh products need to set the maximum number of nodes in the design phase, otherwise when sending a command to get a certain state of all nodes, such as Lightness Get All, the cache buffer will not be enough to store the location, resulting in a Lightness Get All being processed repeatedly.

The maximum number of nodes is set by MESH\_NODE\_MAX\_NUM (default 105).

```
#if WIN32
#define MESH_NODE_MAX_NUM 1000 // 1000
#elif (FEATURE_LOWPOWER_EN)
#define MESH_NODE_MAX_NUM 105 // no need to many for LPN to save retention RAM.
#elif DEBUG_CFG_CMD_GROUP_AK_EN
#define MESH_NODE_MAX_NUM 305
#else
#define MESH_NODE_MAX_NUM 105 // gateway and node should keep the same, because of mesh
$\Gamma$ command cache..
#endif
```

To modify the number of network nodes: Modify MESH\_NODE\_MAX\_NUM (default 105) to the desired value. Note The gateway and nodes should be configured to the same value.



The corresponding RAM consumption for each additional node is shown in the following table:

	must	must	must	must	depend on MD_REMOTE_PROV (remote provision)	depend on MD_MESH_OTA_EN (device firmware update)	total
	sizeof (cache buf t)	sizeof (status record t)	gateway_seg_buf	sizeof (VC node info)	sizeof(VC_node_info_t ->dev key candi)	sizeof(fw_distribut_srv_proc_t ->fw receiver list t)	(Tax)
mesh node	6	0	0	0	0	0	6
gateway	6	4	2	20	16	7	55

Figure 6	5.8:	RAM_	_Cost_	_for_	_each_	_node
----------	------	------	--------	-------	--------	-------

The cache\_buf is used to cache sequence number and so on to cache the sequence number of all nodes, etc.

Note: After the gateway sets MESH\_NODE\_MAX\_NUM more than 200, an error will be prompted here when compiling:

```
STATIC_ASSERT(ARRAY_SIZE(gw_node_info) <= (FLASH_ADR_VC_NODE_INFO_END -
Grad FLASH_ADR_VC_NODE_INFO)/sizeof(VC_node_info_t)); // make sure enough flash area to save</pre>
```

Because the default 4KB flash sector can not store so many nodes' information, so you need to find a contiguous flash area to store. Then modify FLASH\_ADR\_VC\_NODE\_INFO and FLASH\_ADR\_VC\_NODE\_INFO\_END accordingly.

# 6.11 Telink Customized Mode for Sending Mesh Messages via Extended Broadcast Package extend\_adv

Extended advertising packet: extend ADV

## 6.11.1 Function Introduction

The B85 chip and protocol stack support sending extend ADV, but the SIG mesh spec does not define sending mesh messages via extend ADV yet. In some scenarios, we need to use extend ADV to improve the efficiency of sending messages, such as transmitting compressed image data, performing mesh OTA, etc. Therefore, we define a mode to send messages via extend ADV.

This mode specifies that one of the formats of the extend ADV defined by the BLE spec is used, as shown below:

Low Energy Overview Message Log 🗮 Spectrum	4 ▷ x	Details	<b>д х</b>
Protocol: Single 🗸 All layers 🔶 🛹 🚥 🧆 🖗 📳 🕑 🖇 🎝 🧑 🚗   29 items displayed 🛛 🗸 🔍	🔕 🔹 Search 🔹 🛛 🔮	▼ All fields Show in overview Display	- Search -
Item Item	Time 🕂 🗸 🔿	Name	Value
🖃 🛞 Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 miliseconds)	16:50:43.464 310 400	Master Address	A4:C1:38:B3:5F:DC
B B B Mesh Access Message (Seq=0x007C35, NID=0x53, SRC=0x0042.u, DST=0xFFFF.a, AID=0x33)	16:50:43.464 310 400	Slave Address	Unknown BD_ADDR
G a S Mesh Access Message (ADV, NID=0x53, SRC=0x0042.u, DST=0xFFFF.a, AID=0x33)	16:50:43.464 310 400	😑 📲 Link-Layer Packet	
🖃 🙀 AUX_ADV_IND Packet (A4:C1:38:B3:5F:DC, AdvA   AdvDataInfo   Adv Data, #79->, Mesh Message, NID=0x53) (1.17 ms)	16:50:43.464 310 400	E de Mandar	
*** AUX_ADV_IND Packet (A4:C1:38:B3:5F:DC, AdvA   AdvDataInfo   Adv Data, #79->, Mesh Message, NID=0x53)	16:50:43.464 310 400	DDUTune	ALIX ADV THD
⊞	16:50:43.503 793 200	4 DEL	ADX_ADV_IND
⊞	16:50:43.533 750 500	( PEU (chsal)	Reserved (0)
⊕ 🔐 🚱 Mesh Access Message (ADV, NID=0x53, SRC=0x0042.u, DST=0xFFFF.a, AID=0x33)	16:50:43.564 321 000	A TyAdd	Public
⊕ 🔐 🚱 Mesh Access Message (ADV, NID=0x53, SRC=0x0042.u, DST=0xFFFF.a, AID=0x33)	16:50:43.603 580 400	( PEL(PyAdd)	Reserved (0)
⊕ 😭 🐼 Mesh Access Message (ADV, NID=0x53, SRC=0x0042.u, DST=0xFFFF.a, AID=0x33)	16:50:43.638 417 200	Payload Length	136 hytes
🕃 😡 Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:43.964 042 500	Evtended Header Length	9 bytes
🕃 💮 Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:44.464 794 600	Adv Mode	Non Contertichie / Non Scannable
🟵 🛞 Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:44.965 406 700	Extended Header	Hor connectable for Scandole
🗷 🛞 Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:45.466 708 900	Elags	AdvA   AdvDataInfo   Adv Data
🟵 🛞 Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:45.967 793 400	Advertising Address	44-C1-38-B3-5E-DC
🗉 🛞 Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:46.468 893 200	Adv Data Info	
🕀 🛞 Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:46.969 512 900	Advertising Data ID (DID)	0x665
🕀 🛞 Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:47.470 676 400	Advertising Set ID (SID)	0x0
Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:47.970 619 500	Advertising Data	
🕀 🎲 Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:48.471 568 200	Raw Data	7D 2A 53 E1 D9 ED 46 DE 25 1C E
Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:48.973 736 700	🖃 🔧 Mesh Message	
🕀 🚱 Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:49.474 302 500	Length	125
🕀 😡 Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:49.974 282 600	Data Type	Mesh Message
Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:50.475 901 400	VI VI	0x0
Mesh Generic OnOff Set Unacknowledged (On Off-Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:50.977 019 500	🔷 NID	0x53
Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:51.478 580 400	Obfuscated Network Header	F1 D9 ED 46 DF 25
Mesh Generic OnOffSet Unacknowledged (On Off-Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:51.978 016 500	Encrypted data and MIC	1C F2 A8 0E 51 BA C3 C3 D3 4D
Mesh Generic OnOff Set Unacknowledged (On Off-On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:52.478 711 900	Non-significant Part	0 bytes
Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:52.978 429 100		Valid
Mesh Generic OnOff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:53.479 321 000	😑 🔩 Raw Content	
Mesh Generic OnOff Set Unacknowledged (On Off=Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:53.979 714 200	Raw Data	07 88 09 09 DC 5F B3 38 C1 A4 6 🗸
Mesh Generic UnUff Set Unacknowledged (On Off=On, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:54.480 798 900	Details	
Mesh Generic OnOff Set Unacknowledged (On Off-Off, Number of Steps=Immediate, Step Resolution=100 milliseconds)	16:50:54.980 352 400		

Figure 6.9: extend\_ADV format

And the ADV payload is increased from 31 bytes to 245 bytes (ADV\_EXTEND\_PAYLOAD\_MAX), that is, the length of network PDU is increased by 214bytes (CONST\_DELTA\_EXTEND\_AND\_NORMAL), and the logic of other packet sending remains unchanged, including transmit interval and transmit count. When the sent access layer, i.e. (opcode + parameters) exceeds (11+214 = 225) bytes, the segment packet grouping process will also be executed.

To summarize, when transmitting at full load, the packet sending speed is increased to about 225/11 = 20 times the original speed.

## 6.11.2 Test Methods

Telink

## 6.11.2.1 Node Configuration

The EXTENDED\_ADV\_ENABLE of firmware SDK is set to 1.

After enabling EXTENDED\_ADV\_ENABLE, by default, firmware SDK only sends mesh OTA command in extended ADV format, such as FW\_UPDATE\_START, BLOB\_CHUNK\_TRANSFER and BLOB\_BLOCK\_STATUS, while other packets are still sent in the same way as before, i.e., segmented packets, because we have to consider the common commands can also be interconnected with that from other manufacturers. See function: is\_not\_use\_extend\_adv(); for details. (Note: Older versions before V3.3.3 are is\_extend\_unseg2short\_unseg()).

If there is a need for all commands with access layer less than 225 bytes (including op code) to be unsegment packet, return 0 in is\_not\_use\_extend\_adv().

If adding the rule that all vendor op codes are sent in extend ADV format, then return 0 in is\_not\_use\_extend\_adv() when judging that (IS\_VENDOR\_OP(op)) returns 1.

## 6.11.2.2 Provisioner Configuration

• sig\_mesh\_tool.exe Upper Configuration

The value of the "ExtendAdv" control needs to be modified in the following way:

CMD sig_mesh_master.ini 💌 INI BULKOUT ASCI	I 🔽 :	log 🗆 AutoSaveLog 2 retry Clear Save Save 🔽 Hex 🗆 Adv Stop Scan rp_scan ex_scan OTA Rx test
met_level LPN_get_level LPN_get_onoff	^	fastbind Extend Adv: None     ✓     cononsidered 24.022 (INEN) None     Fastbind Extend Adv: None
lightness_get_Panel Note:retry count field of LPN distrib start is chan LPN fw distrib ota start 04	je	<pre>&lt;0000&gt;14:02:24:072 [INFO 0TA Only Unidex searching:: 12 34 56 78 12 34 56 78 00 00 00 00 00 00 00 00</pre>
LPN_initiator_start_v_apply4 LPN_initiator_start_v_only4		
fw_update_info_get fw_update_info_get_all		



See is\_not\_use\_extend\_adv() for details on how to handle this function.

- (1) Select None.
- (2) Select OTA only, then the host computer only sends the default FW\_UPDATE\_START, BLOB\_CHUNK\_TRANSFER, BLOB\_BLOCK\_STATUS, BLOB\_PARTIAL\_BLOCK\_REPORT and other ops to the mesh OTA with extend ADV(). The purpose of sending with extend ADV is only to speed up the mesh OTA, other commands are compatible and nodes that do not support extend ADV can control each other.
- (3) Selecting all means that the host computer sends all commands with an access layer length (opcode + parameters) less than 225 bytes in single-packet extend ADV format.
  - Mobile App Configuration

Please turn on the Extended Long Pack option:

setting – setting – Extend Bearer Mode select "Extend GATT & ADV".

## 6.11.2.3 Precaution

Currently, only the B85 and B91 support the extend ADV function. Other chip models are not supported at this time.

# 6.12 Application of Soft Timer

## 6.12.1 Introduction of Soft Timer

(This is just an introduction to how to use it, details can be found in the B85 single connection handbook).

In order to facilitate users to do some simple timer tasks, Telink BLE SDK provides blt software timer demo, and all the source code is provided. Users can use the timer directly after understanding its design idea, or they can do some modification design by themselves.

The soft timer is especially suitable for adding timer tasks in low-power applications, so that the timer can be woken up to complete the timer tasks even in the sleep state. The soft timer can also be used in non-low-power applications.



The source code is in vendor/common/blt\_soft\_timer.c and blt\_soft\_timer.h. If you want to use it, change the following macros to 1 first:

#define BLT\_SOFTWARE\_TIMER\_ENABLE 0 //enable or disable

The blt soft timer is a query timer designed based on system tick, its accuracy cannot be as accurate as hardware timer, and it needs to be queried all the time in main\_loop.

We have scheduled: blt soft timer is used when the timing time is more than 5ms, and the requirement of time error is not particularly high. The most important feature of blt soft timer is that it is not only queried in main\_loop, but also ensures that the timer can be woken up and executed in time after entering suspend, which is based on the "application layer wake-up timer" introduced in the section of low-power wake-up. Currently, the design supports up to 4 timers running at the same time, actually users can modify the following macros to realize more or less timers.

#define MAX\_TIMER\_NUM 4 //timer max number

## 6.12.2 Soft Timer Initialization

Call the following API for initialization:

#### void blt\_soft\_timer\_init(void):

It can be seen that the initialization on the source code registers blt\_soft\_timer\_process as a callback function for the application layer to wake up early.

```
void blt_soft_timer_init(void){
```

bls\_pm\_registerAppWakeupLowPowerCb(blt\_soft\_timer\_process);

#### }

## 6.12.3 Query Processing for Soft Timer

The query processing of the blt soft timer is implemented using the blt\_soft\_timer\_process function:

#### void blt\_soft\_timer\_process(int type):

The type of blt\_soft\_timer\_process parameter has the following two cases: O means querying the function in main\_loop, and 1 means the function is accessed when an early timer wakeup occurs.

#define MAIN\_LOOP\_ENTRY 0
#define CALLBACK\_ENTRY 1



## 6.12.4 Task Configuration of Soft Timer

If the user wants to use a timer to realize certain functions, he can use the following API to add a timer task, using the method ():

(1) Define your own soft timer function: this function's function is to send ALL ON commands periodically (just an example).

```
int soft_timer_switch_send_all_on(void)
{
    access_cmd_onoff(ADR_ALL_NODES, 0, G_ON, CMD_NO_ACK, 0);
    LOG_USER_MSG_INFO(0, 0, "%s", __func__);
    return 0;
}
```

(2) Adding timer task

Use the following API to add.

#### int blt\_soft\_timer\_add(blt\_timer\_callback\_t func, u32 interval\_us);

func is a task function to be executed periodically; interval\_us is the timing time in us.

The int return value of the timed task func is handled in three ways:

- If the return value is less than 0, the task is automatically deleted after execution. You can use this feature to control the number of times the timer is executed.
- Returns 0, the previous interval\_us is always used for timing.
- If the return value is greater than 0, the return value is used as the new timer period in us.

## 6.12.5 Task Deletion of Soft Timer

In addition to using the above return value less than 0 to automatically delete a timer task, you can also use the following API to specify the timer task to be deleted.

int blt\_soft\_timer\_delete(blt\_timer\_callback\_t func):

## 6.12.6 Example of Soft\_timer Cycle Send Command

The following example implementation is based on the 8258\_mesh\_switch project.

- (1) Turn on BLT\_SOFTWARE\_TIMER\_ENABLE.
- (2) In the execution of blt\_soft\_timer\_init(); after adding blt\_soft\_timer\_add() can be. The sample code is as follows:

The following code starts a soft timer task by pressing the key RC\_KEY\_R for the first time. This task is to call soft\_timer\_switch\_send\_all\_on() every 500ms to send a command. When the time is not up, the node is in sleep state.

Pressing key RC\_KEY\_R again closes this task and stops sending commands.

```
void mesh_proc_keyboard ()
{
    .....

else if (kb_event.keycode[0] == RC_KEY_R){// will enter here once when RC_KEY_R is pressed
    and release.
    static u32 press_cnt = 0;
    press_cnt++;
    if(press_cnt & 0x01){
        blt_soft_timer_add(soft_timer_switch_send_all_on, 500 * 1000); //
    }else{
        blt_soft_timer_delete(soft_timer_switch_send_all_on);
    }
  }
}
```

For the definition of the return value of soft\_timer\_switch\_send\_all\_on(), please refer to the parameter "func" of blt\_soft\_timer\_add function.

# 6.13 Use of the Long Sleep Interface

The Long sleep interface is not recommended for any sleep within 230 seconds. This is because the timing method needs to be modified after waking up and the timing accuracy is reduced a bit. Also the SUS-PEND\_MODE mode of long sleep should not be used because of the high power consumption.

Using the Long Sleep interface, you can set the sleep time to a maximum of 37 hours.

## 6.13.1 Function Name

```
/**
 * @brief This function servers to wake up the cpu from sleep mode.
 * @param[in] sleep_mode - sleep mode type select.
 * @param[in] wakeup_src - wake up source select.
 * @param[in] wakeup_tick - the time of sleep.unit is 31.25us,1ms = 32.
 * @return indicate whether the cpu is wake up successful.
 */
int cpu_long_sleep_wakeup(SleepMode_TypeDef sleep_mode, SleepWakeupSrc_TypeDef wakeup_src,
 * unsigned int wakeup_tick);
```

## 6.13.2 Use Methods

- Enable MESH\_LONG\_SLEEP\_WAKEUP\_EN.
- If there is a need for deep retetion mode, PM\_DEEPSLEEP\_RETENTION\_ENABLE needs to be enabled.



• Call cpu\_long\_sleep\_wakeup.

Test Example:

(1) Call cpu\_long\_sleep\_wakeup(SUSPEND\_MODE, PM\_WAKEUP\_TIMER, 40 \* 32 \* 1000) in the main loop; means to set up a wakeup after 20s of sleep, and add a log to record the current time before running the long sleep function.

```
LOG_USER_MSG_INFO(0, 0, "system_time_s: %d, system_time_100ms: %d, system_time_ms: %d!",

→ system_time_s, system_time_100ms, system_time_ms);

cpu_long_sleep_wakeup(DEEPSLEEP_MODE_RET_SRAM_LOW32K, PM_WAKEUP_TIMER, 40 * 32 * 1000);
```

- (2) Observe if the time result printed by the log is the same as the sleep time.
- (3) Observe if system\_time\_100ms, system\_time\_s are accurate.

```
[14:40:51.484 (Rx)]
[USER]:(USER)Start user init...
[14:40:51.541 (Rx)]
[USER]:(USER)system_time_s: 0, system_time_100ms: 0, system_time_ms: 5!
[14:41:31.540 (Rx)]
[LIB]:(sdk)mesh tx NoAck,op:0x4e82(LIGHTNESS_STATUS),src:0x19f0,dst:0xffff,sno:0x000001 par_len:5 par:00 00 ff ff 0a
[14:41:31.558 (Rx)]
[USER]:(USER)system_time_s: 40, system_time_100ms: 400, system_time_ms: 40011!
[14:42:11.537 (Rx)]
[USER]:(USER)system_time_s: 80, system_time_100ms: 800, system_time_ms: 80026!
```

Figure 6.11: Long sleep 40s test

(4) Modify the 40 seconds in step 1 to 600 seconds, and repeat steps 2 and 3.

# 6.14 Wakeup Source Identification Interface

The wakeup source identification interface can get the current wakeup source, there are four wakeup sources in total.

#### enum{

CPU_POWER_RESET,	// Power-on reset wakeup
CPU_WATCHDOG_RESET,	// Watchdog reset wakeup
CPU_PAD_WAKEUP,	<pre>// Wake up with a button</pre>
CPU_TIMER_WAKEUP,	// Timed wakeup

};

/\*\*

## 6.14.1 API Function Name

**(brief** This function server to get cpu wakeup source

```
* @return
                 CPU_WATCHDOG_RESET: watchdog reset.
                 CPU_PAD_WAKEUP: gpio wakeup.
                 CPU_TIMER_WAKEUP: timer wakeup.
                 CPU_POWER_RESET: power reset.
 *
                function called must be after "cpu_wakeup init()" and before wakeup io
 * @note
   setting(if exist).
\hookrightarrow
 */
int get_cpu_wakeup_source()
{
    if(read_reg8(0x72) & BIT(0)){
        write_reg8(0x72, BIT(0)); // manual clear watchdog reset flag after read.
        return CPU_WATCHDOG_RESET;
    }
    u8 val = analog read(0x44);
    if((val & WAKEUP_STATUS_TIMER_PAD ) == WAKEUP_STATUS_PAD){
        return CPU_PAD_WAKEUP;
    }
    else if((val & WAKEUP_STATUS_TIMER_CORE ) == WAKEUP_STATUS_TIMER_CORE){
        return CPU_TIMER_WAKEUP;
    }
    return CPU_POWER_RESET;
}
```

## 6.14.2 Use Methods

- (1) Call int get\_cpu\_wakeup\_source() where you need to get the wakeup source, and get the wakeup source based on the return value.
- (2) After calling the wake-up source test interface, clear bit(0) of digital register 0x72 to zero, otherwise the next call to the function will default to recognizing the wake-up source as a watchdog wake-up.

#### Note:

The wakeup source recognition interface should not be called until after cpu\_wakeup init(), because the MCU is not yet able to perform analog\_read() to read the analog registers before cpu\_wakeup init() is executed.

# 6.15 Key Scanning

The demo sdk turns on UI\_KEYBOARD\_ENABLE to enable the key scanning function, which detects the input of matrix keyboard keys or buttons.

The keypad detection is described in detail in the "Keystroke Scanning" section of this document, AN-21112301-C\_Telink B85m BLE Single Connection SDK Developer Handbook.pdf. The download link is:



Chinese version:

AN-21112301-C\_Telink B85m BLE Single Connection SDK Developer Handbook.pdf

English version:

AN-21112300-E\_Telink B85m BLE Single Connection SDK Developer Handbook.pdf

The sdk has already been adapted for the 8258 dongle, 8258 development board and 8258 Switch PCBA by default. If you want to redo the PCBA, you can focus on the following aspects:

## 6.15.1 Matrix Keyboard Mode

- (1) KB\_LINE\_MODE set to 0.
- (2) KB\_LINE\_HIGH\_VALID valid level setting, O for low validity, 1 for high validity.
- (3) KB\_DRIVE\_PINS, KB\_SCAN\_PINS modified per new PCBA.
- (4) Configure the FUNCTION, INPUT, and pull-down attributes of the GPIOs corresponding to KB\_DRIVE\_PINS and KB\_SCAN\_PINS.
- (5) If there is a need to modify the KEY map, just modify KB\_MAP\_NORMAL.

## 6.15.2 Button Mode

- (1) KB\_LINE\_MODE set to 1.
- (2) KB\_LINE\_HIGH\_VALID valid level setting, 0 for low validity, 1 for high validity.
- (3) KB\_DRIVE\_PINS has no real meaning, just set it to 0 or the first GPIO of the scan pin, KB\_SCAN\_PINS is modified according to the new PCBA.
- (4) Configure the FUNCTION, INPUT, and pull-down attributes of the GPIO corresponding to KB\_SCAN\_PINS.
- (5) If there is a need to modify the KEY map, just modify KB\_MAP\_NORMAL.

# 7 Vendor Model Introduction

# 7.1 Adding vendor model

Normally, it is not necessary for users to add model, because currently SIG model is completed, vendor model can use the already added vendor model: VENDOR\_MD\_LIGHT\_C, VENDOR\_MD\_LIGHT\_S, only need to add op code.

If user want to publish multiple status for current vendor model, new model need to be added. It not recommended to do so. Please contact us, or refer to MD\_SCENE\_EN to add new model.

# 7.2 Adding vendor command register reference

## 7.2.1 vendor\_opcode

Command and opcode are refer to the same item in this article, op code (1BYTE) + vendor id(2BYTE).

Vendor model have 64 op codes in total. Note, it's not each product type has 64, it's all the product with the same vendor id in the whole mesh network has 64 in total. When MESH\_USER\_DEFINE\_MODE chooses MESH\_NORMAL\_MODE, 32 must be reserved for Telink, i.e., 0xCO—0xDF, and 0xEO—0xFF is for users. It it recommended to use them by sub-commands way. Some Telink self-defined function will be disabled if user use more than 32 op codes. Please contact us in this case.

The maximum length of vendor command parameters is 377byte, but SIG mesh bottom layer will automatically de-pack packets longer than 8 byte, and the efficiency will be reduced. Therefore, it is recommended that keep frequently used control commands not longer than 8 byte.

## 7.2.2 Steps of Adding Vendor Opcode

VENDOR\_OP\_MODE\_SEL is set to the default VENDOR\_OP\_MODE\_DEFAULT.

In order to facilitate the user to quickly add the vendor opcode, the user can directly use the opcode demo defined by VENDOR\_OP\_USER\_DEMO\_EN, and no need to add a new opcode. the user can directly use these four opcodes and only need to change the corresponding callback functions cb\_vd\_user\_demo\_\_\_\_\_\_ set(), cb\_vd\_user\_demo\_get() to the expected function, then it can be used quickly. In addition, users adding new vendor opcode can also refer to the opcode demo defined by VENDOR\_OP\_USER\_DEMO\_EN to add more vendor opcode.

#### #if (VENDOR\_OP\_USER\_DEMO\_EN)

CMD\_NO\_STR(VD\_MESH\_USER\_DEMO\_SET, 0, VENDOR\_MD\_LIGHT\_C, VENDOR\_MD\_LIGHT\_S,

- → cb\_vd\_user\_demo\_set, VD\_MESH\_USER\_DEMO\_STATUS),
  CMD\_NO\_STR(VD\_MESH\_USER\_DEMO\_GET, 0, VENDOR\_MD\_LIGHT\_C, VENDOR\_MD\_LIGHT\_S,
- cb\_vd\_user\_demo\_get, VD\_MESH\_USER\_DEMO\_STATUS),
   CMD\_NO\_STR(VD\_MESH\_USER\_DEMO\_SET\_NOACK, 0, VENDOR\_MD\_LIGHT\_C, VENDOR\_MD\_LIGHT\_S,
- $\hookrightarrow$  cb\_vd\_user\_demo\_set, STATUS\_NONE),

Additionally, if there is data to be stored, call mesh\_common\_store() inside cb\_vd\_user\_demo\_set().

```
mesh_common_store(FLASH_ADR_MD_VD_LIGHT);
```

Then the parameter sno\_vd\_user\_demo added to model\_vd\_light\_t will be stored in this sector of flash FLASH\_ADR\_MD\_VD\_LIGHT. After re-powering up, sdk has implemented mesh\_flash\_retrieve() to read the contents of the sector to the corresponding global variable. Note that the structure and size of model\_vd\_light\_t must not change before or after OTA, otherwise the data will be read abnormally after OTA.

This document takes the group of commands VD\_GROUP\_G\_SET / VD\_GROUP\_G\_GET / VD\_GROUP\_G\_SET\_NOACK / VD\_GROUP\_G\_STATUS as an example to introduce the following.

## 7.2.2.1 Add Definition of Vendor Opcode

```
// op cmd 11xxxxxx yyyyyyy yyyyyyy (vendor)
// -----from 0xC0 to 0xFF
#if (VENDOR_OP_MODE_SEL == VENDOR_OP_MODE_SPIRIT)
    . . . . . .
#elif(VENDOR_OP_MODE_SEL == VENDOR_OP_MODE_DEFAULT)
// ----- 0xC0 to 0xDF for telink used
    . . . . . .
#define VD_GROUP_G_GET
                                      0xC1
#define VD_GROUP_G_SET
                                      0xC2
#define VD_GROUP_G_SET_NOACK
                                      0xC3
#define VD_GROUP_G_STATUS
                                      0xC4
   . . . . . .
#endif
```

## 7.2.2.2 Add Registration of Vendor Opcode

Vendor model registration reference code vendor\_model.c implementation of mesh\_cmd\_sig\_func\_t const mesh\_cmd\_vd\_func[] = {...}.

```
const mesh_cmd_sig_func_t mesh_cmd_vd_func[] = {
#if (VENDOR_OP_MODE_SEL == VENDOR_OP_MODE_SPIRIT)
.....
#elif(VENDOR_OP_MODE_SEL == VENDOR_OP_MODE_DEFAULT)
.....
```

```
CMD_NO_STR(VD_GROUP_G_SET, 0, VENDOR_MD_LIGHT_C, VENDOR_MD_LIGHT_S, cb_vd_group_g_set,
VD_GROUP_G_STATUS),
CMD_NO_STR(VD_GROUP_G_GET, 0, VENDOR_MD_LIGHT_C, VENDOR_MD_LIGHT_S, cb_vd_group_g_get,
VD_GROUP_G_STATUS),
CMD_NO_STR(VD_GROUP_G_SET_NOACK, 0, VENDOR_MD_LIGHT_C, VENDOR_MD_LIGHT_S, cb_vd_group_g_set,
STATUS_NONE),
CMD_NO_STR(VD_GROUP_G_STATUS, 1, VENDOR_MD_LIGHT_S, VENDOR_MD_LIGHT_C, cb_vd_group_g_status,
STATUS_NONE),
Hendif
.....
```

};

#### Note:

mesh\_cmd\_vd\_func[] is a const array, so this is a read-only array, thus can save RAM space.

## 7.2.2.3 mesh\_cmd\_sig\_func\_t introduction

```
typedef struct{
    u16 op;
    u16 status_cmd;
    u32 model_id_tx;
    u32 model_id_rx;
    cb_cmd_sig2_t cb;
    u32 op_rsp;
}mesh_cmd_sig_func_t;
```

- op: new-added command's opcode, no matter it is SIG command or vendor command, is expressed as u16, vendor command do not need to fill the vendor id bytes, library bottom layer will add automatically.
- status\_cmd: If the opcode is "status command" corresponding to certain "acknowledge request command", e.g. VD\_LIGHT\_ON/OFF\_STATUS, the "status\_cmd" should be set as 1; otherwise it should be set as 0. When model\_id\_rx is client model, "status\_cmd" should be set as 1. This status\_cmd flag is used in Library.
- model\_id\_tx: Corresponding model ID sending this command. E.g., when publish status, first, check mesh\_cmd\_vd\_func[] according to op to get model\_id\_tx, then get the corresponding global veriables, such as model\_sig\_g\_on/off\_level.on/off\_srv, then get the model\_sig\_g\_on/off\_level.on/ off\_srv->com. pub\_adr and its publish parameter, finally publish status.
- model\_id\_rx: Corresponding model ID receiving this command. If the node does not have corresponding model id in composition data, this opcode won't be processed.

When supports a specific model, the model parameters should be checked to determine if the model has bond corresponding app key, when the destination address is a group address, if the model has follow the corresponding group.

- When this command is received, callback processing function mesh\_rc\_data\_layer\_access\_cb()->p\_res->cb() is invoked, users can process their own app in this callback function.
- op\_rsp: If this opcode is "acknowledge request command", the "op\_rsp" should be set as corresponding ack command; otherwise it should be set as "STATUS\_NONE". The sending end will use this to determine if it has received the corresponding status response after it send the command.

## 7.2.2.4 Adding Command Callbacks

Adding a callback function for the VD\_GROUP\_G\_SET command

```
_int cb_vd_group_g_set(u8 *par, int par_len, mesh_cb_fun_par_t *cb_par)
 {
      int err = -1:
      int pub_flag = 0;
     //model_g_light_s_t *p_model = (model_g_light_s_t *)cb_par->model;
vd_group_g_set_t *p_set = (vd_group_g_set_t *)par;
      u8 sub_op = p_set->sub_op;
5
     if(!cb_par->retransaction){
          cb_vd_group_g_sub_set p_cb_set = (cb_vd_group_g_sub_set)search_vd_group_g_func(sub_op, SEARCH_VD_GROUP_G_FUNC_
-
          if(p_cb_set)
              pub_flag = p_cb_set(par, par_len, cb_par);
÷1.
          }else{
              return -1;
          }
if(par_len >= 3){
    if(par[2] && (par[2]<=128) && BIT_IS_POW2(par[2])){</pre>
              blt_rxfifo.num = par[2];
          3
 #endif
      if(VD_GROUP_G_SET_NOACK != cb_par->op){
-
          err = cb_vd_group_g_get(par, par_len, cb_par);
-
     }else{
          err = 0;
     }
     -if(!err && pub_flag){
          if(is_vd_onoff_op(sub_op)){ // only onoff need publish now
              model_pub_check_set(ST_G_LEVEL_SET_PUB_NOW, cb_par->model, 1);
          }
     }
      return err;
 }
```



## 7.2.2.5 Add TID Registration

In general, it is not necessary to add a TID; see this section for a description of the use of TIDs.: Example of Adding a Knowledge-command.

If the added command code requires a TID field (), it also needs to be additionally registered inside is\_cmd\_with\_tid\_vendor(), as detailed in the example section of Example of Adding a Knowledge-command.

## 7.2.3 Example of Adding a Knowledge-command

acknowledge-command, i.e. request command with status response.

Take VD\_GROUP\_G\_SET for example.

1) Add the content below in the "mesh\_cmd\_vd\_func[]":

(VD\_GROUP\_G\_SET, 0, VENDOR\_MD\_LIGHT\_C, VENDOR\_MD\_LIGHT\_S, cb\_vd\_group\_g\_set, VD\_GROUP\_G\_STATUS)

2) This command needs the "TID (Transmit ID)" field. Therefore, it's needed to add corresponding branch in the "is\_cmd\_with\_tid\_vendor()", and mark the location of the TID field in the access payload.

In the library, there will be a global variable mesh\_tid that manages TID uniformly. When a command is sent, it will be automatically increased by one and copied to the TID field of the command parameter area. Please refer to Vendor model format for detail.

TID purpose: If repeated TID is received within a specific duration (currently it's set as 6s by default), the corresponding action won't be implemented, but it will respond with response. This recognition action is implemented in the library, while the upper APP can directly judge the flag "cb\_par->re-transaction".

TID normally is for light status control commands. TID is to prevent acting repeatedly when receive retry in a specific duration, thus cause wrong delay time, for example, when the node receive OFF command, the corresponding delay time is 1s, and when the delay time passes for 0.8 s, receives retry command, if this command is executed, the delay time will re-count for 1 s, so the phenomena is that the node will be off after 1.8s. This will also cause light flash, e.g., 2 app control the same node at the same time, one act generic on, the other generic off, and both have retry action (message have different sequence numbers but same TID), for this case, what we expected is, light only execute 1 on and 1 off, the final status is determined by the last received command. This can be done if there is TID identification, without TID, the light may execute multiple on/off actions, thus causes flash.

In SIG standard commands, only light control command, such as on/off set, level set, lightness set, CT set, use TID.

Commands like lightness get, and config set/get in config model do not use TID.

Do not add vendor command if not necessary, do not waste any byte of the limited effective bytes.

- Edit the function "the cb\_vd\_group\_g\_set()", invoke the "light\_on/off\_idx()" to execute light on/off action.
- 4) Since this command is a command that requires an ack reply, write the corresponding ack function vd\_light\_tx\_cmd\_onoff\_st(), which calls mesh\_tx\_cmd(VD\_GROUP\_G\_STATUS,.....) for ack reply inside the function.

#### Note:

- To reply to status after receiving a command, you need to call mesh\_tx\_cmd\_rsp() to reply, not mesh\_tx\_cmd2normal(), because in a network with multiple network keys, app keys, when receiving a packet, the packet is decrypted with whatever key is used, and when replying to status, the packet must be encrypted with the corresponding key for encryption. So use mesh\_tx\_cmd\_rsp().
- mesh\_tx\_cmd2normal\_primary() use the first key by default to send, normally is used for send command, when reply status, do not use this function.

5) Assemble the interface "vd\_cmd\_on/off()" sending the "VD\_GROUP\_G\_SET" command.

"rsp\_max" indicates the number of nodes that need response.

- When the "adr\_dst" is unicast, the "rsp\_max" can be set as 1 (recommended) or 0.
- When the "adr\_dst" is group, the "rsp\_max" should be set as the number of elements owned by group according to the record in APP database.

## 7.2.4 Add Unacknowledged command

Take "VD\_GROUP\_G\_SET\_NOACK" as an example.

1) Add the content below in the "mesh\_cmd\_vd\_func[]":

(VD\_GROUP\_G\_SET\_NOACK, 0, VENDOR\_MD\_LIGHT\_C, VENDOR\_MD\_LIGHT\_S, cb\_vd\_group\_g\_set, STATUS\_NONE)

- 2) This command needs the "TID (Transmit ID)" field. Please refer to the method of adding acknowledge command.
- 3) Compile the function "cb\_vd\_group\_g\_set()" (shared with "VD\_GROUP\_G\_SET"). Please refer to the method of adding acknowledge command.
- 4) This command does not need ack response.
- 5) Assemble the interface "vd\_cmd\_on/off()" sending the "VD\_GROUP\_G\_SET\_NOACK" command.

Please refer to the method of adding acknowledge command.

## 7.2.5 Publish function registration

The vendor model's publish functionality is generally not needed.

If necessary, the publish parameter of the light node model is set by the CFG\_MODEL\_PUB\_SET command, the model has the publish function. When the model status changes, it will automatically publish a status message to the publish address configured by the publish parameter. In addition, the publish parameter can also be configured to send periodically, please check publish command parameter definition spec in [4.3.2.16 Config Model Publication Set] for detail.

In order to implement the above automatic publish function, you need to register the publish function for the model to send status messages, as shown below:

# void mesh\_model\_cb\_pub\_st\_register() { ..... MODEL\_PUB\_ST\_CB\_INIT(model\_vd\_light.srv, &vd\_light\_onoff\_st\_publish); ..... }

#### Figure 7.2: publish function

When the status changes, or after the publish cycle time expires, the mesh stack will call back this function and send a status message.

## 7.3 Add the Vendor Opcode Subcommand

"vendor opcode introduction" This section describes how to use Vendor subcommands. Customers can choose whether to use the subcommands or not according to their needs. The following describes how to use the vendor opcode subcommand.

## 7.3.1 Vendor Subcommand Range

The sub opcode of the Vendor sub-command occupies one byte, and there are only 256 values in total. 0x00 to 0x7f is reserved for Telink, and the range for user is 0x80 to 0xff, as detailed in the comment of sdk's vd\_group\_g\_func[].

## 7.3.2 Steps of Adding Vendor Subcommand

In order to facilitate the user to quickly add the vendor subcommand, the user can directly use the demo bracketed by VENDOR\_SUB\_OP\_USER\_DEMO\_EN without adding a new vendor subcommand, the user can directly use this subcommand and only need to change the corresponding callback functions vd\_rx\_group\_g\_sub\_op\_user\_demo\_set(), vd\_rx\_group\_g\_sub\_op\_user\_demo\_st() to quickly use the subcommand. Also, user added subcommands can refer to VENDOR\_SUB\_OP\_USER\_DEMO\_EN bracketed demo to add more subcommands.

```
#if VENDOR_SUB_OP_USER_DEMO_EN
    {VD_GROUP_G_SUB_OP_USER_DEMO, vd_rx_group_g_sub_op_user_demo_set,
    vd_rx_group_g_sub_op_user_demo_st}
#endif
```

In addition, if you need to store data, you can call mesh\_common\_store() in vd\_rx\_group\_g\_sub\_op\_user\_demo\_set().

mesh\_common\_store(FLASH\_ADR\_MD\_VD\_LIGHT);

and then the parameter sno\_vd\_sno\_sub\_op\_user\_demouser\_demo added to model\_vd\_light\_t is stored in this sector of flash FLASH\_ADR\_MD\_VD\_LIGHT. after re-powering up, the sdk has already implemented the ability to retrieve the contents of this sector to the corresponding global variable through mesh\_flash\_ retrieve() to read the contents of this sector into the corresponding global variable. Note that the structure and size of model\_vd\_light\_t must not change before or after OTA, otherwise the data will be read abnormally after OTA.

The following describes an example of adding subcommands to the VD\_GROUP\_G\_SET / VD\_GROUP\_G\_GET / VD\_GROUP\_G\_STATUS group of commands.

## 7.3.2.1 Add the Definition of the Vendor Subcommand

The demo SDK defines vendor's on and off as two separate commands, mainly for compatibility with old version. If you want to add this kind of command, you only need to add a sub-command, and then use another byte in the parameter area to indicate on or off. So the following is about VD\_GROUP\_G\_ON only.

```
enum{/*vendor generic group, op code include C1-C4*/
.....
VD_GROUP_G_ON = 1, // compatible with legacy
.....
};
```

## 7.3.2.2 Add Registration of the Vendor Subcommand

Vendor subcommand registration reference code vendor\_model.c vd\_group\_g\_func\_t vd\_group\_g\_func = {.....} implementation.

```
vd_group_g_func_t vd_group_g_func[] = {
    /* telink use sub op from 0x00 to 0x7f*/
    ......
    {VD_GROUP_G_ON, vd_group_g_light_onoff, vd_light_tx_cmd_onoff_st},
    ......
};
```

#### Note:

vd\_group\_g\_func is a const type array, so it is read-only and cannot be rewritten. This saves RAM space. In addition, if the added command requires a TID field, you need to register it in is\_cmd\_with\_tid\_vendor(), please refer to the "Adding an acknowledge-command" section for more details.

## 7.3.2.3 vd\_group\_g\_func\_t Introduction

```
typedef struct{
    u32 sub_op;
    cb_vd_group_g_sub_set cb_set;
    cb_vd_group_g_sub_tx_st cb_tx_st;
    //cb_vd_group_g_sub_rx_status cb_rx_status; // TBD, only client may use.
}vd_group_g_func_t;
```

- sub\_op: subcommand.
- cb\_set: Sets the handler function to be invoked when the command is received.
- cb\_tx\_st: Set response function with sending status.
- cb\_vd\_group\_g\_sub\_rx\_status: The gateway device receives the processing function of the STATUS command for this subcommand, which is currently not enabled.

#### 7.3.2.4 Adding Subcommands Callback Functions

Adding a callback function for the VD\_GROUP\_G\_ON subcommand

```
int vd_group_g_light_onoff(u8 *par, int par_len, mesh_cb_fun_par_t *cb_par)
{
    int pub_flag = 0;
    vd_light_onoff_set_t *p_set = (vd_light_onoff_set_t *)par;
    int light_idx = cb_par->model_idx;
    int on = !!p_set->sub_op; // make sure bool
    light_onoff_all(on);
    if(vd_onoff_state[light_idx] != on){
        vd_onoff_state[light_idx] = on;
        pub_flag = 1;
    }else{
    }
    return pub_flag;
}
```

## 7.3.3 Adding Acknowledge Type Subcommand

The acknowledge-command is a request command with a status response, indicating that the command requires a status response.

Take VD\_GROUP\_G\_ON as an example.

Two conditions need to be met for a subcommand to reply: the larger command needs to be VD\_GROUP\_G\_SET or VD\_GROUP\_G\_GET, and the member variable cb\_tx\_st inside vd\_group\_g\_func[] is not NULL.

(1) Add vd\_light\_tx\_cmd\_onoff\_st in vd\_group\_g\_func[] at the corresponding location.

```
vd_group_g_func_t vd_group_g_func[] = {
    /* telink use sub op from 0x00 to 0x7f*/
    .....
    {VD_GROUP_G_ON, vd_group_g_light_onoff, vd_light_tx_cmd_onoff_st},
    .....
};
```

(2) Write vd\_light\_tx\_cmd\_onoff\_st function

## 7.3.4 Add Subcommands of Type Unacknowledge

Take VD\_GROUP\_G\_ON as an example.

Subcommands do not require a reply, one of the following conditions is satisfied: the larger command uses VD\_GROUP\_G\_SET\_NOACK or the member variable cb\_tx\_st inside vd\_group\_g\_func[] is NULL.

## 7.3.5 Write API for Sending VD\_GROUP\_G\_ON Command

vd\_cmd\_onoff()

### 7.3.6 Example of Adding an Empty Vendor Subcommand

The empty vendor subcommand example is an example where the callback function is empty. So to add the vendor subcommand empty example the user only needs to perform two steps.

(1) Adding the vendor subcommand definition

```
enum{/*vendor generic group, op code include C1-C4*/
.....
VD_GROUP_LOOP_ON = 1, // compatible with legacy
.....
};
```

(2) Adding registration of vendor subcommands

Vendor subcommand registration reference code vendor\_model.c vd\_group\_g\_func\_t vd\_group\_g\_func = {.....} implementation.

```
vd_group_g_func_t vd_group_g_func[] = {
    /* telink use sub op from 0x00 to 0x7f*/
    ......
    {VD_GROUP_LOOP_ON, NULL, NULL},
    ......
};
```



# 8 Global Configuration File Introduction

# 8.1 mesh\_config.h

#### PROXY\_HCI\_SEL:

For debugging, developer can choose PROXY\_HCI\_GATT by default.

#### DEBUG\_VENDOR\_CMD\_EN:

Enable/disable vendor model debug command. Enabled by default.

#### FAST\_PROVISION\_ENABLE:

This is a private mode, can provision with multiple nodes at the same time, supports group action and relay network. Disabled by default.

#### MESH\_USER\_DEFINE\_MODE:

Define authentication mode during provision, MESH\_NORMAL\_MODE: no OOB mode; others are static OOB mode, please refer to Connect with a platform.

#### SUBSCRIPTION\_BOUND\_STATE\_SHARE\_EN:

The purpose is to add the group number to the models listed in sub\_share\_model\_sig[] and sub\_share\_model\_vendor[] automatically after receiving the command to set the group number for onoff model. This is because the group number information is shared between models with state binding, such as Onoff model and lightness model.

Additionally, in private mode, models that do not have a state binding relationship can be configured to share group number information with each other by adding the corresponding group number to sub\_share\_model\_sig[] and sub\_share\_model\_vendor[].

#### PROVISION\_FLOW\_SIMPLE\_EN:

Same as the standardized provision, provision nodes one by one, i.e., only one node is configuring network at the same time. When node receives app key add, automatically binds key to every model. Provisioner does not need to send key bind command. Simplify provision process and reducing provision time.

#### AIS\_ENABLE / MI\_API\_ENABLE:

Please refer to Connect with a platform.

#### LIGHT\_TYPE\_SEL:

Select light type. Currently supported light types are mutually exclusive. Please refer to section 1.3 LIGHT\_TYPE\_SEL Introduction.

The following are model on/off control macro, e.g., MD\_LIGHTNESS\_EN, when it is enables, whether it enables client or server model, or both, is determined by MD\_SERVER\_EN, MD\_CLIENT\_EN and MD\_CLIENT\_VENDOR\_EN. Check introductions of these 3 macros below.

#### LIGHT\_TYPE\_CT\_EN:

Enable / Disable CT light related model, includes Light CTL Server, Light CTL Setup Server, Light CTL Temperature Server, Light CTL Client.

#### LIGHT\_TYPE\_HSL\_EN:



Enable / Disable HSL light related model, includes Light HSL Server, Light HSL Hue Server, Light HSL Saturation Server, Light HSL Setup Server, Light HSL Client.

#### MD\_LIGHT\_CONTROL\_EN:

Enable / Disable (Default) Lighting Control related model on/off control, includes Light LC Server, Light LC Setup Server, Light LC Client.

#### MD\_LIGHTNESS\_EN:

Enable / Disable Lightness related model, includes Light Lightness Server, Light Lightness Setup Server, Light Lightness Client.

#### MD\_LEVEL\_EN:

Enable (Default) / Disable Generic Level Model. Each status can have a corresponding level model.

#### MD\_MESH\_OTA\_EN:

Enable / Disable (Default) Mesh\_OTA\_Model interface.

#### MD\_ONOFF\_EN:

Enable (Default) / Disable Generic On/off Model.

#### MD\_DEF\_TRANSIT\_TIME\_EN:

Enable (Default) / Disable Generic Default Transition Time Model.

#### MD\_POWER\_ONOFF\_EN:

Enable (Default) / Disable Generic Power On/off Model. Enable / Disable at the same time with MD\_DEF\_TRANSIT\_TIME\_EN, because the parameters of these 2 models are save in the same flash sector.

#### MD\_TIME\_EN:

Disable (Default) / Enable Time Model.

#### MD\_SCENE\_EN:

Disable (Default) / Enable Scene Model.

#### MD\_SCHEDULE\_EN:

Disable (Default) / Enable Schedule Model. Disable / Enable at the same time with MD\_TIME\_EN, because schedule depends on time.

#### MD\_PROPERTY\_EN:

Enable / Disable Property model, includes Generic User Property Server, Generic Admin Property Server, Generic Manufacturer Property Server, Generic Client Property Server, Generic Property Client.

#### MD\_LOCATION\_EN:

Enable/Disable Location model, includes Generic Location Server, Generic Location Setup Server, Generic Location Client.

#### MD\_SENSOR\_EN:

Enable/Disable Sensor model, includes Sensor Server, Sensor Setup Server, Sensor Client.

#### MD\_BATTERY\_EN:

AN-17120400-E7



Enable/Disable Battery model, includes Generic Battery Server, Generic Battery Client

#### MD\_SERVER\_EN:

Enable the SIG and vendor models of the enabled server models when the value is set to 1, e.g., lightness server and VENDOR\_MD\_LIGHT\_S.

#### MD\_REMOTE\_PROV:

Enable/Disable Remote provision model. Default disable.

#### MD\_CLIENT\_EN:

Enable the client SIG model when the value is set to 1, e.g. lightness client.

#### MD\_CLIENT\_VENDOR\_EN:

Enable client vendor model: VENDOR\_MD\_LIGHT\_C when the value is set to 1.

#### MD\_VENDOR\_2ND\_EN:

Enable the second vendor server model VENDOR\_MD\_LIGHT\_S2 when the value is set to 1. Normally vendor model needs only 1.

#### Note:

Generally nodes do not need Client Model, so Client Model is disabled for light side by default so as to save RAM. Client model is controlled by MD\_CLIENT\_EN and MD\_CLIENT\_VENDOR\_EN, some light node need to enable vendor client model but not SIG client model.

#### FACTORY\_TEST\_MODE\_ENABLE:

Enable (Default) / Disable factory test mode. For the convenience of factory test, in the case of no provision, default key can be used to implement simple operations such as turning on/off node, adjusting luminance.

#### MANUAL\_FACTORY\_RESET\_TX\_STATUS\_EN:

Set whether to send NODE\_RESET\_STATUS to notify gateway or app after 5 times of booting/reset.

#### KEEP\_ONOFF\_STATE\_AFTER\_OTA:

Set whether to keep the on/off status of the light before reset.

#### ELE\_CNT\_EVERY\_LIGHT:

Element no. of each light. E.g., a CT light need 2 elements, most model will put it in the first element, only Light CTL Temperature Server and corresponding level model are in the second element.

Since lightness and CTL Temperature can both be controlled by level model commands, if there is only 1 element address, when receiving level set command, it is impossible to determine whether it is to control lightness or Temperature.

Note the difference between LIGHT\_CNT and ELE\_CNT.

LIGHT\_CNT is how many same lights in the BLE module, e.g., 2 CT lights.

ELE\_CNT = ELE\_CNT\_EVERY\_LIGHT \* LIGHT\_CNT is how many elements in this node. It is also the element address no. when provision.

#### FEATURE\_FRIEND\_EN:

Set whether to support Friend Feature



#### FEATURE\_LOWPOWER\_EN:

Set whether to support Low Power Feature.

#### FEATURE\_PROV\_EN:

Provision switch, need to enable.

#### FEATURE\_RELAY\_EN:

Set whether to support Relay Feature.

#### FEATURE\_PROXY\_EN:

Set whether to support Proxy Feature.

#### MAX\_LPN\_NUM:

Set the number of low power nodes supported by one friend node. Currently it's set as 2, it is recommended to limit this value less than 10(the maximum verified number) if the user need to modify this number. Too big value will cause higher possibility of packets conflict when friend reply respond to multiple LPN, and thus cause time delay and higher power consumption of LPN node, the RAM consuming will also increase.

#### USER\_DEFINE\_SET\_CCC\_ENABLE:

Must enable. Set whether App controlled node report notify/indication.

#### SEND\_STATUS\_WHEN\_POWER\_ON:

Set whether to send luminance state packet when power on, default sending address is Oxffff.

## 8.2 mesh\_node.h

#### SUB\_LIST\_MAX:

The maximum number of subscribed addresses (i.e., group number) supported by each model.Versions before V3.3.0 (not included) cannot be modified by the user, because the macro is used in the library, and can be modified later than that version. When the modified number is greater than 8, in order to save RAM and flash parameter storage area, the default subscription of virtual address is turned off, i.e., VIRTUAL\_ADDR\_ENABLE is equal to 0. Generally speaking, you cannot use the virtual address, if you need to turn it on, just set VIRTUAL\_ADDR\_ENABLE to 1.

#### BIND\_KEY\_MAX:

Maximum supporting bind key number, cannot be modified, because this macro is used in library.

#### SCENE\_CNT\_MAX:

Maximum configurable scene number, can be modified.

## 8.3 app\_mesh.h

#### 8.3.1 Macro introduction

#### TRANSMIT\_CNT\_DEF:



Set message default transmit cnt, i.e., retry time of each command

Retry time = TRANSMIT\_CNT\_DEF + 1.

#### TRANSMIT\_INVL\_STEPS\_DEF:

Set message default transmit interval, i.e., retry interval.

Retry interval = (TRANSMIT\_INVL\_STEPS\_DEF + 1) \* 10ms + (0-10) ms.

#### TRANSMIT\_CNT\_LPN\_ACCESS\_CMD:

For LPN node, control commands' transmit cnt is defined by TRANSMIT\_CNT\_DEF, e.g., friend request, friend poll, other message is defined by TRANSMIT\_CNT\_LPN\_ACCESS\_CMD, e.g., on/off status.

#### TRANSMIT\_CNT\_DEF\_RELAY, TRANSMIT\_INVL\_STEPS\_DEF\_RELAY:

Relay's transmit count and transmit interval.

#### MESH\_ADV\_CMD\_BUF\_CNT:

Set message transmitting buffer size, excludes relay message.

#### MESH\_ADV\_BUF\_RELAY\_CNT:

Set relay message of relay message.

#### SEC\_NW\_BC\_INV\_DEF\_100MS:

Set security beacon's transmitting interval when provision, unit is 100ms.

#### **8.3.2 Function introduction**

mesh\_tx\_cmd(material\_tx\_cmd\_t \*p)

This is a common function to send command.

1) Parameters:

```
type typedef struct{
union{ //point to parameter address
u8 *par;
u8 *p_ac;
};
union{ //parameter length
u32 par_len;
u32 len_ac;
};
u16 adr_src; //source address
u16 adr_dst;//destination address
u8* uuid; //point to virtual address
model_common_t *pub_md; // point to model parameter
u32 rsp_max; //number of nodes that need response
u16 op; // command code
u16 nk_array_idx; // network_key index
```



u16 ak\_array\_idx; // app\_key index u8 retry\_cnt; // number of retry times }material\_tx\_cmd\_t;

Parameter values are determined by the parameters of the invoking function "mesh\_tx\_cmd2normal\_primary(u16 op, u8 \*par, u32 par\_len, u16 adr\_dst, int rsp\_max)".

2) Return value

If the return value is 0, it indicates successful command execution.

If the return value is not zero, it indicates transmission failure, e.g. currently there's a command being sent, new command cannot be accepted (busy state), certain parameter is illegal, and etc.

int mesh\_tx\_cmd\_primary(u16 op, u8 \*par, u32 par\_len, u16 adr\_dst, int rsp\_max)

This function serves to fix "adr\_src" as "ele\_adr\_primary", and then assemble "mesh\_tx\_cmd()".

## 8.4 app\_provision.c

#### u8 is\_provision\_success():

Get the status if the node is provisioned successfully.

#### u8 is\_provision\_working():

Get the status if the node is in provision process.

## 8.5 mesh\_node.c

is\_own\_ele():

Determine if node's adr is the element address of its own.

## 8.6 mesh\_common.c file introduction

#### HCI fifo:

hci\_tx\_fifo and hci\_rx\_fifo are fifos to define and transmit data by peripherals, e.g., gateway nodes and gateway firmware USB communication.

#### mesh\_get\_proxy\_hci\_type():

Define proxy type, PROXY\_HCI\_GATT by default. PROXY\_HCI\_USB is debug mode, not open to user.

#### mesh\_tid\_save():

Function to save TID. E.g. Commands such as generic on/off need to use tid. If deep sleep mode is not executed, it's not needed to save the tid (just initialize it as 0 after power on). If deep mode is executed, e.g. switch, each key press will initialize all variables including tid, in this case, the tid should be saved.



#### adv\_filter\_proc():

IRQ RX will filter the received verified correct packets with this function, e.g., abandon connectable packet. See code for detail.

#### const u16 sub\_share\_model[]:

```
const u16 sub_share_mode= {
   SIG_MD_G_ONOFF_S, SIG_MD_G_LEVEL_S, SIG_MD_LIGHTNESS_S, SIG_MD_LIGHTNESS_SETUP_S,
   SIG_MD_LIGHT_CTL_S, SIG_MD_LIGHT_CTL_SETUP_S, SIG_MD_LIGHT_CTL_TEMP_S,
   SIG_MD_LIGHT_HSL_S, SIG_MD_LIGHT_HSL_SETUP_S, SIG_MD_LIGHT_HSL_HUE_S,
   SIG_MD_LIGHT_HSL_SAT_S,
   SIG_MD_SCENE_S, };
```

Refer to SUBSCRIPTION\_SHARE\_EN introduction in mesh\_config.h.

These Models are bonded by default, i.e. when setting subscribing address (assign group), these models will take effect at the same time.

#### How to call:

Receive CFG\_MODEL\_SUB\_ADD -> mesh\_rc\_data\_layer\_access\_cb() -> mesh\_cmd\_sig\_cfg\_model\_sub\_set()
-> share\_model\_sub\_by\_rx\_cmd() -> share\_model\_sub()。

#### entry\_ota\_mode():

The SDK will callback this function after OTA start command is received.

#### ota\_condition\_enable():

Condition to allow GATT OTA. When GATT connection is successful, and "set proxy filter" is received, "pair\_login\_ok" will be set as 1. (Note: set proxy filter need to be encrypted/decrypted with network key when receiving/transmitting.)

#### proc\_telink\_mesh\_to\_sig\_mesh():

It serves to detect whether firmware type before OTA is SIG mesh or other SDK, e.g. Telink mesh. If it is not SIG mesh, product switch and parameter initialization will be executed.

#### mesh\_ota\_reboot\_proc():

After mesh OTA is finished, delay for 1.5s and then reboot.

How to call: main\_loop() -> mesh\_loop\_process() -> mesh\_ota\_reboot\_proc()

#### mesh\_ble\_connect\_cb:

Callback this function when GATT connect successfully.

#### mesh\_ble\_disconnect\_cb:

Callback this function after GATT disconnect.

#### update\_para\_change\_MTU():

It serves to request for BLE connection parameter update as needed, and prevent starting parameter update during discovery and provision.

#### gatt\_adv\_prepare\_handler():



1) relay\_adv\_prepare\_handler()

Relay buffer is independent, use different fifo with TX command, relay buffer can transmit packet during TX command transmit interval.

Priority: TX command -> relay -> connectable packet.

2) Others are GATT packets.

#### app\_advertise\_prepare\_handler ():

When BLE stack bottom layer allows to send adv packet, it will callback this function. If there's adv packet (including connectable adv packet, beacon packet) to be sent in current task, it's only needed to set the parameter "p" as the pointer of the structure.

Priority: message Friend Node send to LPN after it receive LPN poll > TX command > relay > connectable adv packet.

1) get\_adv\_cmd():

The return value is pointer of mesh message packet to be sent. If it's non-zero value, it indicates there's packet to be sent, including MESH\_ADV\_TYPE\_MESSAGE, MESH\_ADV\_TYPE\_BEACON of SECURE\_BEACON type.

```
2) mesh_adv_cmd_set()
```

Copy packet to be sent to BLE stack.

```
3) p_bear -> trans_par_val:
```

It includes transmit count and transmit interval.

```
4) mesh_rsp_random_delay_step
```

When the node receive a group address as destination address, it need to add Random delay for response. Check mesh\_rc\_data\_layer\_access\_cb() for detail.

5) adv\_retry\_flag

Serves for cancelling network transmit interval, continuous transmission and etc., e.g., poll sent by LPN after build friendship.

#### app\_l2cap\_packet\_receive ():

When BLE stack receives packet with payload, it will callback this function, and then invoke the function "blc\_l2cap\_packet\_receive()" to analyze the data. During debugging, developer can print out the data for the convenience of analysis.

#### chn\_conn\_update\_dispatch():

Negligible currently.

#### sim\_tx\_cmd\_node2node():

It serves to send unreliable light ON/OFF command with the interval of three seconds for demo demonstration.

#### usb\_id\_init():



It's used to configure USB ID. When multiple dongles are connected with one PC simultaneously, they must be configured with different IDs so as to be recognized as different devices by PC.

#### ble\_mac\_init():

When parameter location of MAC is illegal value, it will randomly generate a MAC and save it.

#### mesh\_scan\_rsp\_init():

It serves to fill in the fixed field of "scan rsp" during initialization, e.g. mac address. Mac needs to be used when building database, and iOS system cannot directly obtain it from the AdvA field of adv packet data. Therefore, it should be marked in the content of scan response.

#### mesh\_scan\_rsp\_update\_adr\_primary():

It serves to fill in the fixed field of "scan rsp" during initialization, e.g. mac address. Mac needs to be used when building database, and iOS system cannot directly obtain it from the AdvA field of adv packet data. Therefore, it should be marked in the content of scan response.

#### publish\_when\_powerup():

Boot, send corresponding status after a random interval (publish\_powerup\_random\_ms), notify app or gateway note to get online.

#### mesh\_vd\_init():

Common processing part related to mesh of multiple projects. It's invoked in "mesh\_init\_all()".

#### mesh\_global\_var\_init():

Initialization function of global structure variable, executed before reading related parameters stored in flash. It serves to set default value of compiling, and if there are related parameters in flash, the values in flash will be used.

#### model\_sig\_cfg\_s\_cps:

I.e. composition data. For related definitions, please refer to the structure definition of model\_sig\_cfg\_s\_cps and spec.





#### set\_unprov\_beacon\_para():

- p\_uuid: pointer to set uuid. The length is 16 bytes.
- p\_info: pointer to set oob\_info. The length is 2 bytes.
- p\_hash: pointer to set hash value of URI. The length is 4 bytes.
- uri\_para: pointer of uri connection. The maximum length is 40 bytes.
- uri\_len: Length of actually used data in uri part. The maximum length does not exceed 40.

The parameters above serve to configure beacon packet of unprovisioned node.

#### set\_provision\_adv\_data():

- p\_uuid: pointer to set uuid. The length is 16 bytes.
- oob\_info: pointer to set oob\_info. The length is 2 bytes.

The parameters above serve to configure parameters of adv packet part when provision is not finished.

#### set\_proxy\_adv\_pkt():

- p\_hash: pointer to set hash. The length is 8 bytes.
- p\_random: pointer to set random. The length is 8 bytes.
- node\_identity: It indicates adv packet type.

If "node\_identity" is 0, it indicates adv type is "advertising with Network ID", in this case "p\_hash" and "p\_random" won't take effect.

If "node\_identity" is 1, it indicates adv type is "advertising with Node Identity", in this case "p\_hash" and "p\_random" will take effect.

The parameters above serve to configure adv packet to send proxy connection after provision is finished.

#### uart\_drv\_init()/usb\_bulk\_drv\_init():

Serial port and USB initialization, select serial port or USB via the macro "HCI\_ACCCESS". Use "blc\_register\_hci\_handler" to register callback function. User can invoke "my\_fifo\_push\_hci\_tx\_fifo" to push data to be reported into "hci\_tx\_fifo".

#### set\_material\_tx\_cmd():

Set transmission parameter:

- 1) op: vendor op code, input 1 byte, no need to input vendor id, it will be fulfilled automatically.
- 2) rsp\_max: only effect to status replied command, serves to detect whether receives enough status. When destination address is unicast, the value is 0 or 1(0 and 1 are the same, the detection is done when receiving the status), when destination address is group, the value is the node number of the group.
- 3) retry\_cnt: only effect to status replied command, when the command is sent for a while, and no specified rsp\_max status is received, then trigger retry flow, the retry time is determined by retry\_cnt.
- 4) uuid: when the sending destination address is virtual address, need to input uuid, otherwise it is 0.
- 5) nk\_array\_idx: mesh supports multiple netkey, this is to set the array index no. in netkey array, note, it is not the provision global netkey index.



- 6) ak\_array\_idx: mesh supports multiple appkey, this is to set the array index no. in appkey array, note, it is not the provision global appkey index.
- 7) pub\_md: when execute publish status, it need to be set as pointer point to model, because when sending massage, mesh\_model\_pub\_par\_t parameter in pub\_md-> pub\_par need to be used, e.g., ttl, network transmit, network count, appkey index, but not use default values. Set pub\_md to 0 when not execute publish status.

#### mesh\_tx\_cmd2normal\_primary():

Node actively send command API, see set\_material\_tx\_cmd() for parameter detail.

#### SendOpParaDebug\_vendor():

In WIN32 mode, analysis of gateway, app, par\_tmp[2:3], see ini format analysis.

#### is\_need\_response\_to\_self():

Set if need reply status when receive command sending by the function itself and need to answer with status.

#### mesh\_rc\_data\_layer\_access\_cb():

When node receives a command sent to itself (condition: model supports, destination address matches) will call this function.

- 1) Vendor Op code range in VD\_OP\_RESERVE\_FOR\_TELINK\_START and VD\_OP\_RESERVE\_FOR\_TELINK\_END is opcode reserved for Telink, not open to users.
- 2) mesh\_need\_random\_delay : when node receives group address as destination address, need to add a Random delay to avoid multiple nodes respond at the same time when response.
- 3) p\_res->cb: this is the corresponding callback function for each op code, mesh\_cmd\_sig\_func[]->cb and mesh\_cmd\_vd\_func[]->cb.

#### mesh\_rsp\_handle\_cb():

Reports status status message gateway received to firmware, via USB or UART.

#### hci\_send\_data\_user():

Buffer data of hci tx fifo, the first 2 byte is len, the third is data type to tell data type. Here is HCI\_RSP\_USER, other please refer to hci\_type\_t.

#### mesh\_tx\_reliable\_stop\_report():

Callback function when gateway send reliable command, and the stop condition is fulfilled.

#### app\_hci\_cmd\_from\_usb():

In blt\_sdk\_main\_loop() function, callback app\_hci\_cmd\_from\_usb() to handle commands sent by firmware, analyze and execute the commands via app\_hci\_cmd\_from\_usb\_handle().

#### app\_hci\_cmd\_from\_usb\_handle ():

The corresponding data is in ini format, check ini chapter for detail.

# 8.7 cmd\_interface.h file introduction

#### access\_cmd\_get\_level():

This function serves to obtain level value of element by setting "opcode" as "G\_LEVEL\_GET" and then assembling "mesh\_tx\_cmd()".

#### access\_cmd\_set\_level():

This function serves to set level value of element by assembling "mesh\_tx\_cmd()". When ack is 1, set opcode as "G\_LEVEL\_GET". When ack is 0, set opcode as "G\_LEVEL\_SET\_NOACK". The SDK will manage and implement tid parameters used in this command, so these parameters are negligible for upper development.

#### access\_set\_lum():

This function serves to set level value by inputting "lum" (range is  $0\sim100$ ) and assembling "access\_cmd\_set\_level ()".

#### access\_cmd\_onoff():

This function serves to set on/off value of element by assembling "mesh\_tx\_cmd()". When ack is 1, set opcode as "G\_ON/OFF\_GET". When ack is 0, set opcode as "G\_ON/OFF\_SET\_NOACK". The SDK will manage and implement tid parameters used in this command, so these parameters are negligible for upper development.

## 8.8 vendor\_model.c file introduction

This file mainly introduces transmission of opcode corresponding to vendor model, as well as corresponding callback function to be executed after this opcode is received.

Note: non-provisioner nodes use only 1 vendor id, and will show this id in composition data, Vendor model has 64 op code in total. Please be noted, this is not 64 for a product, it's 64 for all products. So please use it wisely, use as many sub-commands as possible.

Telink also uses some vendor op code for self-define features, so currently 0xCO-0xDF is reserved for Telink, and 0xEO-0xFF is for other users.

#### Register of vendor opcode

See Chapter 3.2.

#### mesh\_search\_model\_id\_by\_op\_vendor():

This function serves to search for related resources in the array "mesh\_cmd\_vd\_func[]" via opcode. User does not need to modify this function.

#### vd\_cmd\_key\_report():

This function serves to report key press event and it's used in the project "8258\_mesh\_switch".

It can be considered that "int SendOpParaDebug(u16 adr\_dst, u8 rsp\_max, u16 op, u8 \*par, int len);" is equivalent to "mesh\_tx\_cmd\_primary()".

#### is\_cmd\_with\_tid\_vendor():

This function serves to check whether this opcode needs to carry tid and return value accordingly. If this opcode needs to carry tid, it will return 1, and return the location of tid in parameter area via "tid\_pos\_out"; otherwise, it will return 0.


## 8.9 mesh\_test\_cmd.c file introduction

This file serves to save implementation of test commands.

# 9 8258 MESH Project Introduction

## 9.1 app\_config\_8258.h

#### PCBA\_8258\_SEL:

Select PCBA, default is 48pin dongle board, the other 2 are reference board.

#### FLASH\_1M\_ENABLE:

Enable this macro when internal flash is 1M because the flash map is different, and for the initial configuration, MAC is 0Xff000 while 512K MAC is 0x76000.

#### HCI\_ACCESS:

Set HCl interface.

- HCI\_USE\_USB: use USB
- HCI\_USE\_UART: use UART

HCl is not needed for data transmission by default.

#### UART\_GPIO\_SEL:

Set UART IO.

#### HCI\_LOG\_FW\_EN:

Firmware disables this function by default, user can enable this if needed, refer to log output chapter for detail.

#### ADC\_ENABLE:

Set whether to enable ADC or not.

#### ONLINE\_STATUS\_EN:

Private mesh SDK online status function. Send real time status data, optimize real-time function of publish.

#### DUAL\_MODE\_ADAPT\_EN:

SIG mesh + ZigBee dual modes.

#### DUAL\_MODE\_WITH\_TLK\_MESH\_EN:

Enable SIG mesh + private mesh SDK dual modes.

#### TRANSITION\_TIME\_DEFAULT\_VAL:

Default transition time is used when power on, and receiving a command supporting transition variable, e.g., generic on/off, but there is no transition variable in this command, the light will act according to TRANSITION\_TIME\_DEFAULT\_VAL. The default transition time is 1s, to disable transition, set TRANSI-TION\_TIME\_DEFAULT\_VAL to 0. Refer to trans\_time\_t for detail.

#### SW1\_GPIO/SW2\_GPIO:

Two buttons on dongle board, used for debugging, disabled by default.

To enable, first verify IO, then modify corresponding PULL\_WAKEUP\_SRC\_XXX and XXX\_INPUT\_ENABLE.

#### PWM\_R/ PWM\_G/ PWM\_B/ PWM\_W:

Set IO corresponding to PWM.

#### GPIO\_LED:

Set led light IO, e.g., when the provision is completed, reset to factory configuration, the definition of light flashing.

## 9.2 app.c file introduction

### 9.2.1 Customization of Adv packet and Adv response packet

#### Advertising packet

Connectable adv packet: Currently SIG MESH spec has already defined all fields for connectable adv packet format. Please refer to the structure "PB\_GATT\_ADV\_DAT" or spec for details.

#### Advertising response packet

User can customize adv response packet by modifying the array "u8 tbl\_scanRsp [] = {}" via "mesh\_scan\_rsp\_init()". The maximum length of adv response packet can reach 31 bytes, only a part of which is used currently. User can configure the "rsv" field as needed in "mesh\_scan\_rsp\_init()".

```
typedef struct{
    u8 len;
    u8 type;
    u8 mac_adr[6];
    u16 adr_primary;
    u8 rsv_telink [10]; // not for user
    u8 rsv_user[11];
}mesh_scan_rsp_t;
```

## 9.2.2 Configuration of FIFO part

```
MYFIF0_INIT(blt_rxfifo, 64, 16);
MYFIF0_INIT(blt_txfifo, 40, 32);
```

The two functions serve to configure packet Rx buffer and Tx buffer in BLE stack bottom layer.Generally it's not recommended to modify them unless RAM size is not large enough.

### 9.2.3 app\_event\_handler ()

Callback processing function: When BLE stack receives adv (include connectable adv packet, beacon packet, etc), connect request packet, BLE connection parameter update packet, BLE connection termination, and etc, this callback function will be invoked after the event to process correspondingly.



Currently all beacons used for SIG MESH communication are processed in the branch "(subcode == HCI\_SUB\_EVT\_LE\_ADVERTISING\_REPORT)".

For processing of other events, please refer to corresponding code. Currently only simple LED indicating light processing is contained, and related functions can be added if needed.

#### HCI\_SUB\_EVT\_LE\_ADVERTISING\_REPORT:

Processing branch after receiving adv packet. User can add corresponding event and processing function under this branch.

#### HCI\_SUB\_EVT\_LE\_CONNECTION\_COMPLETE:

Event callback generated after BT connection is established. BLE stack bottom layer will callback to this branch after BT connection is established.

#### HCI\_CMD\_DISCONNECTION\_COMPLETE:

After BT connection is terminated, BLE stack bottom layer will callback to this branch.

## 9.2.4 main\_loop ()

- mesh\_loop\_proc\_prior(): function with high priority for real time features
- blt\_SDK\_main\_loop (): main\_loop function of BLE stack.
- proc\_led(): LED indicating light event processing function.
- factory\_reset\_cnt\_check(): factory reset processing function. Support reset method of five power on operations. Please refer to factory reset section.
- mesh\_loop\_process(): SIG mesh related loop function, including retry mechanism of reliable command, segment ack timeout response, TID timeout detect mechanism, and etc.
- sim\_tx\_cmd\_node2node(): Demo demonstration interface of ON/OFF command timed transmission.

## 9.2.5 user\_init()

- proc\_telink\_mesh\_to\_sig\_mesh(): To implement OTA between sig mesh and telink mesh with incompatible parameter format, it's needed to initialize parameters. In current SDK, when mesh type change is detected, parameter area to be used by new mesh will be cleared.
- bls\_ll\_setAdvParam(): Define parameters including adv packet interval, currently not recommended to modify.
- blc\_ll\_setAdvCustomedChannel(): Customize adv channel. Sig mesh requires to use standard channel 37/38/39. However, during test process, for the convenience of debugging, the channel can be changed.
- bls\_ll\_setAdvEnable(1): Enable transmission of adv packet.
- rf\_set\_power\_level\_index (MY\_RF\_POWER\_INDEX): The default setting of transmit power is 3dbm, if you need to modify the transmit power, just modify the macro MY\_RF\_POWER\_INDEX. If there is a dynamic modification in the middle of the process, need to call rf\_set\_power\_level\_index (my\_rf\_power\_index) when restoring.
- mesh\_init\_all(): sig mesh related initialization.



## 9.2.6 void proc\_ui()

This function mainly implements UI related processing, e.g. button detect function, as well as corresponding test code.

## 9.3 app\_att.c file introduction

#### pb\_gatt\_provision\_out\_ccc\_cb():

Enable transmission of "provision out" part. Only when "provision\_Out\_ccc" is set as "O1 OO", can mesh node normally return command.

#### pb\_gatt\_Write ():

Callback function corresponding to uuid of "my\_pb\_gatt\_in\_UUID" and used to process provision command.

#### proxy\_gatt\_Write():

Callback function to process proxy command. The command head of proxy command include three types: MSG\_PROXY\_CONFIG, MSG\_MESH\_BEACON, MSG\_NETWORK\_PDU.

- MSG\_PROXY\_CONFIG: It's used to configure white list and black list for proxy communication.
- MSG\_MESH\_BEACON: It's used to control reception of beacon command (notify).
- MSG\_NETWORK\_PDU: It's used to control ON/OFF command.

#### attribute\_t my\_Attributes[]:

Service list in SIG\_mesh containing basic att, as well as att contents related to SIG\_mesh part.

## 9.4 light.c file introduction

#### Modify IO pins

It's only needed to modify IO pins corresponding to PWM\_R / PWM\_G / PWM\_B / PWM\_W.

#define PWM_R	GPI0_PC2	//red
#define PWM_G	GPIO_PC3	//green
#define PWM_B	GPIO_PB6	//blue
#define PWM_W	GPIO_PB4	//white
<pre>typedef struct{</pre>		
u32 gpio;		
u8 id;	// pwm id	
u8 invert;	// pwm invert	feature
u8 func;	// PWM first ;	function or second function
u8 rsv[1];		
}light_res_hw_t	;	
light_res_hw_t	light_res_hw[L]	[GHT_CNT][4];



light\_res\_hw defines PWM IO features.

func: GPIO\_PC1, GPIO\_PC4, GPIO\_PD5 have 2 PWM output functions, here defines whether to use function 1 or function 2.

In PWM macro definition, all 4 leds, i.e., RES\_HW\_PWM\_R/ RES\_HW\_PWM\_G/ RES\_HW\_PWM\_B/ RES\_HW\_PWM\_W, on dongle board/reference board are listed, but for a specific type of light, not all of them are needed, light\_res\_hw is to define this, e.g.:

LIGHT\_TYPE\_CT: select RES\_HW\_PWM\_R and RES\_HW\_PWM\_G, red is for warm light bead, green is for cold light bead.

LIGHT\_TYPE\_HSL: RES\_HW\_PWM\_R, RES\_HW\_PWM\_G, RES\_HW\_PWM\_B are corresponding to RGB beads respectively, when modify light, change HSL to RGB in dim\_refresh(), to drive LED.

LIGHT\_TYPE\_LPN\_ONOFF\_LEVEL: select RES\_HW\_PWM\_R, currently can only control onoff because the low power consumption of retention.

LIGHT\_TYPE\_PANEL: default value is 3, occupying 3 element addresses, with 3 onoff server models, and the corresponding beads of these 3 onoff models are RES\_HW\_PWM\_R/ RES\_HW\_PWM\_G/ RES\_HW\_PWM\_B.

#### Set PWM Frequency

Just modify PWM\_FREQ.

Note: PWM tick overflow will cause STATIC\_ASSERT(PWM\_MAX\_TICK < 0x10000) error when compiling. PWM tick is 16 bits, and the default PWM clock is PLL clock, when PWM\_FREQ is too small, PWM tick will overflow, in this case, user can set PWM frequency division, i.e., PWM\_CLK\_DIV\_LIGHT. Generally it is not needed.

#### ct\_flag:

Used only in LIGHT\_TYPE\_CT\_HSL mode. There are CT bead and HSL bead in this mode, but only 1 will light up at the same time. ct\_flag is 1, indicates this is CT bead and 0 indicates HSL bead.

#### light\_res\_sw\_save:

This variable includes all light status related parameters need to be saved, e.g., lightness, CT and etc. Note, all data are transfer to generic level format (range from -32768 ~ 32767) before saved.

The reason why save data in level format:

- (1) all status value can transfer to level,
- (2) level is the most accurate
- (3) save only 1 parameter for the same status, e.g., for CT value, you can't save both CT value and the transferred level value, because these 2 values may not synchronize.

In general, all status are saved in level format, otherwise may lose accuracy.

#### Nonlinear correspondence of luminance and PWM value

Developer can modify the array "rgb\_lumen\_map[]" according to actual light characteristic.

// 0-100% (pwm's value index: this is pwm compare value, and the pwm cycle is 255\*256)

const u16 rgb\_lumen\_map[101] = {}



#### mesh\_global\_var\_init\_light\_sw():

The initial value when first booting is the default compiling value, when corresponding parameter is saved to flash, then use the saved value.

#### light\_res\_sw\_load():

Status parameter for load light from flash.

#### light\_pwm\_init():

Call this function after read light status. Set to OFF after initializing PWM register, then check if need to enable and if it need transition parameter.

#### light\_par\_save\_proc():

The light status will save to flash 3s after it changed to avoid writing flash too often.

#### light\_dim\_set\_hw():

Set PWM output.It will be executed after node receives  $G\_LEVEL\_SET'/G\_LEVEL\_SET\_NOACK''$  command.

Idx and idx2: light\_res\_hw[idx][idx2].

idx: light count index, e.g., when a BLE module has 2 CT lights

Idx2: light bead index.

#### light\_dim\_refresh ():

When light status changes, call this function to refresh PWM output value. In this function, users can get lightness, CT value and etc., users can calculate PWM value based on this value according to their own dimming algorithm.

The default algorithm is, change the CT value of standard lightness to 0-100 scale, then check rgb\_lumen\_map[101] to find the corresponding PWM output value.

#### get\_light\_pub\_list():

Check all status need to be published when light status changes.

#### temp\_to\_temp100():

Change 800-20000 CT value to 0-100 scale.

#### temp100\_to\_temp():

Change 0-100 scale to 800-20000 CT value.

#### light\_g\_level\_set\_idx\_with\_trans():

```
typedef struct{
```

```
s32 step_1p32768; // (1 / 32768 level unit)
u32 remain_t_ms; // unit ms: max 26bit: 38400*1000ms
u16 delay_ms; // unit ms
s16 present; // all value transfer into level, include CT.
s16 present_1p32768;// (1 / 32768 level unit)
s16 target;
}st_transition_t;
```



Set transition parameter when receive command like level set/lightness set and transition is needed. "1p32768" in st\_transition\_t means dividing one level scale in 32768 units, i.e, the unit of this value is 1/32768, thus can avoid floating calculation and enhance calculation efficiency.

step\_1p32768: the changing value for each LIGHT\_ADJUST\_INTERVAL during transition.

remain\_t\_ms: remain value in the command changes to ms.

delay\_ms: delay\_ms value in the command changes to ms.

present: real time value of level during transition.

present\_1p32768: remaining part of present, unit is 1/32768 level scale.

target: target level.

#### light\_transition\_proc():

Transition polling processing function. When the transition finishes, i.e., level reach target level, if this transition is triggered by scene load, call scene\_target\_complete\_check(i); to label that the scene is valid.

When the transition finishes, check and transmit publish status.

#### led\_onoff\_gpio():

In deep retention sleep or deep sleep mode, the output during sleep is done by setting pull-up/pull-down. In this case, PWM stops working, as well as gpio function, gpio output enable, gpio output registers, only analog registers setting pull-up/pull-down works.

#### proc\_led():

LED indicating light polling processing function.

#### rf\_link\_light\_event\_callback ():

It's LED indicating light register event. By using this method, light blinking is executed in main\_loop and it won't influence processing of other events.

In LPN mode, proc\_led() can not be polled all the time because the SDK enters deep mode, thus the light flashes slowly, not as we expected. Current solution is to set faster flashing parameter when led indicating light is needed, then keep polling proc\_led(), process other function after the flash ends. Normal LPN products need no LED, only for development.

Flashing scene introduction:

LGT\_CMD\_SET\_MESH\_INFO(LGT\_CMD\_PROV\_SUC\_EVE): light flashing when provision succeeds.

LGT\_CMD\_FRIEND\_SHIP\_OK: scene generated by LPN when LPN builds friendship with friend successfully.

LGT\_CMD\_SET\_SUBSCRIPTION: receiving message to modify subscription address

LGT\_CMD\_BLE\_ADV: BLE disconnecting scene, disable by default, only for debug mode.

LGT\_CMD\_BLE\_CONN: BLE connecting scene, disable by default, only for debug mode.

LGT\_CMD\_SWITCH\_POWERON: flash once when power switches to on.

LGT\_CMD\_SWITCH\_PROVISION: switches to PROVISION mode.

LGT\_CMD\_SWITCH\_CMD: switch sends press button command.

PROV\_START\_LED\_CMD: gateway starts provision flow to a node.



PROV\_END\_LED\_CMD: provision flow finishes.

LGT\_CMD\_DUAL\_MODE\_MESH: switch modes in dual mode status.

#### show\_ota\_result():

It's processing function of light blinking indication after OTA is finished.

#### show\_factory\_reset():

It's processing function of light blinking indication after implementing factory reset operation.

How to Introduce Customized Dimming Algorithm:

Default dimming algorithm: refer to light\_dim\_refresh().

Users can modify dimming algorithm by changing light\_dim\_refresh(), i.e, get standard value with this function, then call their own dimming algorithm:

Lightness:

st\_transition\_t \*p\_trans = P\_ST\_TRANS(idx, ST\_TRANS\_LIGHTNESS); u16 lightness = get\_lightness\_from\_level(p\_trans->present);

CT value:

```
u16 temp = light_ctl_temp_prensent_get(idx);
HSL(RGB) value:
```

Refer to light\_dim\_refresh(), get HSL.h/HSL.s/HSL.l or RGB.r/ RGB.g/ RGB.b based on dimming algorithm needs.

On/off: defined by lightness.

# 10 Provisioner (Gateway) Project Introduction

## **10.1 Provisioner Function Introduction**

## 10.1.1 adv-bearer and gatt-bearer

Provisioning is to add an unallocated device into mesh network via Provisioner, so that the device can become a node in the mesh network. The provision process mainly allocates network key and IV index (key parameters to determine whether it's the same network), as well as unicast adr (address allocated in the network). The Provisioner can use network parameters and unicast adr to access and control corresponding node, e.g. turn on/off light, adjust luminance.

Provision supports two types of link channels:

- Implement communication in adv-bearer channel via adv packet. This section mainly introduces the adv-bearer part.
- Implement communication via BLE connection and gatt-bearer. For the implementation of gatt-bearer, please refer to section 7.2, and Android/iOS APP corresponding to SIG\_mesh can implement corresponding functions.

## **10.2 Provisioner Principle**

## 10.2.1 Command Interaction of Provisioner

The provisioner uses "adv-bearer" to add unprovisioned device (unpaired node) into network, and adopts BT channel 37, 38 and 39 to communicate with unprovisioned device. By transferring parameters (e.g. random, key) between the provisioner and unprovisioned device, network parameters and address allocation are exchanged to finally add the unprovisioned device into the network. Please refer to section 5.3 in sig\_mesh document "Mesh\_v1.0" for details.

## 10.2.2 Timing Sequence Chart of adv Provisioner



Figure 10.1: adv Provisioner Timing Sequence Chart

The "provision\_dat" command contains three network parameters including network key, IV index and unicast adr, and implements the function of network formation. Function invoking relationship chart for the packet Tx part of adv-provision:



Figure 10.2: Function Invoking Relationship Chart for the packet Tx Part of Adv-provision

Function invoking relationship chart for the packet Rx part of adv-provision:



Figure 10.3: Function Invoking Relationship Chart for the Packet Rx Part of Adv-provision



## 10.2.3 Timing Sequence Chart of GATT Provisioner

Figure 10.4: gatt provisioner Timing Sequence

By using the method of gatt-provision, the function of provision can be implemented more quickly. The "int  $pb_gatt_Write (void *p)$ " is the entry function of "gatt\_provision" part.

Packet Tx function entry of gatt\_provision:



Figure 10.5: Packet Tx Function Entry of gatt\_provision

Packet Rx function entry of gatt\_provision:

Telink

T



Figure 10.6: Packet Rx Function Entry of gatt\_provision

## 10.3 app.c file introduction

In the provisioner project of current SDK, only the "app.c" file needs customized modifications.

Customization of Adv packet and Adv response packet

Please refer to Section 9.2.1.

#### Configuration of fifo part

Please refer to Section 9.2.2.



#### HCI (USB/UART) Report Data

my\_fifo\_push\_hci\_tx\_fifo (u8 p, u16 n, u8 head, u8 head\_len)

p: point to address of the data to be sent

n: data length

head: head of the data

head\_len: length of the head (O if not specified)

Call my\_fifo\_push\_hci\_tx\_fifo (u8 p, u16 n, u8 head, u8 head\_len) to send data to hci\_tx\_fifo, the hci\_tx\_fifo data will be sent in the callback function. Call back function is registered in user\_init.

- UART: blc\_register\_hci\_handler (blc\_rx\_from\_uart, blc\_hci\_tx\_to\_uart);
- USB: blc\_register\_hci\_handler (app\_hci\_cmd\_from\_usb, blc\_hci\_tx\_to\_usb);

#### app\_event\_handler ():

Refer to section 9.2.3.

#### main\_loop ():

Refer to section 9.2.4.

#### user\_init():

Refer to section 9.2.5.

#### proc\_ui():

The "proc\_ui" function configures IO pin scanning with the interval of 40ms to detect IO change. The interface function "access\_cmd\_onoff" is finally used to send ON/OFF command. By pressing SW1/SW0, an ON/OFF command will be sent with the interval of 100ms.

## 10.4 Provisioner operation and APIs

Mesh stack runs in gateway dogle. Node information are saved in provisioner flash with the address of FLASH\_ADR\_VC\_NODE\_INFO(0x3f000), 1 sect or is 4K, so the maximum saved node number is 200. If more than 200 nodes can not continue to add the corresponding nodes to the network correctly, you need to manually delete some offline nodes, or expand the storage area, to expand the method, please refer to "Method to Modify the Maximum Number of Nodes in a Mesh Network".

## 10.4.1 Format of SIG\_MESH\_TOOL ini file

Gateway operation will use "SIG\_MESH\_TOOL". User can click control buttons on the interface, or send out command via the left "command list" window (double click cmd line) or the bottom "edit control" window (compile cmd and press Enter). The provisioner will handle this in the corresponding branch of app\_hci\_cmd\_from\_usb\_handle after it receives the command.

CD #4g_mesh_master.ini INI BULKOUT ASCII V log C fastkind 2 resry Clear Save Save V Hex Adv Stop Scan rg_scan OTA Bx text Inputees_lines_gift Inputees_lines_gift Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees_lines_get Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees Inputees	8 Telink master Found	BBX 11 / mar	AaBt Autor Autor	Autors Autors Autors	×
<pre>Index Interact for interac</pre>	CMD sig_mesh_master.ini v INI B	ULKOUT ASCII 🔽 Log 🗌 fastbir	nd 2 retry Clear Save Save 🔽 Hex	Adv Stop Scan rp_scan	OTA Rx test
<pre>g_onff_trans_1 g_onoff_trans_0 g_onoff_trans_0 g_onoff_trans_0 femote_prov_capa_get remote_prov_scan_get remote_prov_scan_stop remote_prov_scan_stop remote_prov_link_get remote_prov_link_open</pre>	<pre>ingntness_linear_fff ightness_linear_get ight_col_set ight_col_set ight_col_set ight_col_def_get ight_col_temp_range_set ight_col_temp_range_set ight_col_temp_range_get ight_col_temp_range_get ight_set_set ight_set_set ightness_set ig</pre>	E	LOG print		
remote_prov_link_close cfg_node_identity_get cfg_node_identity_get → ALL ▼ chn_set USB connect Path: search_file Mesh a3 ff 00 00 00 00 00 00 00 00 ff ff 82 02 01 00 → CMD edit UART ▼ GATE_RESET Mesh_ota Gate_ota Prov Close	<pre>g_off_trans_1 g_onoff_trans_0 g_onoff_trans_0 g_onoff_trans_0 g_onoff_get remote_prov_capa_get remote_prov_scan_get remote_prov_scan_get remote_prov_scan_get remote_prov_link_get cemote_prov_link_open remote_prov_link_close cfg_node_identity_get cfg_node_identity_get a3 ff 00 00 00 00 00 20 0f ff 82 02 01 00</pre>		<pre></pre>	search_file E_RESET Mesh_ota Gate_ota	Mesh Prov Close

Figure 10.7: SIG\_MESH\_TOOL

The command has 2 formats in ini file, SIG model and vendor model.

## 10.4.2 SIG model format taking g\_all\_on as an example

	1	2	3	4	5	6	7	8	9	10	11
FI	ag	nk_idx	ak_idx	reliable retry cnt	eliable rsp_max	dst	ор	Onoff	TID	Transition time	Delay
e8	ff	0000	0000	00	00	ff ff	82 02	1	0	nc	nc

#### Figure 10.8: g\_all\_on

The first 2 bytes are identifier, defined by Telink, used to identify communication packet head, for gateway it is 0xE8FF, for app(including mobile app and kma dongle firmware) is 0xA3FF.

Parameter structure as below:

#### typedef struct{

u16 nk_idx;	//netkey index
u16 ak_idx;	//app_key index

u8 retry\_cnt; // time of app layer, when rsp\_max is not received, app layer will retry u8 rsp\_max; // expected answer number u16 adr\_dst; // destination address // first byte of op\_code u8 op; u8 par[MESH\_CMD\_ACCESS\_LEN\_MAX]; //rest byte of op\_code and parameter

}mesh\_bulk\_cmd\_par\_t;

Among them, retry\_cnt is in VC tool, if set to 0, it means to use the value of "retry" control, the default is 2 retry

2. VC tool will automatically change the value of retry\_cnt in the INI data to 2. If it is set to OxFF, it means that no retry is required, that is, the VC tool will automatically change the value of retry\_cnt in the INI data to O.

Refer to set\_material\_tx\_cmd() for detail of nk\_idx, ak\_idx, rsp\_max.

#### Note:

If TID is 0, then it will be maintained and managed by protocol stack, if TID is not 0, then use this as TID, so that users can maintain TID on their own.

### 10.4.3 Vendor Model Format

							<u>)</u>				
:	1	2	3	4	5	6	7	8	9	10	11
FI	ag	nk_idx	ak_idx	reliable retry cnt	eliable rsp_max	dst	ор	op_rsp	tid_pos	para[0]	para[1]
e8	ff	0000	0000	02	00	ff ff	C2 1102	c4	2	1	0

Take CMD-vendor\_on as example

#### Figure 10.9: CMD vender on

Different with SIG model, there are 2 more parameters, op\_rsp and tid\_pos, these 2 parameters are pseudo parameters, and will not be sent to light node.

op\_rsp: set corresponding response opcode(vendor id is not compulsory)

tid\_pos: set tid position (0 means no tid bytes, 1 means tid is in para[0], 2 means para[1]...), op\_rsp and tid\_pos 's configuration should be unified with that of firmware.

The purpose of these 2 parameter, is that provisioner need to support more vendor id's op code, and vendor op may be added at any time, we cannot compile all these information in the program, so we add these information via ini.

### 10.4.4 Burn Nodes

Burn 2 8258 dongles: 1 8258 provisioner node(8258\_mesh\_gw.bin), 1 dongle node(8258\_mesh.bin).

Refer to Debugging Tool Instructions for burning steps.



### 10.4.5 Add Light via Provisioner

The gateway cooperates with the lighting process of the SIG\_mesh\_tool tool and the corresponding command format (hexadecimal representation).

1) Plug the provisioner dongle to the USB port. Start the "SIG\_MESH\_TOOL", and select "tl\_node\_gateway.ini". The top side of the tool will show "Found", which indicates the gateway device (provisioner) is found.

The tool will automatically obtain the uuid and mac address of the gateway. The command format is:

HCI\_CMD\_GATEWAY\_CTL+ HCI\_GATEWAY\_CMD\_GET\_UUID\_MAC

i.e., e9 ff + 10

The gateway will report uuid and mac after receiving it, the format is:

TSCRIPT\_GATEWAY\_DIR\_RSP+HCI\_GATEWAY\_CMD\_SEND\_UUID+uuid(16 bytes) +mac(6 bytes), i.e., 91+99+uuid(16 bytes)+mac(6 bytes).

2) Power on the 8258 dongle, and then click the "Scan" button on the tool to start scanning for devices.

The corresponding command of the scan control is HCI\_CMD\_GATEWAY\_CTL + HCI\_GATEWAY\_CMD\_START: e9 ff + 00

The command corresponding to the stop control is HCI\_CMD\_GATEWAY\_CTL + HCI\_GATEWAY\_CMD\_STOP: e9 ff + 01

8 Telink sig_mest Found 2	1.40		42-4 J		×
CMD tl node gateway.ini	Log retry 0	Clear Save Save	ave Hex Adv	Stop Scan	OTA By test
mesh_bulk_set_par				0	A
gateway_provision_start	ľ				
gateway_provision_stop				-	
gateway_set_network key					
gateway_set_app_key					
gateway_clear_node_para					
gateway_mesh_bulk_cmd_debug					
g_all_on					
g_all_off					
g_mac_on_unrel					
g_mac_off_unrel					
g_all_level32768_unrel					
g_all_level_32767_unrel					
gateway_test command					
test config					
cfg_cps_get					
cfg_sec_nw_bc_get					
cfg_sec_nw_bc_set					
cfg_ttl_get					
cfg_ttl_set					
cfg_nw_transmit_get					
cfg_nw_transmit_set					
cfg_relay_get					
cfg_relay_set					
cfg_friend_get					
cfg_friend_set					
cfg_gatt_proxy_get					
cfg_gatt_proxy_set					
cfg pub get sig					
cfg_pub_set_sig					
cfg_sub_add_sig					
cfg_sub_del_sig					
cfg_sub_del_all_sig					
cfg_sub_ow_sig					
cfg_sub_get_sig					
g_set_level					
gateway_HEARTBEAT_PART					
cfg_heartbeat_pub_get					
cfg_heartbeat_pub_set					
cfg_heartbeat_sub_get					-
cfg_heartbeat_sub_set +	4				
	,				
		USB	UART	connect Prov	Mesh Close

Figure 10.10: Add light via provisioner

 After clicking the scan control button, the gateway will report the received unprovision beacon in the following format:



TSCRIPT\_GATEWAY\_DIR\_RSP+ HCI\_GATEWAY\_CMD\_UPDATE\_MAC+unprovision beacon. i.e.: 91+88+mac(6 bytes)+ unprovision beacon.

The scanned devices will be shown in the device list. Double click the target device which needs provision, the corresponding command is:

HCI\_CMD\_GATEWAY\_CTL+HCI\_GATEWAY\_CMD\_SET\_ADV\_FILTER+6 byte mac address

i.e.:e9 ff + 08 + mac(6 bytes).



Figure 10.11: unprovision beacon

4) Click "Prov" to enter provision interface.



a. The corresponding command of the Provision control is: HCI\_GATEWAY\_CMD\_GET\_PRO\_SELF\_STS.

That is: e9 ff Oc.

b. After receiving the command, the gateway will return whether there is configuration information and the number of elements of the gateway. The corresponding command format is: TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_CMD\_PRO\_STS\_RSP + provision\_flag + pro\_net\_info.

That is: 91 8b + provision\_flag + pro\_net\_info. pro\_net\_info is 25 bytes of provision data. The format is as follows:

```
typedef struct{
    u8 net_work_key[16]; //network key
    u16 key_index; //network key index
    union{
        mesh_ctl_fri_update_flag_t prov_flags;
        u8 flags; // iv update flag
    };
    u8 iv_index[4]; // iv index
    u16 unicast_address;
}provision_net_info_str;
```

TSCRIPT\_GATEWAY\_DIR\_RSP+HCI\_GATEWAY\_CMD\_SEND\_ELE\_CNT+total element: i.e., 91+8c+ total element.

- c. If the provision flag is 0, it means that the gateway has no configuration information. The SetPro Internal control is enabled. Fill in the provision interface with relevant parameters of pro\_net\_info and click SetPro\_interval to set the gateway configuration information. Then two commands will be issued automatically:
- HCI\_CMD\_GATEWAY\_CTL + HCI\_GATEWAY\_CMD\_SET\_PRO\_PARA + pro\_net\_info (the first command)

i.e.,: e9 ff + 09 + pro\_net\_info

 HCI\_CMD\_GATEWAY\_CTL + HCI\_GATEWAY\_CMD\_SET\_DEV\_KEY + unicast address + device key (the second command)

i.e.,: e9 ff + Od +gateway address+device key

If the "SetPro Internal" button is disabled, it indicates the gateway has configured network parameters. Just skip parameter setting step.

provision	
SetPro_internal       test_cmd         send_cmd       ff ff 00 82 05         send       send         provision       send         network_key       11 22 c2 c3 c4 c5 c6 c7 d8 d9 da db dc dd de df	Static aky_idx 00 00 app_key 60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48 bind_all
key_index     00 00     iv_index     11 22 33 44       iv_update_flag     0     unicast_adr     01 00       Provision     Disconnect     V	
filter_operation filter_type white_list v filter_data 11 22 33 44 SetFilter Add_mac RM_mac	

Figure 10.12: SetPro Internal

- 5) Click the "Provision" button to implement provision. During provision process, related log will be printed out and shown on the main interface. After successful provision, the "bind\_all" button becomes enabled.
- a. The corresponding command of the Provision control is: HCI\_CMD\_GATEWAY\_CTL+HCI\_GATEWAY\_ CMD\_SET\_NODE\_PARA+ pro\_net\_info

i.e. e9 ff+0a+ ro\_net\_info.

b. During the Provision process, the allocated addresses are reported in the following format:

TSCRIPT\_GATEWAY\_DIR\_RSP +HCI\_GATEWAY\_RSP\_UNICAST+unicast addr,

i.e. 91+80+unicast address.

c. The node information will be reported after the provision is completed in the following format:

TSCRIPT\_GATEWAY\_DIR\_RSP+ HCI\_GATEWAY\_CMD\_SEND\_NODE\_INFO+ VC\_node\_info\_t

i.e. 91+8d+ VC\_node\_info\_t.

VC\_node\_info\_t is defined as following:

```
typedef struct{
    u16 node_adr; // primary address
    u8 element_cnt;
    u8 rsv;
    u8 dev_key[16];
}VC_node_info_t;
```

d. The status of the provision will be reported after the provision is completed in the following format:



TSCRIPT\_GATEWAY\_DIR\_RSP+HCI\_GATEWAY\_CMD\_PROVISION\_EVT+ gateway\_prov\_event\_t

i.e.: 91 + 89 + gateway\_prov\_event\_t.

gateway\_prov\_event\_t is defined as following:

```
typedef struct{
    u8 eve;//1 means success
    u16 adr;
    u8 mac[6];
    u8 uuid[16];
}gateway_prov_event_t;
```



Figure 10.13: Provision

### 10.4.6 app\_key binding

After provision is finished, it's also needed to bind the app\_key for model by clicking the "bind\_all" button.

a. The command corresponding to bind\_all is: HCI\_CMD\_GATEWAY\_CTL+ HCI\_GATEWAY\_CMD\_START\_KEYBIND
 + fast\_bind ++app\_key index(2 byte)+app\_key(16 bytes).

i.e. e9 ff + Ob + fast\_bind + app\_key index(2 byte)+app\_key(16 bytes).

CMD       tl_node_gat         gateway_provision       gateway_provision         gateway_set_app_kt       gateway_set_app_kt         gateway_clear_motod       gateway_set_app_kt         gateway_set_app_kt       gateway_set_app_kt         gateway_set_app_kt       gateway_set_app_kt         gateway_set_app_kt       gateway_set_app_kt         gateway_set_app_kt       gateway_set_app_kt         g_all_off       g_all_ktewl_street         g_all_level_street       gateway_set_app_kt         g_all_level_street       gateway_set_app_kt         g_ast_off       gateway_set_app_kt         gateway_set_app_kt       gateway_set_app_kt         gateway_set_app_kt       gateway_set_app_kt         gatepp_kt       gateway_set_app_kt	st_cmd end_cmd ff ff 00 82 05 send app_key 60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48 c6 c7 d8 d9 da db dc dd de df [bind_all] iv_index 11 22 33 44 unicast_adr 05 00
ctg_sec_nv_coset     Provision     Disconnect       cfg_ttl_get     filter_operation       cfg_nv_transmit_get     filter_type     filte_list v       cfg_rind_get     cfg_gatt_proxy_get       cfg_gatt_proxy_get     cfg_pub_get_sig       cfg_rbub_get_sig     cfg_mac	r_data 11 22 33 44
cfg sub add sig	<1356>15:10:53:7/6 [INFO]:(GATEWAY)the status notiry data rsp
cfg_ub_del_sig cfg_ub_del_all_sig cfg_ub_del_all_sig cfg_ub_get_sig g_set_level 	<pre>     : 04 00 00 00 11 02 01 00     <pre>     : 04 00 00 00 11 02 01 00     <pre>     : 04 00 01 00 10 :Status Rsp: 04 00 01 00 80 3e 02 04 00 00 00 11 02 01 00     <pre>     : 04 00 01 00 80 3e 02 04 00 00 01 10 2 01 00     <pre>     : 04 00 01 00 80 3e 02 04 00 00 01 10 2 01 00     <pre>     : 04 00 01 00 80 3e 02 04 00 00 01 10 2 01 00     <pre>     : 05 :: 05:: 05:: 05:: 05:: 05:: 05</pre></pre></pre></pre></pre></pre></pre>
	USB UARI Connect Prov Mesh Close

Figure 10.14: bind\_all

When fast\_bind is 1: the gateway will only send appkey add command. The provisioned device needs to enable the default binding function (PROVISION\_FLOW\_SIMPLE\_EN is set to 1).

When fast\_bind is 0: the gateway binds all model ids by default. To save time, users can choose the model ids to be bound. The gateway opens the macro MD\_BIND\_WHITE\_LIST\_EN. For the model ids to be bound, refer to the master\_filter\_list [] in the Mesh\_common.c file. Users can modify it as needed.

b. During the App\_key bind process, the gateway will call u8 gateway\_model\_cmd\_rsp (u8 \* para, u8 len) to return the status information of the bound model in the format: TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_RSP\_OP\_CODE + parameter.

i.e.: 91 + 81 + appkey bind status

Telink

c. App\_key bind will return HCI\_GATEWAY\_CMD\_KEY\_BIND\_EVT after completion, indicating success or time\_out. The format is:

TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_CMD\_KEY\_BIND\_EVT +result

i.e.: 91 + 8a + result. (1:success 2:time\_out)

## 10.4.7 Light on/off Control

After app\_key binding, click "Mesh" to enter mesh interface, or directly double click the INI command "g\_all\_on/g\_all\_off" in the left command window of the main interface to implement on/off control.

😵 Telink sig_mesh Found		2 3 3 5		P 142-4		X
CMD tl_node_gar provision	n				X	Rx test
gateway provisio gateway provisio gateway.set_netw gateway.set_netw gateway.clear.no g_all_off g_mac_of_unrel g_mac_of_unrel g_all_level_3276 gat gat cfg_unel_gat cfg_sec_nw_bc_ge cfg_sec_nw_bc_ge cfg_sec_get cfg_sec_get cfg_nu_transmit_ cfg_nw_transmit_ cfg_rel_set cfg_ria_set cfg_ria_set cfg_ria_set cfg_ria_set cfg_ria_set cfg_ria_set cfg_ria_prox_get cfg_gat_prox_get cfg_gub_set_sig cfg_pub_set_sig cfg_sw_add_sig	Aesh 21 0004 On Off O 100 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	Nodes ffff Group_S GrpDelAll_S GetPub_S SecNwBc TTL transmit Relay GetCPS DelNode 16,00,00,00,00,00	✓ Reliable       Group_C       GrpDeIAII_C       Friend       Proxy       LEVEL       C/T       RFU	Group         Svr         Clut           0         On         Off         Svr         Clut           1         On         Off         I         I           2         On         Off         I         I           3         On         Off         I         I           4         On         Off         I         I           5         On         Off         I         I           6         On         Off         I         I           7         On         Off         I         I           8         On         Off         I         I           9         On         Off         I         I           10         On         Off         I         I           11         On         Off         I         I           12         On         Off         I         I           13         On         Off         I         I	Ē	
<pre>cfg_sub_del_all_sig cfg_sub_ow_sig cfg_sub_get_sig g_set_level </pre>	-gateway_HEARTBEAT_PART	<1361>15:12:23:239 <1362>15:12:23:255 : 01 00 01 00 82 44 <1363>15:12:23:328 <1364>15:12:23:328 <1364>15:12:23:347 : 04 00 01 00 82 44	[INFO]:Status Rsp: [INFO]:(GATEWAY)th iff ff [INFO]:Status Rsp: [INFO]:(GATEWAY)th iff ff	01 00 01 00 82 4e ff ff e status notify data rsp 04 00 01 00 82 4e ff ff e status notify data rsp "" B UART <b>c</b>	onnect Prov 1	Mesh Close

Figure 10.15: Light on/off control

After sending the onoff command, the corresponding node reports the status in the format:

Status Rsp\_\_\_\_\_: 04 00 01 00 82 04 00 01 0a

The corresponding structure is:

Telink

т

<pre>typedef struct{</pre>	
u16 len; // <i>length</i>	
u16 src; // source address	
u16 dst; // destination address	
u8 data[ACCESS_WITH_MIC_LEN_MAX];	<pre>// access layer(op code, parameters)</pre>
<pre>}mesh_rc_rsp_t;</pre>	

## **10.4.8 Provisioner Control Flow Chart**



Figure 10.16: Provisioner Control Flow Chart

## 10.4.9 Smart Provision

One Click networking corresponds to Smart Provision inside the code.

The one-click networking function is based on the gateway project, and then remove sig\_mesh\_tool.exe tool on PC, only used the gateway dongle for networking. It is applicable to simple network scenarios, the later also do not need through the UI interface to the network nodes for too much configuration, mainly to use the commonly used control functions.

### 10.4.9.1 Difference between Smart Provision and Normal Networking

- Gateway normal networking mode: It requires a host computer or an app to perform networking, and after the networking is completed, various configurations can be performed on the network nodes through the host computer.
- Smart Provision mode: You can network without the host computer, need gateway dongle only.

In addition, in order to simplify the networking process and save networking time, the node side needs to open the PROVISION\_FLOW\_SIMPLE\_EN, so that after the gateway dongle sends the App key add during networking, it ends the networking process and no longer sends the app key bind command, and the node to be networked automatically triggers the app key bind on itself after receiving the App key add.

If PROVISION\_FLOW\_SIMPLE\_EN is not turned on at the node side, then it needs:

- (1) Modify prov\_uuid\_fastbind\_mode() to return 1 directly inside the function.
- (2) Change GATEWAY\_APPKEY\_ADD\_HEAD from

{(u8)HCI\_CMD\_GATEWAY\_CTL, HCI\_CMD\_GATEWAY\_CTL>>8, HCI\_GATEWAY\_CMD\_START\_KEYBIND, 1}

to

{(u8)HCI\_CMD\_GATEWAY\_CTL, HCI\_CMD\_GATEWAY\_CTL>>8, HCI\_GATEWAY\_CMD\_START\_KEYBIND, 0}

## 10.4.9.2 Principle Decription

The distribution process of one-key networking uses the standard distribution method of sig, only that the command interaction process of the host computer distribution is moved to the application layer of the dongle firmware at gateway side. After the provision start key is pressed, the main process mesh\_smart\_provision\_proc() handles the provisioning status and simulates the host computer to push commands into the hci rx fifo. In the interface function gateway\_common\_cmd\_rsp(), which is reported to the host computer, mesh\_smart\_provision\_rsp\_handle() is called to handle the message processing.

The initial iv index of the gateway after network allocation is SMART\_IV\_INDEX. Network key and app key are randomized values. See smart\_gateway\_provision\_data\_set() for details.

When networking, if you only want to add nodes that meet certain conditions, you can modify the filtering rules inside the function prov\_uuid\_fastbind\_mode() in the HCl\_GATEWAY\_CMD\_UPDATE\_MAC branch of the mesh\_smart\_provision\_rsp\_handle() function.

### 10.4.9.3 Function Decription

The process of one-key distribution will be the same as normal networking, the gateway will send invite, start and other commands. For example: mesh\_adv\_prov\_send\_invite() sends invite command, mesh\_adv\_prov\_send\_start\_cmd() sends start command.

## **10.4.9.4 Testing Process**

(1) SDK settings

Gateway: need to turn on SMART\_PROVISION\_ENABLE

Node: need to turn on PROVISION\_FLOW\_SIMPLE\_EN

If you do not want PROVISION\_FLOW\_SIMPLE\_EN to be turned on at the node side, please refer to the introduction inside "Difference between Smart Provision and Normal Networking" to configure it.

(2) Initial networking

Gateway one-key network function, enable this function to burn 8258\_mesh\_gw.bin file to 8258 dongle and then press the key SW2, the gateway will automatically add the unallocated nodes within one-hop range to the network, within 30 seconds the unallocated nodes cannot be searched for to exit the process of network allocation, press SW1 to control the switching of nodes in the network.

(3) The steps for adding node again after the network has been established and running for a period of time.

Press the networking key SW2 again.

# 11 Mesh LPN Project Introduction

## 11.1 LPN Node and Implementation Method

Note: All figures in this section are derived from Sig Mesh spec.

## 11.1.1 LPN and friend

Low-Power feature: Rx side can run with obviously low duty cycle in mesh network. By enabling radio receiver only when necessary, the duty cycle is minimized to decrease node's power consumption. This is implemented by establishing friendship between LPN (Low Power Node) and FN (Friend Node).

A LPN can only establish friendship with a single FN, while a FN can establish friendship with multiple LPNs.

When a friendship is established, if the LPN node has previously established a friendship with another FN node, the LPN will inform the new FN node through the friend request command, and the FN node will call friend\_cmd\_send\_clear() to send the clear command to notify the old FN node to clear the friendship with the LPN counterpart. The low-power node will poll (Poll) the friend node with a longer period, say 2 seconds (FRI\_POLL\_INTERVAL\_MS) or longer. Check to see if there is a new message, and if so, get the message. After establishing the friendship, the LPN node reports the current subscription list (i.e., all group number information) to the FN node. Then if the FN receives a message whose destination address matches these group numbers or the LPN's element address, it caches the message, and then sends the cached message to the LPN when it receives a Poll command from the LPN. the FN and the LPN interact with each other with the Poll and update commands, which have iv index information in them, and can perform the iv update flow.

Friend feature: To help LPN running, the FN will store the information to be sent to the LPN, and only initiate transmission when there's obvious request from the LPN.

## 11.1.2 Friendship Parameters

LPN needs to find FN and initiate a "Friendship Establish" process to establish friendship with it. Following shows some key parameters which are configured during the "Friendship Establish" process and serve to manage LPN behavior.

- ReceiveDelay is the time that elapses between when the LPN sends a request to the buddy node and when it starts listening to the response. This gives the friend node time to prepare the response and send it back. Specified by the LPN through this macro FRI\_REC\_DELAY\_MS and communicated to the FN through the friend request command.
- 2) ReceiveWindow is the timing used by the LPN to listen for responses. Specified by the FN through this macro FRI\_REC\_WIN\_MS and communicated to the LPN through the friend offer command. the following figure depicts the timing involving ReceiveDelay and ReceiveWindow.



Figure 11.1: Timing Sequence of ReceiveDelay and ReceiveWindow

3) PollTimeout sets the maximum time that may elapse between two consecutive requests sent by the LPN to its friend node. It is specified by the LPN through this macro LPN\_POLL\_TIMEOUT\_100MS and is communicated to the FN through the friend request command. If the friend node fails to receive a request from the LPN before the PollTimeout timer expires, the friendship relationship will be terminated.

## 11.1.3 Establish Friendship

The process to establish friendship in BT mesh network is shown as below:

**Step 1** LPN issues a "Friend Request" message which does not support relaying. Only the FN within the direct radio range will process this message, and other nodes without "friend" features will discard this message. The "Friend Request" message contains parameters of LPN, including "ReceiveDelay", "ReceiveWindow" and "PollTimeout".

**Step 2** If a FN nearby supports specific requirement in the "Friend Request" message, it will prepare a "Friend Offer" message and send it back to the LPN. This message contains various parameters, including supported ReceiveWindow size, available message queue size, available subscription list size, and RSSI value measured by the FN.

**Step 3** When the LPN receives the "Friend Offer" message, it will adopt a specific algorithm to select suitable FN. This accurate algorithm may take various cases into consideration: Some device may give priority to the ReceiveWindow size, so as to minimize power consumption; some device may pay more attention to the RSSI value, so as to ensure high-quality link with FN. It depends on product developer.

**Step 4** The LPN will send a "Friend Poll" message to the selected FN.

**Step 5** After the "Friend Poll" message from the LPN is received, the FN will respond with a "Friend Update" message to finish "Friendship Establish" process and supply security parameters.



## 11.1.4 Friendship Message Exchange

After friendship is established, the FN will store all messages of the LPN in the "Friend Queue". These messages are so-called "stored message". The figure below shows message exchange between the FN and the associated LPN.



#### Figure 11.3: Friendship Message Exchange

When the FN receives a message from the LPN addressing to this node, the FN will buffer this message by



storing it in the "Friend Queue" area. As shown in the figure above, the FN stores "Message 1" and "Message 2" for the LPN.

The LPN will periodically enable its transceiver, and send "Friend Poll" message to the FN so as to check whether there's any stored message buffered for the LPN.

The FN will first send a stored message to the LPN as the response to the "Friend Poll".

After each reception of message from the FN, the LPN will continue to send "Friend Poll" message until it receives a "Friend Update" message with the "MD (More Data)" field set as "O". "MD=O" means there's no more message buffered in the FN for the LPN. Then the LPN stops the polling to the FN.

## 11.1.5 Security

Master Security Material: It's derived from network key (NetKey), and it can be used by other nodes within the same network. Message encrypted by using "Master Security Material" can be decoded by any node within the same network.

Friend Security Material: It's derived from network key (NetKey), as well as extra counter number generated by the LPN and FN. Message encrypted by using "Friend Security Material" can only be decoded by the LPN and the FN processing this message.

Friend messages encrypted by using "Friend Security Material" include: "Friend Poll", "Friend Update", and "Friend Subscription List".

Friend messages encrypted by using "Master Security Material" include: "Friend Clear" and "Friend Clear Confirm".

Any other non-control message from the LPN to the FN will set the "credential\_flag" in corresponding model publish parameter as needed, so as to determine whether the encryption method is "Master Security Material" or "Friend Security Material". The default value of the "credential\_flag" is 0, corresponding to "Master Security Material" encryption.

### 11.1.6 Friendship Termination

If the FN fails to receive a "Friend Poll", "Friend Subscription List Add" or "Friend Subscription List Delete" message before the "PollTimeout" expires, the friendship between the FN and the LPN is terminated.

The LPN can initiate friendship termination program by sending a "Friend Clear" message to the FN, so that the FN will terminate their friendship.

## 11.2 Friendship Sleep and Working Mechanism

### 11.2.1 FN Receive Packet Processing Interface

void mesh\_friend\_ship\_proc\_FN(u8 \*bear)

• Where bear is not empty, it indicates that a friendship-related command was received.



• When bear is empty, it means it is a polling call inside main\_loop() to detect and handle timing events and timeout events.

```
void mesh_friend_ship_proc_FN(u8 *bear)
{
    foreach(i,g_max_lpn_num){ // a friend node may establish friendship with many LPN, so check
  all LPN.
\hookrightarrow
        mesh_fri_ship_proc_fn_t *proc_fn = &fri_ship_proc_fn[i];
        if(!bear){
           if(proc_fn->status){ // (FRI_ST_IDLE != proc_fn->status)
               if(FRI_ST_OFFER == proc_fn->status){
                   if(clock_time_exceed(proc_fn->offer_tick, proc_fn->offer_delay*1000)){
                        . . . . . .
                       // send friend offer and set to state of receiving friend poll after
                        → received friend request.
                       friend_cmd_send_fn(i, CMD_CTL_OFFER);
                       . . . . . .
                       mesh_friend_ship_set_st_fn(i, FRI_ST_POLL);
                    }
               }else if(FRI ST POLL == proc fn->status){
                   // add 500ms, because handling response of POLL was delay some ten ms.
                   if(clock_time_exceed(proc_fn->offer_tick,
                    // timeout to receive friend poll from LPN after send friend offer to
                        \leftrightarrow LPN,
                       // means that LPN did not receive offer, or LPN did not select current
                        \leftrightarrow FN as friend node,
                       // or FN did not receive the friend poll from LPN.
                       mesh_friend_ship_proc_init_fn(i);
                   }
               }else if(FRI_ST_TIMEOUT_CHECK == proc_fn->status){
                   if(clock_time_exceed_100ms(proc_fn->poll_tick, (u32)
                    // timeout to receive friend poll from LPN, then will disconnect this
                        ↔ friendship.
                       friend ship disconnect fn(i, FS DISCONNECT TYPE POLL TIMEOUT);
                   }
               }
           }
           if(proc_fn->clear_poll){ // clear by other FN
               if(clock_time_exceed_100ms(proc_fn->clear_start_tick, (u32)
                // when the timeout expires, even if the clear response has not been
                    \leftrightarrow received yet, the clear command will stop being sent.
                   mesh_stop_clear_cmd(i);
               }else{
```

```
if(clock_time_exceed_100ms(proc_fn->clear_cmd_tick,

→ proc_fn->clear_int_100ms)){

                        . . . . . .
                        // Gradually reduce the frequency of sending clear commands.
                        // please refer to mesh V1.1 spec "Figure 3.24: Friend Clear procedure
                        ↔ example" of "3.6.6.3.1 Friend establishment".
                        friend_cmd_send_fn(i, CMD_CTL_CLEAR);
                    }
                }
            }
            if(proc_fn->clear_by_lpn_tick && clock_time_exceed(proc_fn->clear_by_lpn_tick,
             → 5*1000*1000)){
                // when received friend clear, should not clear at once, and need to delay some
                → time to clear Friendship.
                // because LPN may retry sending friend clear command when not receive clear

→ confirm.

                friend_ship_disconnect_fn(i, FS_DISCONNECT_TYPE_CLEAR);
            }
        }else{
            ..... // to process packet received
        }
    }
}
```

## 11.2.2 Processing Interface for Packets Sent by FN to LPN

```
mesh_friend_response_delay_proc_fn()
```

When FN needs to send packet to LPN, for example, send friend update, when poll delay reaches the end, it needs to send the packet as soon as possible, instead of waiting for the adv interval (10ms) set by bls\_II\_setAdvParam() to reach like other ordinary network PDUs, then send the packet. So we poll the tick with mesh\_friend\_response\_delay\_proc\_fn(), and when the time is up, call fn\_quick\_send\_adv() to send the packet immediately. In addition, since the message sent to the LPN needs to be checked again to see if the message needs to be updated, such as whether the segment block ack needs to be updated, etc. (refer to the processing of get\_cache\_buf\_for\_poll()), it needs to wait until the update is done, and then perform the encryption of the network layer before sending. See the handling of mesh\_friend\_response\_delay\_proc\_fn() below for details:

```
void mesh_friend_response_delay_proc_fn(u8 lpn_idx)
{
    fn_ctl_rsp_delay_t *p_delay = &fn_ctl_rsp_delay[lpn_idx];
    int print_cache_flag = 0;
```

```
if(p_delay->delay_type && clock_time_exceed(p_delay->tick, fn_req[lpn_idx].RecDelay * 1000 -
    → 1800)){
                // 1800us: encryption pkt time
       if(DELAY_POLL == p_delay->delay_type){
           if(p_delay->poll_rsp){
               if(fn_other_par[lpn_idx].cache_overwrite){
                   p_delay->poll_rsp = get_cache_buf_for_poll(lpn_idx, 1, 1); //
  cache_overwrite will be clear inside.
               }
               if(bear_tx_len <= MESH_BEAR_SIZE){</pre>
                    . . . . . .
                   // no encryption before, because need to check buffer in
                    → mesh_fri_cmd2cache_(), then to set cache_overwrite or not.
                   mesh_sec_msg_enc_nw_rf_buf((u8 *)(&bear_temp->nw),
→ mesh_lt_len_get_by_bear(bear_temp), FRIENDSHIP,
→ lpn_idx,0,fn_other_par[lpn_idx].nk_sel_dec_fn, 0);
                    . . . . . .
                   mesh_tx_cmd_add_packet_fn2lpn((u8 *)bear_temp);
               }
               . . . . . .
           }
           mesh_fri_ship_proc_fn_t *proc_fn = &fri_ship_proc_fn[lpn_idx];
           if(proc_fn->clear_delay_cnt){
               proc_fn->clear_delay_cnt--;
               if(0 == proc_fn->clear_delay_cnt){ // make sure establish friendship success
                   friend_cmd_send_fn(lpn_idx, CMD_CTL_CLEAR); // use normal fifo, not
→ mesh_adv_fifo_fn2lpn_
                    . . . . . .
               }
           }
       }else if(DELAY_SUBSC_LIST == p_delay->delay_type){
           friend_cmd_send_subsc_conf(p_delay->adr_dst, (u8)p_delay->par_val);
       }else if(DELAY_CLEAR_CONF == p_delay->delay_type){
           . . . . . .
           friend_cmd_send_clear_conf(clear.LPNAdr, (u8 *)&clear,
sizeof(mesh_ctl_fri_clear_t));
       }
       p_delay->delay_type = 0;
   }
   if(my_fifo_data_cnt_get(&mesh_adv_fifo_fn2lpn)){
                             // "poll rsp" may be delay when in BLE_S window, so quickly send
       fn_quick_send_adv();
  here again. and also "send_subsc_conf /send_clear_conf" need quick send.
```

}

}

## 11.2.3 LPN Packet Processing Interface

```
void mesh_friend_ship_proc_LPN(u8 *bear)
```

- Where bear is not empty, it indicates that a friendship-related command was received.
- When bear is empty, it means it is a polling call inside main\_loop() to detect timing events.

```
void mesh_friend_ship_proc_LPN(u8 *bear)
{
    . . . . . .
    if(!bear && is_mesh_adv_cmd_fifo_empty()){
        if(fri_ship_proc_lpn.poll_retry && clock_time_exceed(fri_ship_proc_lpn.poll_tick,
         → poll_retry_interval_ms*1000)){
            fri_ship_proc_lpn.poll_retry--;
            if(0 == fri_ship_proc_lpn.poll_retry){
                ..... // Logic for handling FN replies that are not received after the poll has
↔ been sent and the time limit has expired
            }
        }
        else if(subsc_list_retry.retry_cnt && clock_time_exceed(subsc_list_retry.tick,

→ timeout_ms)){

            subsc_list_retry.tick = clock_time(); // also refresh when send_subsc
            subsc_list_retry.retry_cnt--;
            ..... // Logic for handling when the LPN reports the subscription list after the
_{\leftrightarrow} friendship has just been successfully established and no response is received from the FN
\rightarrow after the timeout period.
        }
    }
    mesh_cmd_bear_t *p_bear = (mesh_cmd_bear_t *)bear;
    //mesh_cmd_nw_t *p_nw = &p_bear->nw;
    mesh_cmd_lt_ctl_unseg_t *p_lt_ctl_unseg = &p_bear->lt_ctl_unseg;
    u8 op = -1;
    if(bear){
        op = p_lt_ctl_unseg->opcode;
    }
    if(0 == fri_ship_proc_lpn.status){ // LPN Processing branch after a friendship has been
    \rightarrow successfully established, or before a friend request has been sent.
        if(bear){
            if(CMD_CTL_SUBS_LIST_CONF == op){
```
```
..... // After sending Friend Subscription List Add, the processing branch of
↔ Friend Subscription List Confirm is received.
                      // See 3.6.5.7 Friend Subscription List Add for details.
           }else if(CMD_CTL_UPDATE == op){
               . . . . . .
               // Receives the Friend update processing branch. Includes processing of the iv

→ index, etc.

               iv_update_key_refresh_rx_handle(&p_update->flag, p_update->IVIndex);
           }
       }else{
           if(is_friend_ship_link_ok_lpn() && is_mesh_adv_cmd_fifo_empty() &&
             clock_time_exceed(fri_ship_proc_lpn.poll_tick, get_lpn_poll_interval_ms() *
            → 1000)){
               // When the LPN doesn't need to sleep at a certain time, then it can't execute
                ↔ the event of sending a friend poll periodically via
                → mesh_friend_ship_start_poll() inside user_init_deepRetn(). So here we add
                \rightarrow the handling of checking again if we need to send a friend poll or not. If
                \leftrightarrow there is a retention wakeup, then the processing here is not executed.
               mesh_friend_ship_start_poll();
           }
       }
   }else{
       switch(fri_ship_proc_lpn.status){ // Be true only during establishing friendship.
           case FRI_ST_REQUEST:
               if(is_mesh_adv_cmd_fifo_empty() && clock_time_exceed(fri_ship_proc_lpn.req_tick,
                → FRI_REQ_TIMEOUT_MS * 1000)){
                    . . . . . .
                   friend_cmd_send_request();
                    . . . . . .
                   mesh_friend_ship_set_st_lpn(FRI_ST_OFFER); // After sending the request, it
\leftrightarrow enters the state of waiting to receive the offer.
               }
               break;
           case FRI_ST_OFFER:
               if(bear){
                   if(CMD_CTL_OFFER == p_lt_ctl_unseg->opcode){
                       if(0 != lpn_rx_offer_handle(bear)){ // Includes a comparison to select
                        → an optimal FN
                            break;
                       }
                   }
               }else{
                   if(clock_time_exceed(fri_ship_proc_lpn.req_tick,
                    → FRI_ESTABLISH_PERIOD_MS*1000)){
                       if(mesh_lpn_par.FriAdr){
                            mesh_lpn_par.link_ok = 1;
```

```
mesh_friend_key_update_all_nk(0, 0); // After 1 second, determine
  the best FN and then update the corresponding friend key
                            . . . . . .
                       }
                       mesh_friend_ship_set_st_lpn(FRI_ST_POLL);// Enter the state of sending
  Friend Poll
                   }
               }
               break:
           case FRI_ST_POLL:
               if(is_friend_ship_link_ok_lpn()){
                    if(is_mesh_adv_cmd_fifo_empty()){
                       mesh_lpn_par.poll.FSN = 0; // init
                       // send poll
                       fri_ship_proc_lpn.poll_retry = FRI_GET_UPDATE_RETRY_MAX + 1;
\hookrightarrow
                        friend_cmd_send_poll(); // Press the Friend poll into the send packet
\rightarrow fifo, checking at the top of mesh_friend_ship_proc_LPN() when the time is up before sending
  the packet
                        t_rec_delay_and_win = mesh_lpn_par.req.RecDelay +
   mesh_lpn_par.offer.RecWin;
                       mesh_friend_ship_set_st_lpn(FRI_ST_UPDATE);// Go to Waiting to receive
  friend update
                   }
               }else{
                    lpn_no_offer_handle(); // Check that if no offer is received during the
\leftrightarrow Waiting to Receive Offers phase, the friendship creation fails and a resend of the Friend
  request is initiated
               }
               break;
           case FRI_ST_UPDATE:
               if(bear){ // current state is establishing friendship
                    if(CMD_CTL_UPDATE == p_lt_ctl_unseg->opcode){
                       // Friend update received, Friendship creation complete.
                       //friendship establish done
                       mesh_lpn_par.req.PreAdr = mesh_lpn_par.FriAdr;
                        iv_update_key_refresh_rx_handle(&p_update->flag, p_update->IVIndex);
                       mesh_friend_ship_proc_init_lpn();
                        friend_ship_establish_ok_cb_lpn();
                    }
               }else{
                    if(clock_time_exceed(fri_ship_proc_lpn.poll_tick,
                    → t_rec_delay_and_win*1000)){
                       // If no Friend update is received after the timeout period, return to
                        ↔ the FRI_ST_POLL phase and resend the Friend Poll.
```

# 11.2.4 FriendShip Sleep Mechanism

Timed events, including timed wake-up packets, are based on the soft timer mechanism. For soft timer related content, please refer to this section "Application of Soft Timer".

The mesh\_lpn\_adv\_interval\_update() refreshes the broadcast (wake-up) interval of the LPN according to the different states of the LPN, thus changing the interval of the friend request/poll commands.

# 11.2.5 Friendship Working Mechanism

The LPN node enables the low power management mechanism of BLE by turning on BLE\_REMOTE\_PM\_ENABLE. The details of this mechanism can be found in the BLE handbook, such as "AN-21112301-C\_Telink B85m BLE Single Connection SDK", "Developer Handbook.pdf" in the "Low Power Management (PM)" section. In short, the mechanism is realized by soft timer:

- Inside the user init, the sleep management module is registered with blc\_ll\_initPowerManagement\_module(), including ll\_module\_pm\_cb, etc.
- In ADV state, ADV interval is defined, and soft timer realizes to send broadcast packet once per interval, and then main loop executes to sleep management unit II\_module\_pm\_cb() in blt\_sdk\_main\_loop(), then soft timer sets the next wakeup time point according to ADV interval, and then enters into sleep. Then when the time is up, MCU wakes up, executes user\_init\_deepRetn(), and sends the next broadcast packet .....
- In the GATT connected state, the interval becomes the connected interval; the other mechanisms are the same.

The working mechanism of LPN is as follows:

- (1) At the beginning, it is in un-networked state, user\_init() -> user\_init\_peripheral -> mesh\_lpn\_adv\_interval\_update( will wake up and send the connectable broadcast packet periodically with the interval of the connectable broadcast packet as the soft timer event. By default, PB-ADV and PB-GATT are supported, so inside user\_init\_peripheral(), judge and call bls\_pm\_setSuspendMask (SUSPEND\_DISABLE) to turn off the Sleep mechanism if it is in un-networked state.
- (2) After the lpn node allocates the netkey and other information in the network, the provisioner starts the key bind process. Since the time of the key bind process is uncertain, the LPN is judged by the mesh\_lpn\_state\_proc(), and when no key bind command is received for 3 seconds (LPN\_START\_REQUEST\_AFTER\_BIND\_MS), the entire provisioning process is considered to have been

completed. Then it calls the mesh\_friend\_ship\_set\_st\_lpn(FRI\_ST\_REQUEST) interface to enter the FRI\_ST\_REQUESTt state, and call mesh\_friend\_ship\_set\_st\_lpn() inside mesh\_friend\_ship\_set\_st\_lpn() to set the interval of the soft timer periodic event to FRI\_REQ\_TIMEOUT\_MS.

```
void mesh_lpn_state_proc()
{
. . . . . .
    if(lpn_provision_ok){
        . . . . . .
    }else{
        if(!is_provision_success()){
             . . . . . .
        }else{
            if((!lpn_provision_ok) && node_binding_tick && clock_time_exceed(node_binding_tick,
             → LPN_START_REQUEST_AFTER_BIND_MS*1000)){
                lpn_provision_ok = 1;// provison and key bind finish
                gatt adv send flag = GATT LPN EN;
                mesh_friend_ship_set_st_lpn(FRI_ST_REQUEST);
                if(BLS_LINK_STATE_CONN == blt_state){
                     bls_ll_terminateConnection(0x13); // disconnect to establish friendship
                }
            }
        }
    }
. . . . . .
}
```

Or, after powering down and re-powering up, inside proc\_ui() call mesh\_friend\_ship\_set\_st\_lpn(FRI\_ST\_REQUEST).

(3) After that, send Friend Request in mesh\_friend\_ship\_proc\_LPN() to enter the friendship creation process.

Note: For every mesh message sent by LPN, it will call mesh\_lpn\_sleep\_prepare(u16 op) function to set the PM and update the callback function and time point for the next task via soft timer.

The function friend\_cmd\_send\_request() sends the Friend Request by executing mesh\_lpn\_sleep\_prepare() to set the next wakeup point after FRI\_ESTABLISH\_REC\_DELAY\_MS and then lpn\_quick\_tx() to send the packet immediately.

- (4) After sending friend request, it will wait for FRI\_ESTABLISH\_PERIOD\_MS (default is 1.1 seconds), within 1.1 seconds, if it doesn't receive any friend offer, the MCU will set the next wakeup time point according to the soft timer in blt\_sdk\_main\_loop(), and then go to sleep in the sleep management unit II\_module\_pm\_cb(). The MCU will set the next wakeup time according to the soft timer, and then go to sleep. When the time is up, it will wake up and continue to send friend request.
- (5) If a friend offer is received, the process of packet receipt processing and establishing a Friendship is performed, as described in Ipn packet processing interface.



- (6) When the Friendship is established, mesh\_lpn\_adv\_interval\_update() is executed to update the base wakeup period to the poll interval, in addition to this base wakeup period, there are also timer events that are added by the call to mesh\_lpn\_sleep\_prepare() for each packet sending event, and so on.
- (7) Sending a friend poll is triggered by mesh\_friend\_ship\_start\_poll(). It is currently called in three places:
  - user\_init\_deepRetn()->mesh\_friend\_ship\_start\_poll() This is a normal send, i.e., every time the poll interval wakes up, it will be called again to send the poll periodically.
  - mesh\_lpn\_poll\_md\_wakeup()->mesh\_friend\_ship\_start\_poll() This is called when it is detected that the Friend Node's cache still has data to be fetched.
  - mesh\_friend\_ship\_proc\_LPN()->mesh\_friend\_ship\_start\_poll() This is only triggered in special cases. I.e., if you don't enter sleep at a certain time, there is no way to trigger sending a poll via user\_init\_deepRetn()->mesh\_friend\_ship\_start\_poll(), so it is triggered here.

# 11.2.6 Mechanism for LPN to Receive a Destination Address as a Group Number

- Each time a friendship is created, the LPN sends the subscription list add command (CMD\_CTL\_SUBS\_LIST\_ADD).
- The FN node stores the group number list when it receives it, see the processing of friend\_subsc\_list\_add\_adr() for more details
- Subsequently, when the FN receives commands from other nodes with a destination address that
  matches the group number in the group number list, it helps the LPN to cache the information and
  sends it to the LPN for processing when it receives the poll command from the LPN. When testing,
  configure a group number for the LPN. Then, every time the LPN creates a friendship, the LPN will
  automatically issue CMD\_CTL\_SUBS\_LIST\_ADD.

# 11.3 Common Parameter Configuration for LPN

FN stands for Friend Node and LPN stands for Low Power Node for the following contents.

# 11.3.1 Friend Node

- FN\_CACHE\_SIZE\_LOG: The maximum number of messages to be cached for LPN is FN\_CACHE\_SIZE\_LOG times 2.
- FRI\_REC\_WIN\_MS: The minimum reception window required by FN for LPN, default is 20ms. It indicates the time to listen to the broadcast packet after LPN sends Poll, if timeout occurs, it means that Friend node's reply is not received. Then the Poll command will be retransmitted.FRI\_REC\_WIN\_MS cannot be set too small because there are 3 channels for broadcast packet sending and the possibility that the FN is dealing with something else with higher priority, resulting in the timing of the FN's reply to the LPN not being as precise as it should be.

# 11.3.2 Low Power Node

• FRI\_REQ\_TIMEOUT\_MS: Configure the interval for sending friend request. The default is 2 seconds. If the product definition requires lower power consumption, it can be increased according to the actual situation.

- FRI\_ESTABLISH\_WIN\_MS: the maximum time to wait for receiving Friend offer after sending Friend offer. The spec specifies that the time is 1 second, because we want to receive offers from as many FNs as possible, and then choose the best FN. Generally it is not recommended to change this value. However, if the product requires very low power consumption, and only modifying FRI\_REQ\_TIMEOUT\_MS can not meet the demand, then we can consider changing FRI\_ESTABLISH\_WIN\_MS to a smaller value.
- FRI\_POLL\_INTERVAL\_MS: interval of friend poll. The default is 2 seconds, which is a short time, mainly because it is used for single fire switch low power devices, and the command response time can not be too long. If the product definition requires low power consumption, it can be changed according to the actual situation.
- FRI\_POLL\_RETRY\_MAX: LPN does not receive any reply from FN after sending Poll command, when the number of times exceeds this value, LPN will flip the value of FSN in the poll once, and then send the Poll again, if it still doesn't receive any reply from FN for the consecutive FRI\_POLL\_RETRY\_MAX times, LPN will consider that FN is offline. At this point, LPN will disconnect the current friendship and start to send friend request to try to establish friendship with other friend node.
- LPN\_SCAN\_PROVISION\_START\_TIMEOUT\_MS: It means that after LPN sends a friend request, no offer has been received from FN node, if the time exceeds this time, LPN will go to sleep in order to save power, and need to wake up by pressing the key to start sending friend request again. The default time is 60 seconds.

# 11.4 LPN Demonstration

# 11.4.1 Hardware

Telink

This demo is based on the GATT master dongle mode. The operation steps of the APP and gateway modes are similar to the GATT master dongle mode. Note that in the gateway mode, the gateway node itself also supports the friend function.

One 8269 GATT master dongle and two 8258 mesh dongle (one burns 8258\_mesh.bin, supports Friend function by default. The other burns 8258\_mesh\_LPN.bin, which is the LPN node).

### Note:

- LPN supports generic ONOFF by default, generic Level, but can not support lightness and light CT.
- LPN does not receive Oxffff destination addresses. It only receives unicast addresses and subscribed group numbers. Because there are too many Oxffff commands in the air, if the LPN polling interval is long, the commands in the friend cache will be flushed easily.

# 11.4.2 Test method

The time-related macros mentioned below can be modified by customers according to their actual needs.

**Step 1** Mesh friend node (FN) is powered on and provisioned with SIG\_MESH\_TOOL.

**Step 2** Powered on unprovisioned LPN node, at this time, the LPN is in awake state.

After the LPN node is powered on, the red LED will be in the ON mode. In the unprovision state, do not enter the sleep mode, the purpose is to support GATT provision and ADV provision. In this state, if the provisioning process has not started after 1 minute (LPN\_SCAN\_PROVISION\_START\_TIMEOUT\_MS), then the system will enter the deep sleep mode, and ADV will not be sent, LED will be turned off, the purpose is to save power



consumption and avoid working in high power consumption mode for too long. If LPN have entered deep sleep, you need to press SW1 or SW2 defined in mesh\_lpn\_key\_map [] to wake up. After wakeup, LPN will start sending ADV again and waiting for the provision flow.

**Step 3** Provision and bind key process for the unconfigured LPN in the awake mode.

When Bind key is successful. After 3 seconds (LPN\_START\_REQUEST\_AFTER\_BIND\_MS), LPN will automatically reboot, and then set lpn\_provision\_ok to 1, and enter LPN mode, starting to send a friend request command every 2 seconds (FRI\_REQ\_TIMEOUT\_MS).

When the provision is successful, in order to reduce the processing of invalid network messages and reduce power consumption, LPN only receives messages sent from FN through friendship. If you want to receive ordinary network messages, initialize mesh\_lpn\_rx\_master\_key to 1.

**Step 4** When there is FN, it will automatically establish a friendship. Only when the establishment is successful (red light flashes 3 times), LPN can receive message.

After receiving the friend request, FN will automatically reply to the friend offer, and then establish the friendship. If the establishment is successful, the friend\_ship\_establish\_ok\_cb\_lpn () will be called back and the red light will flash 3 times (LGT\_CMD\_FRIEND\_SHIP\_OK). Then it starts sending friend POLL in a 2 second period (FRI\_POLL\_INTERVAL\_MS). After the FN receives the POLL, if there is a cache message that needs to be sent to the LPN, it will send the message to the LPN. The default maximum number of Cache message (network PDU) is 4 (2 ^ FN\_CACHE\_SIZE\_LOG).

If there is no FN responding to Friend Request, LPN will keep sending friend request in 2 second cycle.

**Step 5** The "mesh" window displays the LPN node and ONOFF operation.

CMD	sig	_mesh_ma	ster.ini	•	INI	BULKOUT	ASCII	
	, ,		mesh	bulk cr	nd debug			
LPN_g	et_le	vel						
LPN_9	et_011	off						
light	ness_	get_Pane	el					
								=
fw_in	fo_ge	t						
fw_in	to_ge	t_all						
fru di	strib	ution_ge	25 Fort oll					
fw di	strib	ution s	tart 0002					
fw di	strib	ution s	tart 02 0	3				
fw di	strib	ution st	tart 0001	_				
fw di	strib	ution s	top					
fw_di	strib	ution_d	etail_get					
fw_up	date_	get	_					
fw_up	date_	prepare						
fw_up	date_	start						
fw_up	date_	abort						
fw_up	date_	apply						
obj_t	ransf	er_get						
obj_t	ransf	er_star	t					
ODJ_t	ransi	er_abort	5					
	TOCK	transfe:	r_start					
lobj_c	lock	cransie.	-					
lobj_b	nfo g	et.						
sched	uler	get						
sched	acti	on get						
sched	acti	on_set_	off					
sched	_acti	on_set_	on					
sched	_acti	on_set_s	scenel					
time_	set							
time_	get							
time_	zone_	set						
time_	zone_	get						
time_	delta	_set						
time_	role	_get						
time	role	get						
scene	_stor	e						-
	-							
a3 ff	00 0	0 00 00	00 00 04	00 82 (	)5			

Figure 11.4: LPN\_get\_level

Log 🗌 fastbi	nd 2 retry Clear	Save Save	Adv Stop Scar	n rp_scan OTA Rx test
LPN get 1 <0000>18:26: <0001>18:26: <0002>18:26: <0002>18:26: <0003>18:26:	avel 54:519 [INFO]:(common) 54:520 [INFO]:(Basic)t 55:483 [INFO]:(log_win 55:881 [INFO]:(Basic)s	ExecCmd: a3 ff 00 00 0 he mesh access tx cmd 32)mesh_tx_reliable_st dr_src:0x0004,adr_dst:	0 00 00 00 04 00 82 is 0x0582 NULL op: op 0x0582 rsp_ma 0x0001,access rx cmd	05 x 1, rsp_cnt 0 is 0x882 : 82 08 ff 7f
Mes	ss:sse [INPO]:(cma_rsp	)Status ksp	: 04 00 01 00 82	US II /I
	Mesh           001         0002         On         Off         O         1           002         0004         On         Off         O         1	Nodes reliable	All On Off Svr 0 On Off	CInt Schedule Year Custom Custom Month

Figure 11.5: ONOFF operation on LPN

Because LPN does not receive the message whose destination address is Oxffff, it needs to send the command in unicast mode. If you have configured a group number for LPN, you can also send commands in group mode.

Also note that after clicking the "Nodes" button or reopening the "Mesh" window, the VC tool will put all the nodes offline, and then send the lightness get all (destination address is Oxffff) command to regain the node status, but There is no separate send command to the LPN node by unicast destination address, so you need to manually click the "LPN\_get\_level" command or click the ON / OFF command in the mesh window to display the online status, otherwise it is offline.

**Step 6** group operation is the same as normal node operation, please refer to "4.5.2 Group Control (ie Subscription Function Demo)"

**Step 7** The LPN detects that the FN is powered off and automatically searches for a new FN.

When the FN is powered off, LPN retry 8 times (FRI\_POLL\_RETRY\_MAX) POLL command, where the POLL interval is 170ms (FRI\_REC\_DELAY\_MS + FRI\_REC\_WIN\_MS), if the LPN still does not receive a reply from the FN, it is considered that the FN has been powered off, it will disconnect the friendship and callback friend\_ship\_disconnect\_cb\_lpn (), if you need to perform led flashing operation, please add it in the callback function, then resend the friend request to find a new friend node.

**Step 8** For now, one friend node of demo SDK establishes a friendship with two LPNs at the same time by default. If you need to modify it, just set MAX\_LPN\_NUM. The maximum value is 16.

When the LPN is powered off, the FN will detect for 10 seconds (LPN\_POLL\_TIMEOUT\_100MS). If the POLL command has not been received, the node is considered to be powered off. At this time, the FN will clear the LPN information.

**Step 9** Press the key to send the ALL ON / OFF command.

When the LPN is in the retention sleep mode, press SW2 (MESH\_LPN\_CMD\_KEY) to wake up the LPN, and then detect the key through suspend\_handle\_next\_poll\_interval ()-> mesh\_lpn\_wakeup\_key\_io\_get (), and then execute the test\_cmd\_wakeup\_lpn () function to alternately send ALL ON / OFF commands. LPN spontaneously sends the access layer command to use master security credentials to encryption by default.



### Step 10 Reset to Factory Setting

Long press the button SW1 (MESH\_LPN\_FACTORY\_RESET\_KEY) for 3 seconds (LONG\_PRESS\_TRIGGER\_MS) to trigger the factory reset.

# 11.5 app.c file introduction

### Customization of Adv Packet and Adv Response Packet

Please refer to Section 9.

### Configuration of fifo part

Please refer to Section 9.

### app\_event\_handler ():

Please refer to Section 9.

### main\_loop ():

Please refer to Section 9.

### user\_init():

Please refer to Section 9.

### proc\_ui():

This function mainly does some UI processing, such as button detection function, and the corresponding test code. When LPN is in non-GATT ota mode, it will send friend request to establish friend relationship. Press key SW2 (KEY\_), it will send ON/OFF command alternately; Long press key SW1 (KEY\_RESET) for 3 seconds (LONG\_PRESS\_TRIGGER\_MS) to trigger factory reset.

### test\_cmd\_wakeup\_lpn():

When pressing the corresponding command button (SW2 in current demo dongle) to wake up the program, the function "test\_cmd\_wakeup\_lpn()" will be executed. This function will send ON/OFF command. After sending the command, it will enter sleep. This function is only used for demonstration.

### mesh\_lpn\_state\_proc():

This function focuses on the processing of the working state of the LPN node:

- (1) Setting the LED flash when in LPN\_MODE\_NORMAL mode
- (2) LPN has been allocated and has not entered PM for 60 consecutive seconds, return to FRI\_ST\_REQUEST state.
- (3) The LPN is not configured and has not been networked within 60 seconds (LPN\_SCAN\_PROVISION\_START\_TIMEOUT of power-up and goes to sleep.
- (4) The LPN binds the appkey for 3 seconds and then enters the FRI\_ST\_REQUEST state.

### mesh\_lpn\_pm\_proc():

This function mainly manages the function of LPN node, user can handle some PM states in this function. For example, when the LPN node is networked and in connected state, always enable ENABLE\_SUSPEND\_MASK to save power consumption. When the user presses the key, it will not enter the PM demo for 4 seconds.

# 11.6 mesh\_lpn.c file introduction

### mesh\_lpn\_sleep\_prepare ():

This function handles the sleep processing function of LPN. lpn\_sleep.op indicates what command or event needs to go to sleep, and handles the subsequent actions of the event after waking up.

For example, when Ipn\_sleep.op is equal to CMD\_CTL\_POLL, it means that the POLL message has just been sent, and then you need to enter the retention sleep time of receive delay, and then wake up to enter the receive window, as shown below:

```
void mesh_lpn_sleep_prepare(u16 op)
        is_lpn_support_and_en && (BLS_LINK_STATE_CONN != blt_state)){
if(CMD_ST_NORMAL_TX != op){
ENABLE_SUSPEND_MASK;
rf_set_tx_rx_off();// disable tx rx in manual mode,must
class the processory.
    if(is
             CLEAR_ALL_RFIRQ_STATUS
             blt_soft_timer_delete(&mesh_lpn_poll_receive_timeout);
        }
         if(CMD_CTL_REQUEST == op){
             blt_soft_timer_update(&mesh_lpn_rcv_delay_wakeup, FRI_ESTABLISH_REC_DELAY_MS*1000);
         else if((CMD_CTL_POLL == op) || (CMD_CTL_SUBS_LIST_REMOVE == op) || (CMD_CTL_SUBS_LIST_ADD == op)){
             blt_soft_timer_update(&mesh_lpn_rcv_delay_wakeup, mesh_lpn_par.req.RecDelay*1000);
         else if(CMD_ST_NORMAL_TX == op){
             blt_soft_timer_update(&mesh_lpn_poll_md_wakeup, get_mesh_adv_interval());
         else if(CMD ST POLL MD == op){
             mesh_lpn_poll_md_pending
             u32 sleep_ms = FRI_POLL_DELAY_FOR_MD_MS;
#if MD_MESH_OTA_EN
             if(is_blob_chunk_transfer_ready()){
                  sleep_ms = 10; // waiting for chunk message from FN retransmit completed
             #endif
             blt_soft_timer_update(&mesh_lpn_poll_md_wakeup, sleep_ms * 1000);
             #if !WIN32
             blt_rxfifo.rptr = blt_rxfifo.wptr - 1;// clear buf, blt_rxfifo.rptr will ++ in lib
#endif
         else if(CMD_CTL_UPDATE == op){
             #if !WIN32
             blt_rxfifo.rptr = blt_rxfifo.wptr - 1;// clear buf, blt_rxfifo.rptr will ++ in lib
             #endif
             blt_soft_timer_delete(&mesh_lpn_poll_receive_timeout);
         else{//CMD_ST_SLEEP
         }
    else if(CMD ST NORMAL TX == op){
         blt_soft_timer_update(&mesh_lpn_poll_md_wakeup, get_mesh_adv_interval());
    }
}
```

Figure 11.6: mesh\_lpn\_sleep\_prepare

Other customized events are:

CMD\_ST\_SLEEP: After the interaction cycle of a friendship is completed, it enters the retention sleep mode of 2 seconds (friend request interval or poll interval), and then wakes up to enter the interaction of the next cycle.

CMD\_ST\_NORMAL\_TX: Sets the time to next enter mesh\_lpn\_poll\_md\_wakeup after an unsolicited mesh message.



CMD\_ST\_POLL\_MD: After sending the POLL, the FN reply MD (more data) is 1, then sleep for 100ms (FRI\_POLL\_DELAY\_FOR\_MD\_MS), wake up, and continue to send POLL to receive the remaining message.

### mesh\_feature\_set\_lpn():

Initialization of some configurable parameters of LPN. Mainly configure LPN\_POLL\_TIMEOUT\_100MS, the default value is 10 seconds.

Telink Semiconductor

# 12 Switch Project Introduction

# 12.1 Switch function introduction

The Switch mainly serves to add the function of remote control. The provisioner needs to add the switch node into the network, so that the buttons on the switch can be used to control nodes in the mesh network.

# 12.2 Switch principle

As a low power remote control node to control mesh, the switch must trigger pairing mode to implement provision. After the provisioner adds the switch to the mesh network, the switch can control nodes in the network.

# 12.3 app.c file introduction

# zimt semiconductor Customization of Adv packet and Adv response packet

Please refer to Section 9.

### Configuration of fifo part

Please refer to Section 9.

### app\_event\_handler ():

Please refer to Section 9.

### main\_loop ():

Please refer to Section 9.

### user\_init():

Please refer to Section 9.

### proc\_ui ():

The "proc\_ui" function configures key scan with the interval of 4ms. "mesh\_proc\_keyboard" is the interface function for key processing.

- When "keycode" is "RC\_KEY\_A\_ON", the switch will send the all\_on command to turn on all lights in the network.
- When "keycode" is "RC\_KEY\_A\_OFF", the switch will send the all\_off command to turn off all lights in the network.

### proc\_led():

First the configuration function "cfg\_led\_event" is used to configure LED blinking frequency and time. E.g. "cfg\_led\_event(LED\_EVENT\_FLASH\_1HZ\_4S)": configure LED to blink for 4s with the frequency of 1Hz. Then the function "proc\_led" serves to control the processing of LED blinking part.

### mesh\_switch\_init():

The "mesh\_switch\_init" contains setting of two parts:

- Code setting of wakeup IO of switch part, as well as enabling of the wakeup enable flag bit.
- IO setting of LED part. By default, LED pin is configured as GPIO mode with 100kohm pull-down resistor, and LED will blink four times after power on.

### proc\_rc\_ui\_suspend():

The processing function for sleep function part is "proc\_rc\_ui\_suspend()".

The processing of sleep part in current SDK is set as below: In advertising state, if MCU directly enters deep state without sending packets, after wakeup by key press, MCU will continue to enter deep state when packet transmission is finished. After pairing mode is triggered, MCU will enter deep state 30s later, and it won't enter deep state temporarily in link state.

For the processing flow of sleep part, please refer to section 12.7.

### kb\_scan\_key ():

"kb\_scan\_key" is the interface of matrix keyboard scan part. In current SDK, by default "numlock\_status" is set as 0 to indicate the numlock in full keyboard, while "read\_key" is the read key value.

# **12.4 Key Event Detection Process**

# 12.4.1 Code Block

```
void mesh_proc_keyboard ()
{
  static u32
            tick_key_pressed, tick_key_repeat;
  static u8
             kb_last[2];
  int det_key = kb_scan_key (0, 1);
  . . . . . .
  key change:pressed or released
  //
  if (det_key)
            {
     . . . . . .
     if(kb_event.cnt)
     {
        ..... // key was detected pressed. MCU run the code here one time for one press
  action.
     }
     else {
        ..... // key was released . MCU run the code here one time for one release action.
       rc_repeat_key = 0;
       key_released = 1;
     }
     . . . . . .
```

```
}
11
            no key change event
else if (kb_last[0])
{
   // long pressed // key was detected in a continuously pressed state. for each
   ↔ main_loop, MCU run the code here until the key is released.
   if (clock_time_exceed(tick_key_pressed, 2000000)) // long pressed // 2000000 is the
   ↔ threshold for long press detection
   {
       if ((kb_last[0] == RC_KEY_A_ON && kb_last[1] == RC_KEY_1_OFF) ||
          (kb_last[1] == RC_KEY_A_ON && kb_last[0] == RC_KEY_1_OFF))
       {
            if(SWITCH_MODE_NORMAL == switch_mode){ // long pressed event
               switch_mode_set(SWITCH_MODE_GATT);
            }
       }
   }
   . . . . . .
}else{
   ..... // no key was detected.
   key_released = 1;
}
. . . . . .
```

Introduction to key events:

}

- Key pressed: where the comment "key was detected pressed" indicates that a key press was detected.
- key was released: where the comment "key was released" indicates that a key release was detected.
- Long key press: where the comment "// long pressed // 2000000 is the threshold for long press detection" indicates that a long key press was detected.
- No key event: where the comment "no key was detected." indicates that a key press was detected.

Developers can add their own keystroke functionality to the above.

# 12.5 Switch Engineering Long Press Handling Logic

Determine the current key is pressed and use clock\_time\_exceed to start timing from the time the key is pressed, when the set time is reached, then trigger the processing of a long key press.

Example: Press RC\_KEY\_A\_ON and RC\_KEY\_1\_OFF for two seconds to trigger the switch to enter GATT mode.

```
else if (kb_last[0])
{
    // long pressed // key was detected in a continuously pressed state. for each main_loop,
    \leftrightarrow MCU run the code here until the key is released.
    if (clock_time_exceed(tick_key_pressed, 2000000)) // long pressed // 2000000 is the
    ↔ threshold for long press detection
    {
            if ((kb_last[0] == RC_KEY_A_ON && kb_last[1] == RC_KEY_1_OFF) ||
                (kb_last[1] == RC_KEY_A_ON && kb_last[0] == RC_KEY_1_OFF))
            {
                if(SWITCH_MODE_NORMAL == switch_mode){ // long pressed event
                    switch_mode_set(SWITCH_MODE_GATT);
                }
            }
    }
    . . . . . .
}
```

# 12.6 Example of Sending Commands Using the Soft\_timer Cycle

For an example of sending commands using soft\_timer cycle, please refer to this section "Example of Sending Commands Using the Soft\_timer Cycle".

# 12.7 Configuration of Switch Part

# 12.7.1 key table

#define KB_MAP_NORMAL	{\	
{RC_KEY_1_OFF,	RC_KEY_2_OFF,	RC_KEY_1_ON}, \
{ <i>RC_KEY_3_ON</i> ,	RC_KEY_3_OFF,	RC_KEY_2_ON}, \
{ <i>RC_KEY_4_ON</i> ,	RC_KEY_4_OFF,	RC_KEY_R}, \
{RC_KEY_A_OFF,	RC_KEY_A_ON,	RC_KEY_UP}, \
$\{RC\_KEY\_L,$	RC_KEY_DN,	RC_KEY_M},

User can configure the contents of actual "key\_table" according to the number of drive pins and scan pins which correspond to the number of columns and rows respectively.

# 12.7.2 Configure IOs for Drive Pins and Scan Pins

```
#define KB_DRIVE_PINS {GPI0_PB4, GPI0_PB5, GPI0_PB6}
#define KB_SCAN_PINS {GPI0_PE3, GPI0_PE2, GPI0_PE1, GPI0_PE0, GPI0_PD3}
```



Modify macros corresponding to "KB\_DRIVE\_PINS" and "KB\_SCAN\_PINS" according to actually used pins.

Then customize IO attributes for drive pins and scan pins, as shown below:

IO attribute setting corresponding to drive pins:

#define	PB4_FUNC	AS_GPIO
#define	PB5_FUNC	AS_GPIO
#define	PB6_FUNC	AS_GPIO
#define	PULL_WAKEUP_SRC_PB4	MATRIX_ROW_PULL
#define	PULL_WAKEUP_SRC_PB5	MATRIX_ROW_PULL
#define	PULL_WAKEUP_SRC_PB6	MATRIX_ROW_PULL
#define	PB4_INPUT_ENABLE	1
#define	PB5_INPUT_ENABLE	1
#define	PB6_INPUT_ENABLE	1

IO attribute setting corresponding to scan pins:

#define	PE3_FUNC	AS_GPIO		
#define	PE2_FUNC	AS_GPIO		
#define	PE1_FUNC	AS_GPIO		
#define	PE0_FUNC	AS_GPIC	)	
#define	PD3_FUNC	AS_GPIO		
#define	PULL_WAKEUP_SRC_	PD3		MATRIX_COL_PULL
#define	PULL_WAKEUP_SRC_	PE0		MATRIX_COL_PULL
#define	PULL_WAKEUP_SRC_	PE1		MATRIX_COL_PULL
#define	PULL_WAKEUP_SRC_	PE2		MATRIX_COL_PULL
#define	PULL_WAKEUP_SRC_	PE3		MATRIX_COL_PULL
#define	PE3_INPUT_ENABLE	E	1	
#define	PE2_INPUT_ENABLE	Ξ	1	
#define	PE1_INPUT_ENABLE	Ξ	1	
#define	PE0_INPUT_ENABLE	E	1	
#define	PD3_INPUT_ENABLE	E	1	

Suppose it's needed to modify "GPIO\_PB6" as "GPIO\_PB7" in drive pin part, the following parts should be modified accordingly.

1). #define PB6\_FUNC AS\_GPIO----->>>#define PB7\_FUNC AS\_GPIO
2). #define PULL\_WAKEUP\_SRC\_PB6 MATRIX\_ROW\_PULL----->>
#define PULL\_WAKEUP\_SRC\_PB7 MATRIX\_ROW\_PULL
3). #define PB6\_INPUT\_ENABLE 1 ----->>
#define PB7\_INPUT\_ENABLE 1

# 12.7.3 Turn on/off Light via Switch

According to different key values, different commands will be sent so as to process correspondingly.

Please refer to key processing program "mesh\_proc\_keyboard ()". Switch cannot control light nodes before it's added into the network. User can simultaneously press the "RC\_KEY\_A\_ON" and "RC\_KEY\_1\_OFF" button on the switch for more than 2 seconds to trigger pairing mode, and add the switch into the mesh network via the provisioner, so that all light nodes in this network can be turned on/off via the "RC\_KEY\_A\_ON" and "RC\_KE

# 12.8 Switch Operation

First follow the provision operations in section 10.4. Connect the switch with PC USB via Telink burning EVK (as shown in the figure below), and then burn the switch with corresponding firmware.



Figure 12.1: Switch Burning Connection

The switch buttons are shown as below:



Figure 12.2: Switch button

Power on the switch device. After power on, as a low power node, the switch must trigger pairing mode by simultaneously pressing the "RC\_KEY\_A\_ON" and "RC\_KEY\_1\_OFF" for more than 2s, so that it can be added into mesh network via the provisioner.

After the switch triggers pairing mode, its LED light will continuously blink four times to indicate it enters pairing mode. Power on the provisioner (if it's powered down), and wait for 15s or so. The LED on the switch will continuously blink four times to indicate the switch has already been added into the network.

Then the "RC\_KEY\_A\_ON" and "RC\_KEY\_A\_OFF" on the switch can be used to turn on/off all light nodes in the network.

Telink

T



# 12.9 Flow chart for Switch RC

Figure 12.3: Flow chart for switch RC

# 12.10 Flow chart for sleep processing



Figure 12.4: Flow chart for sleep processing

# 12.11 Modify the destination address of button send command

The 4 sets of buttons shown below support modifying the destination address of the command:



Figure 12.5: Switch button

- 1\_ON / 1\_OFF : The default function of the key is to send the onoff command with the destination address 0xC000.
- 2\_ON / 2\_OFF : The default function of the key is to send the onoff command with the destination address 0xC001.
- 3\_ON / 3\_OFF : The default function of the key is to send the onoff command with the destination address 0xC002.
- 4\_ON / 4\_OFF : The default function of the key is to send the onoff command with the destination address 0xC003.

If it is needed to modify the address of the key sending command, for example, to change the destination address of "1\_ON / 1\_OFF" from 0xC000 to 0xD000, you can send the publish set command through the INI command of the host computer, and the configuration example is as follows (the primary address of the remote control node, i.e., the node address of the example remote control node is 0x0025):

cfg\_pub\_set\_sig0025 =e8 ff 00 00 00 00 00 00 25 00 03 25 00 00 0D 00 00 ff 00 15 00 10

Change the destination address of  $2_OFF$  from 0xC001 to 0xD001, as shown in the following example:

cfg\_pub\_set\_sig0026 =e8 ff 00 00 00 00 00 00 00 25 00 03 26 00 01 0D 00 00 ff 00 15 00 10

The parameters of publish set are:

Field	Size (bits)	Description	Req.
Opcode	8	The message opcode	М
ElementAddress	16	Address of the element	М
PublishAddress	16	Value of the publish address	М
AppKeyIndex	12	Index of the application key	М
CredentialFlag	1	Value of the Friendship Credential Flag	М
RFU	3	Reserved for Future Use	М
PublishTTL	8	Default TTL value for the outgoing messages	М
PublishPeriod	8	Period for periodic status publishing	М
PublishRetransmitCount	3	Number of retransmissions for each published message	М
PublishRetransmitIntervalSteps	5	Number of 50-millisecond steps between retransmissions	М
ModelIdentifier	16 or 32	SIG Model ID or Vendor Model ID	М

Table 4.98: Config Model Publication Set message structure

### Figure 12.6: publication\_set\_parameters

For details, refer to "4.3.2.16 Config Model Publication Set" in mesh spec V1.1.

If you want to set up via mobile app, please refer to "Device Setting (Switch Device)" section in the chapter Android and iOS APP User Guide.

If you want to add another set of keys to configure the onoff publish address, such as "5\_ON / 5\_OFF", change the value of ELE\_CNT\_EVERY\_LIGHT to 5, and then configure the onoff publish address of the client model (primary address + 4).

For the description of ELE\_CNT\_EVERY\_LIGHT, please refer to this section "Definition of the number of elements of a node".

# 12.12 IV Index Update Mode for Switch

• Scenario 1: The Switch wakes up every 96 hours, sends a security beacon, then enters the scan adv state, scans for security beacons, and goes to sleep if any of the valid security beacons are scanned. If it is not received after timeout (SWITCH\_IV\_RCV\_WINDOW\_S), it also goes to sleep.

See switch\_trigger\_iv\_search\_mode(int force) for details on handling.

• Scenario 2: After powering down and then re-powering up, switch\_trigger\_iv\_search\_mode(1) is also called to perform the send and scan security beacon action of scenario 1.

# 13 Connect with a Platform

When connect with a certain platform, you need to configure some options, especially the provision method. Select by configuring MESH\_USER\_DEFINE\_MODE.

# 13.1 Normal Mode

# 13.1.1 No OOB provision mode

Configuration method:

Provision uses MESH\_NO\_OOB mode.

VENDOR\_ID is 0x0211

When testing, you can directly use our mobile app or host computer tools to provision.

# 13.1.2 Static OOB provision mode

# 13.1.2.1 Light Node Burn Static oob

When burning the firmware, just write 16 bytes directly in the flash fixed location FLASH\_ADR\_STATIC\_OOB (for example, 0x77800). If there is no burning (all 0xff), it means that no oob mode is used. If you need to modify the flash address, modify the macro FLASH\_ADR\_STATIC\_OOB.

# 13.1.2.2 Light node Device uuid

The device uuid is generated by user\_prov\_multi\_device\_uuid () -> uuid\_create\_by\_mac (tbl\_mac, prov\_para.device\_uuid) by default and can be obtained by the following methods:

- 1) Read prov\_para.device\_uuid through BDT tool
- 2) Obtain unprovision broadcast package through the general APP



Figure 13.1: Device uuid

3) Obtain unprovision broadcast package through TI sniffer



Figure 13.2: unprovision broadcast package

4) when the connection is successful, the device uuid will be printed out

```
<0027>23:03:16:292 [INFO]:(GattProv)CScanDlg::OnConnect:the device uuid is
: 1d 4d 89 b3 27 65 10 3d 8a 0b 29 e4 10 3a cd ab
```

Figure 13.3: Connection successful and print device uuid

5) In the case of gateway provision, when selecting a node obtained by scan, the device uuid will be printed.

```
<0005>00:02:49:081 [INFO]:(GattProv)CScanDlg::provision link:the device uuid is
: 1d 4d 89 b3 27 65 10 3d 8a 0b 29 e4 10 3a cd ab
```

Figure 13.4: Print device uuid at a scan node

# 13.1.2.3 User Customized uuid Method

If the user wants to customize the device uuid, set NORMAL\_MODE\_DEV\_UUID\_CUSTOMIZE\_EN to 1.

# 13.1.2.4 Provisioner static oob database

The Provisioner needs to fill in the oob data of the node to the oob database file (oob\_database.txt):

The data format of oob database is as follows:

device uuid(16byte) + oob(16byte)

Field analysis is as follows:

1d4d89b32765103d8a0b29e4103acdab: The device uuid of the provisioned node.

If there are multiple nodes, the line break can be increased, for example:



# 13.1.2.5 Test steps

Please follow the general process for provision and testing.

The test results of the static oob provision success, the screenshot using GATT master dongle mode is as follows:

		(		
Log fastbind 2 retry (	lear Save Save 🔽 H	ex Adv Stop	Scan rp_scan 0	TA Rx test
<0064>01:51:31:852 [ERR]:(com	non)element count is inval	lid: adr:0x0002, cr	nt:0	•
<0065>01:51:31:864 [ERR]:(com	non)obj_adr 0x0002, not fo	ound VC node info		
<0066>01:51:31:876 [INFO]:(Ga	tProv)SEND:provisioner se	and invite cmd		
: 00 00 🗾 stati	oob 🦯			
<0067>01:51:31:953 [INFO]:(Ga	tProv)RCV:the provision	apa data is		
: 01 02 01 00 00 01 00 00 00	Du oo oo			
- 00 00 00 01 00 00	CPICV/SEND. CHe provision	Start IS		
<0069>01:51:31:982 [INFO]: (Ga	tProv)SEND:provisioner se	and pubkey is		
:		ing public, 12		
67 16 17 4d 14 b7 78 10 fa	3e 02 08 66 ac ab f0 c1	a8 b2 19 a8 22 Oc	1f a1 b7 ce 81 92 c	8 f1 43
47 7b e9 9d 29 40 47 b4 ae	df 80 69 26 2c 05 9b 93	9c 37 03 b9 dd 72	d6 5d fb 2f 90 3f d	12 7f c2
<0070>01:51:32:115 [INFO]:(Ga	tProv)RCV:the pubkey of t	he device is		
:				
79 4e 33 c1 02 15 7d 84 d4	4b c3 d1 c6 7b 86 43 83	f3 6c d1 cd 2e f4	7a 59 11 28 44 5c 1	.b 16 d1
b0 cb 6d 6e 0b b2 c0 dc ff	2e e2 f1 a8 47 1a 91 41	57 fl bc 24 00 9e	3a 74 af df fe 8a 6	3b d0 6d
<0071>01:51:32:130 [INFO]:(Ga	ttProv)SEND:the provisione	er's comfirm is		
: dd 6/ er c/ b2 44 30 8e 3r	50 08 D2 18 II 4a a/	mfirm is		
- d3 41 7f 3b 49 38 8b 5c af	d as 00 cl eb 15 0s	JMIIIM 19		
<0073>01.51.33.802 [INFO]. (Ga	tProv)SEND the provisions	r's random is		
: fd 52 79 a4 a4 cb f5 f5 20	47 47 71 9c 9c c3 ee			
<0074>01:51:33:873 [INFO]:(Ga	tProv)RCV:the device's ra	andom is		
: e2 c5 0f 5d 53 54 65 b6 00 1	o1 90 4d fe 26 4f fb			
<0075>01:51:33:882 [INFO]:(Ga	tProv) the device comfirm	check is success		
<0076>01:51:33:892 [INFO]:(Ga	tProv)SEND:the provisione	er's device info is	3	
: 55 26 26 4d 77 77 9f c9 c9	E3 1b 1b 45 70 70 97 00 00	0 00 11 22 33 44 02	2 00	
<0077>01:51:33:902 [INFO]:(Ga	tProv) the node's dev key:			
: b4 86 f6 48 86 b5 11 2e ae	c ff 17 df de b6 6e			
<0078>01:51:34:032 [INFO]:(Ga	ttprov) RCV:rcv the provisi	on completet cmd,p	provision success	
<0080>01:51:34:049 [INFO]: (Ba	sic)filter send and is 1:	00 01		
<0081>01:51:34:085 [INFO]: (Ba	sic)filter send cmd is 1:	ff ff		
<0082>01:51:34:153 [INFO]: (Ba	sic) the filter rsp is 0: (	0 00 27 14		
<0083>01:51:34:163 [INFO]: (Ba	sic)mesh rc data cfg gatt	dec suc		E
<0084>01:51:34:173 [INFO]:(lo	win32) white list			
<0085>01:51:34:181 [INFO]:(lo	_win32)GATT addr 0x0002,	filter list status	s, ListSize is: O	
<0086>01:51:34:193 [INFO]:(Ba	sic)the filter rsp is 0: (	0 01 23 dd		
<0087>01:51:34:202 [INFO]:(Ba	sic)mesh_rc_data_cfg_gatt	dec suc		
<0088>01:51:34:210 [INFO]:(10	g_win32) white list			Ψ.
·				+
ALL Chn_set	connect input_db	Path:	search_fi	le Mesh
UART	USB output db	GATE RESET Me	sh ota Gate ota Pro	v Close

Figure 13.5: static oob provision success

Capability data, please refer to the following chapters of spec for more detailed analysis.

82







Data analysis is as follows:

01 02 01 00 00 01 00 00 00 00 00 00

01: Provisioning PDU Type, 01 indicates Provisioning Capabilities

02: Number of Elements

01 00: Algorithms

00: Public Key Type

01: Static OOB Type

00: Output OOB Size

00 00: Output OOB Action

00: input OOB Size

00 00: Input OOB Action

# 13.2 Ali Tmall Genies Platform

### 13.2.1 Configuration





Provision uses MESH\_STATIC\_OOB mode.

VENDOR\_ID is 0x01A8.



# 13.2.2 Apply tri-truple from Ali

The default tri-truple information is null(con\_sec\_data[16] is null), code is in user\_ali.c, shown as follow-ing:

	-		
User ali.c		00025:	<pre>#elif(MESH USER DEFINE MODE == MESH SPIRIT ENABLE)</pre>
		00026:	
		00007.	Hacfine and came departer munder pap and 0// in Ali dama any iran mentuca this three
🎇 include "user_al: 🔺		00027:	#deline Als_SAFE_CERTIFY_THREE_PAR_EN 0// In An demo environment, use this three
include app_hea		00028:	<b>#if</b> !AIS SAFE CERTIFY THREE PAR EN
<pre>include "//pi include "vendor r</pre>		00029:	u32 con product id=0x00000002;// little endiness
include "fast_pro		00030:	const u8 con_mac_address[6]={0x9e,0x16,0x11,0x07,0xda,0x78};//small endiness
🛱 include "//pi		00031:	<b>#if</b> 0 // need to open it to make the init three para enable
num2char if (MESH USER DEF)	Ð	00032:	u8 con sec data[16]={ 0x04,0x6e,0x68,0x11,0x27,0xed,0xe6,0x70,
STATIC_ASSERT		00033:	0x94,0x44,0x18,0xdd,0xb1,0xb1,0x7b,0xdc};
con_product		00034:	#else
con_mac_add		00035:	u8 con_sec_data[16];
SIZE_CON_SE		00036:	#endif
# AIS_SAFE_CE		00037:	#else

Figure 13.8: Apply tri-truple

So user need to apply tri-truple from Ali:

(Total 24byte: PID(4byte, smaller end) + MAC(6byte, bigger end) + secret data(16 byte)),

Then burn to FLASH\_ADR\_THREE\_PARA\_ADR(0x78000), and program will read parameter in 0x78000 automatically.

# 13.2.3 Use SDK Default tri-truple

User can use SDK default tri-truple for demo, Open it as follows: (Enable the preset con\_sec\_data [] in the user\_ali.c file)

	-	· • • • • •	
User ali.c		00025:	<pre>#elif(MESH_USER_DEFINE_MODE == MESH_SPIRIT_ENABLE)</pre>
_		00026:	
🗱 include "user_al: 🔺		00027:	#define AIS SAFE CERTIFY THREE PAR EN 0// in Ali demo environment, use this three
<pre># include "app_heal include "//pj</pre>		00028:	#if !AIS SAFE CERTIFY THREE PAR EN
<pre>include "//pi include "wonder ;</pre>		00029:	u32 con product $id=0x\overline{0}0000002; //$ little endiness
include "fast_pr		00030:	const us con mac address [6] = $\{0x9e, 0x16, 0x11, 0x07, 0xda, 0x78\}$ ; // small endiness
🛱 include "//p) 🛱 include "//p)		00031:	<b>#if</b> 0 // need to open it to make the init three para enable
num2char # if (MESH USER DEF)	•	00032:	u8 con sec data[16]={ 0x04,0x6e,0x68,0x11,0x27,0xed,0xe6,0x70,
STATIC_ASSERT		00033:	0x94,0x44,0x18,0xdd,0xb1,0xb1,0x7b,0xdc};
con_product		00034:	#else
con_mac_add	•	00035:	u8 con sec data[16];
SIZE_CON_SE		00036:	#endif
AIS_SAFE_CE		00037:	#else



However, because there is only one default tri-truple, it can only be used for a single node demonstration when testing.

# 13.2.4 Provision via Tmall Genie

Provision can be done directly through Tmall Genie's voice commands.



## 13.2.5 Provision via Firmware

Because Tmall Genie mode only supports the static oob mode by default, oob and tri-truple have a binding relationship, so the firmware needs to know the information of the tri-truple to provision. So you need to add the tri-truple to this file of the firmware:

SIG\_MESH\_Release\_Vxxx -> tools -> telink-ble-phone -> three\_para.txt

SDK default tri-truple information has been added to this file. When adding new tri-truple information, just refer to this format.



Then refer to Provision part in chapter 4 for detail steps.

# 13.2.6 Dual Modes of static oob and no oob

Tmall Genie mode, the node end only responds to the static oob mode by default. If you need to support no oob mode at the same time, change ENABLE\_NO\_OOB\_IN\_STATIC\_OOB from 0 to 1.

# 13.3 Xiaomi Xiao'ai Platform

# 13.3.1 Configuration

		- is - is	
Mesh config.h	00116:	// mesh config (us	er can config)
	00117:	#define MESH NORMAL MODE	0
# IS_VC_PROJECT	00118:	#define MESH CLOUD ENABLE	1
PROXY_GATT_WITH_	00119:	#define MESH SPIRIT ENABLE	2// use this mode should burn in the para in
🗱 if WIN32	00120:	#define <b>MESH AES ENABLE</b>	3
FAST_PROVISION	00121:	#define <b>MESH GN ENABLE</b>	4
FAST_PROVISION	00122:	#define <b>MESH MI ENABLE</b>	5
<pre># endif # ATT_TAB_SWITCH_E</pre>	00123:	#define MESH MI SPIRIT ENABLE	6 // dual vendor
<pre># if WIN32 # TESTCASE FLAG</pre>	00124:	#if PROJECT MESH PRO	
TTS_TEST_EN	00125:	#define MESH USER DEFINE MODE	MESH NORMAL MODE // must normal
TESTCASE_FLAG_	00126:	#elif PROJECT SPIRIT LPN	//
if (0 == TESTC	00127:	#define MESH USER DEFINE MODE	MESH SPIRIT ENABLE // must sprit
else	00128:	#else	
endif	00129:	#define MESH USER DEFINE MODE	MESH MI ENABLE
B endif	00130:	#endif	
<pre>## ifPROJECT_MESI</pre>	00101		





Provision uses Xiaomi defined mesh provision mode

VENDOR\_ID is 0x038F.

# 13.3.2 Certification Data Setting

- R & D test mode: The default certification data of the SDK is used, dev\_cert\_pri [], so it can only be used in the single node test mode. There are 6 certificates by default. Because these certificates are public, if they have been used by others, conflict will happen. So it is recommended for users to use the certificate they applied for and then testing it in production mode.
- Production mode: When production or multi-node network testing is required, flash writing is required. The steps are as follows:

#define	DEMO	CERT	TYPE0	0		
#define	DEMO	CERT	TYPE1	1		
#define	DEMO	CERT	TYPE2	2		
#define	DEMO	CERT	TYPE3	3		
#define	DEMO	CERT	TYPE4	4		
#define	DEMO	CERT	TYPE5	5		
	-		-			
#define	DEMO	CERT	TYPE	DEMO_CERT_TYPE1		
			τO <sup>γ</sup>			
Figure 13.12: Apply certificate						

**Step 1** Define MI\_CER\_MODE as FLASH\_CER\_MODE, generate firmware

Step 2 Burn certification data to DEV\_SK\_FLASH\_ADR(0x7f000)

# 13.3.3 Provision Test 🔨

After burning firmware, please make sure that the MAC address of the flash (512K flash is at 0x76000 and 1M flash is at 0xFF000) is empty (that is, all 0xff), otherwise it will prompt "Unable to connect" or "Provision failure".

When firmware is powered on for the first time, it will extract the MAC from the certificate, write it to the MAC address sector, and generate some necessary parameters.

After the node is powered on, it can be directly provisioned through Xiao'ai (Voice instruction example: "Xiao'ai, add device").

# 13.4 Dual Vendor Mode (Tmall Genies and Xiaomi Xiaoai)

# 13.4.1 Function Introduction

When the node leaves the factory, it will send adv. packets in Ali and Xiaomi modes at the same time, which can be provisioned by either Tmall Genies or Xiao'ai. Once provision is successful, subsequent functions are



performed according to the selected mode, including parameters such as the vendor model and the transmit count. For parameter switching functions, see mesh\_ais\_global\_var\_set ().

The production test function in unprovisioned state of is performed according to the Xiaomi mode.

After provision is successfully, you can execute the kick light command or restore the factory settings to return to the dual vendor mode and select again.

Which mode you are currently in can be viewed through the provision\_mag.dual\_vendor\_st variable.

# 13.4.2 Configuration

Mesh_config.h	00116: //	' mesh config (	(user can config)
	00117: #d	lefine MESH NORMAL MODE	0
# IS_VC_PROJECT	00118: #d	lefine MESH CLOUD ENABLE	1
# IS_VC_PROJECT_MAX # PROXY_GATT_WITH_J	00119: <b>#</b> d	lefine MESH SPIRIT ENABLE	2// use this mode should burn in the pa
<pre># if WIN32 # PROTECT MESH</pre>	00120: #d	lefine MESH AES ENABLE	3
FAST_PROVISION	00121: #d	lefine MESH GN ENABLE	4
FAST_PROVISION	00122: #d	lefine <b>MESH MI ENABLE</b>	5
🗱 endif 🗱 ATT TAB SWITCH EI	00123: #d	lefine MESH MI SPIRIT ENABL	E 6 // dual vendor
if WIN32	00124: <b>#i</b>	f PROJECT MESH PRO	
PTS_TEST_EN	00125: <b>#</b> d	define MESH USER DEFINE MOD	E MESH NORMAL MODE // must normal
🗱 else 🙀 TESTCASE_FLAG_	00126: #e	lif PROJECT SPIRIT LPN	
if (0 == TESTC	00127: #d	define MESH USER DEFINE MOD	E MESH SPIRIT ENABLE // must sprit
dise	00128 <b>#e</b>	alse	
# PTS_TEST_EN	00120: #d	efine MESH USER DEFINE MOD	MESH MT SPIRIT ENABLE
DEBUG_EVB_EN	00130 • #0	andif	A HEOL MI_OTINIT_ENABLE
ifPROJECT_MESI	00131.		

Figure 13.13: Provision method

Provision method and parameter configuration, please check 13.2 and 13.3.



# 14 Factory Reset

# 14.1 8258\_mesh/8269\_mesh Node

# 14.1.1 Function Introduction

### Factory reset reset execution action, please refer to factory\_reset\_handle () or kick\_out():

```
{
irq_disable();
factory_reset(); // Flash erasing
#if DUAL_MODE_WITH_TLK_MESH_EN
UI_resotre_TLK_4K_with_check();
#endif
show_ota_result(OTA_SUCCESS); // LED indication
start_reboot(); // MCU reboot
}
```

If the customer has modified the flash map, or used the customer flash section (the default is 0x7a000 - 0x7f000, and erase is not performed on the area by default), you need to reconfirm the factory\_reset () function to confirm whether there are sector errors or missing erases.

### Power-up sequence detection function factory\_reset\_cnt\_check ():

- (a) After power-on, the power-on sequence will not be detected until after VALID\_POWER\_ON\_TIME\_US (default 50ms). Because it is necessary to filter the pulse voltage generated when some power supplies are powered on.
- (b) clear\_st is 4:

First reset\_cnt\_get\_idx () to obtain the sequence before power off, if it is an odd number, it means that the previous power-on sequence does not meet expectations, directly clear the power-on sequence and restart counting. If it is even, then it is as expected.

Then, check whether the stored power-on sequence value satisfies the condition that triggers a factory reset, and if it does, execute a factory reset. If not, immediately add 1 to the sequence value.

- (c) clear\_st is 3, get the power-on sequence value by get\_reset\_cnt (), get the timing time, and start the timing of the first phase.
- (d) clear\_st is 2, the first phase timing meets the requirements, get the power-on sequence value through get\_reset\_cnt (), get the timing time, and start the second phase timing.
- (e) clear\_st is 1. If the second phase is over and power has not been turned off, it means that the power-on time is not as expected, and the power-on sequence is directly cleared.

# 14.1.2 Default trigger action

Low power node, e.g., LPN, cannot be triggered with booting sequence, because LPN cannot count time and determine timing sequence. User can trigger this with button-pressing, call functions described in 14.1.1.



User can follow the steps below to reset a SIG\_mesh module (non-LPN) to factory default configuration:

Step 1 Power on the SIG\_mesh module, and wait for more than 30s (equivalent to initial power on).

This operation will erase previous power on record, and ensure it conforms to the timing sequence requirement of initial power on rather than any power-on sequence defined by "factory\_reset\_serials[]".

**Step 2** Power cycle the SIG\_mesh module three times. Note: After each power on, the module must be powered down within the range of 0~3s. This operation conforms to the requirement of former three power-on sequences in the "factory\_reset\_serials[]".

**Step 3** Power cycle the SIG\_mesh module two times. Note: After each power on, the module must be powered down within the range of 3~30s. This operation conforms to the requirement of latter two power-on sequences in the "factory\_reset\_serials[]".

**Step 4** Power cycle the SIG\_mesh module.

The "factory\_reset\_handle()" in the "user\_init()" will detect and get the result that previous five power-on sequences match with the trigger requirement of "Factory Reset". The red LED light on the module will blink for 8s with the frequency 1Hz to indicate factory reset success.

### Note:

Since some module may need some time to finish power down, to ensure its MCU is stopped completely, it's needed to wait for a duration (e.g. 2s, depend on module) after power-down operation.

Power-on timing sequence is defined by the array below:

# 14.1.3 Method to modify power-on sequence

To modify power-on sequence, it's only needed to modify/add/delete sequences in the array "factory\_reset\_serials[]" correspondingly, as long as the requirements below are met:

- The left value should be smaller than the right value.
- When it's needed to add/delete sequences, since one sequence corresponds to two values, multiple of two values must be added/deleted correspondingly.

E.g. To modify power-on sequence as six sequences, the following method can be followed.

# 14.1.4 The function of the previous mesh network can be restored after the reset action is triggered

The above 1 ~3 mode is the normal mode for restoring factory settings.

After trigger factory reset, if it does not re-configure the network within a certain period of time (such as 30 seconds), some users may hope to automatically restore the previous network information. The SDK provides 2 APIs needed to implement this function:

- mesh\_reset\_network (u8 provision\_enable): Restore the network information in the ram to the default network state. Since the parameter provision\_enable is 1, the device will send unprovision beacon and pb\_adv, and the device can be reconfigured. At this time, only the RAM data is restored, but the flash information has not changed.
- mesh\_revert\_network (): Reload network information from flash.

Implementation: When the user triggers the factory reset action and enters the kick\_out function, it directly calls mesh\_reset\_network (1) to restore the network information in the ram to the default network, without calling factory\_reset () and start\_reboot ().If the user wants to restore the network, directly call mesh\_revert\_network () to reload the mesh information from the flash.

# 14.2 Gateway Node + Host Computer

To reset the gateway, you need to perform two actions: one is to delete the mesh\_database.json, and the other is to clear the parameter area of the flash of the gateway dongle (for example, by resetting the device 5 times). Because there are two actions that need to be performed, in order to facilitate the operation, a "GATE\_RESET" button is added to the firmware. After executing this button, the above two things will be performed. After the gateway dongle clears its own flash parameter area, it will automatically call start\_reboot () for a soft restart.

Note: this button just resets the gateway itself and will not send commands to reset other nodes.

```
ত Telink
```

												×	
							2						
	log 🗌 f	astbind	2	retry Cl	lear Save.	Save	🔽 Hex 🗌 Adv	Stop	Scan	rp_scan	OTA R	x test	
	<pre>COUDDATS:US:10:43:12:40 [INFO]: (GALEWAT, GALEWAT_CHD_RASET: 49 ET Dining, and the gateway will reset COUD2:15:04:31:043 [INFO]: (common) wait the gateway TINIED Shining, and the gateway will reset COUD2:15:16:22:641 [INFO]: (common) wait the gateway TINIED Shining, and the gateway will reset COUD2:15:16:22:641 [INFO]: (GATEWAY)HCI_GATEWAY_CMD_GATEWAY_CMD_GET_PRO_SELF_STS : e9 ff 0c COUD4:19:43:52:645 [INFO]: (GATEWAY)HCI_GATEWAY_CMD_GET_PRO_SELF_STS : e9 ff 0c COUD5:14:35:26:65 [INFO]: (GATEWAY)HCI_GATEWAY_CMD_SIND_ELE_CMT : 91 8c 01 COUD5:14:17:991 [INFO]: (GATEWAY)HCI_GATEWAY_CMD_GET_PRO_SELF_STS : e9 ff 0c COUD5:19:44:17:991 [INFO]: (GATEWAY)HCI_GATEWAY_CMD_GET_PRO_SELF_STS : e9 ff 0c COUD5:19:44:17:991 [INFO]: (GATEWAY)HCI_GATEWAY_CMD_GET_PRO_SELF_STS : e9 ff 0c COUD5:19:44:17:991 [INFO]: (GATEWAY)HCI_GATEWAY_CMD_DET_SERSP unprovisioned : 91 8b 00 30 84 00 7d 3b 84 00 75 3c 84 00 5a 27 00 00 10 00 00 05 8b 80 100 00 00 cOUD5:144:17:997 [INFO]: (GATEWAY)HCI_GATEWAY_CMD_SIND_ELE_CMT : 91 8c 01</pre>												
*		ALL	-	chn_set	connect	input_db	Path	:		searc	h_file	Mesh	
			•	UART	USB	output_db	GATE	_RESET M	lesh_ota	Gate_ota	Prov	Close	

Figure 14.1: gateway reset

The send command E9 FF 02, E9 FF represents the macro HCI\_CMD\_GATEWAY\_CTL, and the related processing flow is in the function app\_hci\_cmd\_from\_usb\_handle.

```
else if (HCI_CMD_GATEWAY_CTL == type){
    #if IS_VC_PROJECT
    ret = fifo_push_vc_cmd2dongle_usb(buff, n);
    #else
    #if GATEWAY_ENABLE
    ret = gateway_cmd_from_host_ctl(hci_data, hci_data_len);
    #endif
    #endif
```
}

An opcode of O2 indicates HCI\_GATEWAY\_CMD\_RESET, and the related processing flow is in the function gateway\_cmd\_from\_host\_ctl.

```
else if (op_code == HCI_GATEWAY_CMD_RESET){
    factory_reset();
    light_ev_with_sleep(4, 100*1000); //10hz for about the 1s
    start_reboot();
}
```

# 14.3 GATT master dongle + Host Computer

There is no storage parameter in the flash of master dongle, so just delete the mesh\_database.json file of the upper computer, and then reopen the upper computer tool.

# 14.4 LPN Node

Low-power nodes cannot be reset by powering on 5 times, because when it is in sleep state, it will take some time to consume power after power off. Therefore, LPN nodes generally need to press keys and other methods to trigger and call the functions in the first section of this chapter.

DEMO LPN: long pressed SW1 key (MESH\_LPN\_FACTORY\_RESET\_KEY) for more than three seconds (LONG\_PRESS\_TRIGGER\_MS) .it will flash 4 times, indicating that the reset is successful..

# 14.5 Switch Node

Low-power nodes cannot be reset by 5 power-ups. Need to press keys, press and hold the key combination: RC\_KEY\_A\_ON + RC\_KEY\_4\_OFF for more than three seconds, it will flash 4 times, indicating that the reset is successful.

# 15 Fast bind Mode (PROVISION\_FLOW\_SIMPLE\_EN Mode)

# **15.1 Function Introduction**

This mode is a non-standard mode and is used to speed up the provision. The improvements are as follows: After the implementation of the Spec definition provision flow, the unicast address, netkey, and other information are assigned, you need to send get composition data, app key add in order, and perform key bind on each model in the composition data obtained. This is not so complicated, in most cases a device will only have an APPkey. So we defined this Fast bind pattern.

The Fast bind mode is mainly to optimize the key bind part. When the provisioner sends the app key add, it no longer sends the key bind command. After the node receives the app key add, it performs the key bind action on all of its own models. For details, see the code enclosed in the PROVISION\_FLOW\_SIMPLE\_EN macro.

In addition, in order to further simplify provision, the demo app does not need to execute the get composition data command, it directly queries the database to obtain all information of the corresponding composition data through the PID in the device UUID obtained by provision. The following figure shows the function of firmware to write PID to device uuid.

```
void set_dev_uuid_for_simple_flow(u8 *p_uuid)
{
    simple_flow_dev_uuid_t *p_dev_uuid = (simple_flow_dev_uuid_t *)p_uuid;
    memcpy(&p_dev_uuid->cps_head, &gp_page0->head, sizeof(p_dev_uuid->cps_head));
    memcpy(p_dev_uuid->mac, tbl_mac, sizeof(p_dev_uuid->mac));
    // set uuid
}
Figure 15.1: Write PID to device uuid
```

If the customer does not want to obtain the composition data by querying the database, he/she can also modify the flow of the app and add the get composition data command.

# 15.2 Configuration

Node Firmware: set PROVISION\_FLOW\_SIMPLE\_EN to 1.

# 15.3 Function Demonstration

#### 15.3.1 Firmware Configuration

B Telink sig_mesh Found V3.1	- 2-2- mm	and max [111]			×
CMD tl_node_gateway.ini  INI BULK	OUT ASCII	2 retry Clear Save Save	Hex Adv Stop	Scan rp_scan	OTA Rx test

#### Figure 15.2: Firmware Configuration

#### 15.3.2 APP Interface Configuration

Enable this option in the app:

Homepage – setting – settings – Enable Private Mode (Default Bound)

Telink Semiconductor

# **16 Private Fast provision Function**

# 16.1 Function Introduction

Remote provision is done node by node, when the network is big, the provision time is still too long, to solve this problem, we provide this private fast provision function, i.e., in the default key network, add vendor commands, e.g., VD\_MESH\_RESET\_NETWORK, and send network key, app key, iv index to target address Oxffff(send only once, and the whole network can receive it), then assign unicast address one by one according to mac. In this way, user can provision nodes within multiple hops. Device keys are generated based on mac address according to a certain rule, no need to assign by mesh commands.

User can also add un-provisioned new devices into an existing mesh network by fast provision way.

# 16.2 Configuration

Set FAST\_PROVISION\_ENABLE to 1. Compile 8258\_mesh project, download to 3(more than 1)8258 dongles.

GATT master dongle mode and APP support fast provision function by default. The gateway V3.3.4 and later versions support this feature, which is disabled by default.

# 16.3 Function Demo

The following demo shows how to add multiple (more than 2) 8258 nodes into mesh network at the same time for Gateway.

- 1) Power up the 8258 mesh node.
- 2) Set FAST\_PROVISION\_ENABLE to 1, compile the 8258\_mesh\_gw compilation option to get 8258\_mesh\_gw.bin and burn it to the 8258 Dongle, that is, Gateway.
- 3) Plug the Gateway into the USB port, open "SIG\_MESH\_TOOL" and select tl\_node\_gateway.ini, the title bar will show Found indicating that the gateway device is found (Provisioner). The tool will automatically get the uuid and mac address of the gateway, and the format of the commands sent is detailed in the "Provisioner (Gateway) Project Introduction" in the "Provisioner Lighting" section.

T

Felink sig_mesh Found V4.1.0.0	-	□ ×
CHD tl_node_gateway.ini INI BULKOUT ASCII	V Log T AutoSaveLog 2 retry Clear Save Save V Hex Adv Stop Scan rp_scan ex_scan of	A Rx test
<pre>LPN_get_get_ LPN_get_onoff lightness_get_Panel Note:retry count field of LPN distrib start is change LPN_fw_distrib_ota_start_04 </pre>		*
sched_action_get sched_action_set_off sched_action_set_on sched_action_set_scenel		
time_set time_get time_set	ALL chn_set connect input_db Path: OpenFile	e Mesh
ļ	directed UART USB output_db GwReset GwMeshOta GwOtaSelf Pro	V Close

#### Figure 16.1: Normal connection

- 4) Click the "Prov" button at the bottom right corner to enter the provision interface, select Fast prov mode, set the network parameters and then click SetPro Internal to configure the provision data of Gateway. For the command format of "Prov" and "SetPro Internal", please refer to "Provisioner (Gateway) Project Introduction", "Provisioner Lighting".
- 5) Add the appkey of the Gateway: since there is no binding process in the fast provision, if there is no app key added to the Gateway before, you need to add the app key to the Gateway, the corresponding command format is:

HCI\_CMD\_GATEWAY\_CMD + netkey index + appkey index + retry cnt + response max + destination + op + par.

that is: e8 ff + 0x0000 + 0x0000 + 0x00 + 0x01 + gateway address + 0x00 + netkey appkey index(3 bytes) + appkey(16 bytes).

provision	×
Fast prov mode         1           SetPro_internal         2           network_key         b3 12 4d c8 43 bb 8b a6 1f 03 5a 7d 09 38 25 1f	Static           apk_idx         00 00           app_key         60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48           bind_all
key_index 00 00 iv_index 11 22 33 44	
iv_update_flag 0 unicast_adr 0d 00	
Provision 3	
filter_operation	1
filter_type white_list - filter_data 01 00 ff ff	
SetFilter Add_mac	
RM_mac	

Figure 16.2: Click in order

6) Click the "Provision" button to start the network provision.

The corresponding command format of Provision control is HCI\_CMD\_GATEWAY\_CTL + HCI\_GATEWAY\_CMD\_FAST\_PROV\_ + pid(2 bytes) + new device address(2 bytes).

That is: e9 ff + 0x17 + pid + new device address.

Note: The device type to be added can be specified by PID. If all device types are to be added, PID is set to Oxffff.

7) The gateway reports the address assigned to the device during fast provision to sig\_mesh\_tool.exe of PC. The report format is:

TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_CMD\_SEND\_NODE\_INFO + VC\_node\_info\_t.

That is: 0x91 + 0x81 + VC\_node\_info\_t. For details of the VC\_node\_info\_t format, please refer to the "Provisioner Lighting" section in the "Introduction to the Provisioner (Gateway) Project".

Note: Since the valid parameter length of the vendor message is only 8 bytes, the device only returns 6 bytes of mac and 2 bytes of PID during the gateway scanning, and the gateway side will get the number of elements of the device according to the PID, please refer to mesh\_fast\_prov\_get\_ele\_cnt\_callback(u16 pid) for details.

8) After fast provision is completed, you will see all nodes are blinking 3 times. The gateway reports to sig\_mesh\_tool.exe of PC with successful binding event. The report format is:

TSCRIPT\_GATEWAY\_DIR\_RSP + MESH\_KEYBIND\_EVE\_SUC + event.

That is: 0x91 + 0x8a + 0x01.

9) Click the "Mesh" button to enter the mesh window to turn on/off the lights.

# **17** Private online status function demo

# **17.1 Function Introduction**

Currently, the online and offline monitoring mechanism provided by the SIG mesh spec can be implemented through the heartbeat and publish mechanisms. The real-time monitoring of status such as onoff of nodes can be implemented through the publish mechanism.

When both online and offline monitoring and real-time monitoring of status are required, a publish mechanism is required. However, the publish mechanism has the following limitations:

- Publish messages generally need to set relay, so there will be more packets on air.
- The publish period cannot be set too short (it usually takes tens of seconds or longer), otherwise there will be too many packets on air, affecting normal control.
- Sometimes it takes several publish status messages to include all the statuses that need to be reported. At this time, there will be more packets on air.

Therefore, we have added an online status mechanism, the purpose of which is to achieve fast and effective online and offline detection, and to report important status of nodes, while also effectively reducing data packets in the network.

# 17.2 Configuration

Change the ONLINE\_STATUS\_EN of app\_config.h of both mesh and mesh\_provision project from 0 to 1.

# 17.3 Packet Format

The online status data packet is sent by adv of ADV NON CONN IND, and the type of the payload is customized MESH\_ADV\_TYPE\_ONLINE\_ST (0x62), as shown in the figure below.

Time (us)	Channel	Access Address	Adv DDU Type		Adv	PDU Hea	der	AdvA	AdvData	CBC	RSSI	ECE
+319991	Channel	Access Address	Advebbilighe	Type	TxAdd	RxAdd	PDU-Length	AUVA	1E 62 0F 07 31 56 F2 03 47 DA 76 D7 23 28 CC 80	CRC	(dBm)	I Co
=1599997	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0xFFFF82580002	64 13 B8 41 57 93 BA 1C 6B 9E DD 91 9F AB 4A	0x000080	-38	OK
Time (ue)					۸dv	DDII Hoa	der		AduData		DSSI	
1000010	Channel	Access Address	Adv PDU Type	T	Tulda	Destrola	DDTL Taranh	AdvA		CRC	(dBm)	FCS
+360010				Type	TXAdd	RXAdd	PD0-Length		IE 62 UF 0/ 31 5/ F2 U3 4/ DA /6 D5 23 28 CC 80		(ubiii)	
=1960007	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0xFFFF82580002	64 13 B8 41 57 93 BA 1C 6B 9E DD 7E 1D 3E 25	0x000076	-38	OK
<b>T</b> '						001111			1 d D-4		Deet	
Time (us)	Channel	Access Address	Adv PDII Type		Adv	PDU Hea	der	AdvA	AdvData	CPC	RSSI	FCS
+319994	channer	Access Address	Advibbilit	Type	TxAdd	RxAdd	PDU-Length	0000	1E 62 OF 07 31 54 F2 03 47 DA 76 D4 23 28 CC 80	CINC	(dBm)	103
=2280001	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0xFFFF82580002	64 13 B8 41 57 93 BA 1C 6B 9E DD 0C C1 5E 66	0x000084	-38	OK

#### Figure 17.1: Packet format

The effective payload length is 24 bytes. By default, each node requires 6 bytes. For details, please refer to

```
typedef struct{
    u16 dev_adr :15; // don't change include type
    u16 rsv :1;
    u8 sn; // don't change include type
    u8 par[MESH_NODE_ST_PAR_LEN]; //lumen-rsv,
}mesh_node_st_val_t;
```

Figure 17.2: reference details

The effective length of each node is MESH\_NODE\_ST\_PAR\_LEN (3). The specific data filled can be customized according to each product. The default fill BYTE 0 is brightness, BYTE 1 is the color temperature value, and BYTE 2 is reserved. Modify device\_status\_update () according to customer needs.

```
void device_status_update()
{
    // packet
    u8 st_val_par[MESH_NODE_ST_PAR_LEN] = {0};
    memset(st_val_par, 0xFF, sizeof(st_val_par));
    // led_lum should not be 0, because app will take it to be light off
    st_val_par[0] = light_lum_get(0, 1);
    #if (LIGHT_TYPE_CT_EN)
    st_val_par[1] = light_ct_lum_get(0, 1);
    #else
    st_val_par[1] = 0xff; // rsv
    #endif[
    // end
    ll_device_status_update(st_val_par, sizeof(st_val_par));
}
```

Figure 17.3: device\_status\_update

Note: MESH\_NODE\_ST\_PAR\_LEN (3), can be modified, but the larger the length, the slower the transmission speed. And there is no length field to describe this length, so when defining the network, you must first determine the value of MESH\_NODE\_ST\_PAR\_LEN. If the MESH\_NODE\_ST\_PAR\_LEN values configured by nodes in the network are inconsistent, there will be compatibility issues, resulting in data format parsing errors.

Log 🗌 fastbind 2 retry Clear Save Save	e 🔽 Hex 🗆 Adv Stop Scan rp_scan	OTA Rx test
Mesh 2 Mesh	Nodes online s - Group	schedule
001 0002 On Off O 100 002 0004 On Off O 100 3	get/set     All     On     Off     Svr     Clnt       ffff     0     On     Off         Group_S     1     On     Off         2     On     Off	→ O any C custom □ → Month □ Jan □ Fe □ Jul □ Au
	GrpDelAll_S       3       On       Off         GrpDelAll_C       4       On       Off         GetPub_S       5       On       Off         SecNwBc       6       On       Off	Hour C any hour Minute C any minut C every 15 C every 20

# 17.4 SIG\_MESH\_TOOL Firmware Demo

Figure 17.4: SIG\_MESH\_TOOL firmware demo

**Step 1** Choose online status as shown in figure above.

**Step 2** Click Node button as shown in figure above. After clicking, you can see from the log window that you did not send similar commands such as lightness get. Instead, you can directly obtain the online status data in the directly connected node through the custom UUID.

**Step 3** After clicking the Node button, the node display window on the left shows the node information of the current network.

# **18 Telink Proprietary OTA Test Brief**

# 18.1 GATT master dongle OTA for firmware update of BLE directly connected nodes

This mode is a point-to-point BLE Direct Connect OTA.

1) Download the BIN file (New FW) that requires OTA to the flash address of 8269 master dongle starting from 0x20000 according to the instructions in 7.1, burn 8269\_mesh\_master\_dongle.bin from address 0x0000.

Note: The difference of burning New FW:

BDT tool, please refer to the following steps:

Telink Burning and D	ebugging Too	ol (BDT)	_	-	_ 0 %
File View Tool Help	3		6		
<u>8</u> 258 ▼ <sup>1</sup> √ EVK ▼ (	© Se <u>t</u> ting (	🖲 Erase 🧯	Download + Activa	ate 🕨 Run II Pause 🏶 Step 🔍 PC 🖋 Single step 🗸 🧟 Reset 🌚 manual n	node 🔹 📕 <u>C</u> lear
b0 10	ь0	10	c sws	602 06 <b>Stall</b> 602 88	Start
Ŧ	Download			해현 Tdebug 톱 Log windows	
Variable Name	Addr	Len	Value		^
hci_tlk_module_event	42f00	4	0000000	IC82 EVK: Swire ok!	=
blt_event_func	42f04	80	/	TC32 FVK · Swire OK	
ll_host_main_loop_cb	42f54	4	0001c489	Flash Sector (4K) Erase at address 20000	
Il_encryption_dona_ch Il_connCo Il_connTer blc_tlkEve bltMac blt_state bltParam bltData LL_FEATU	A 24558 A ad Addr(H): iash Addr(H): Erase Size(K): tart Addr(H):	SRAM 20009 0 512 40000		Flash Page Program at address 20000 Flash Page Program at address 20400 Flash Page Program at address 20800 Flash Page Program at address 20000 Flash Sector (4K) Erase at address 21000 Flash Page Program at address 21000 Flash Page Program at address 21400 Flash Page Program at address 21800 Flash Page Program at address 21000 Flash Sector (4K) Erase at address 22000 Flash Page Program at address 22000	
blt_p_event_callback	42fa4	4	00000741	Flash Page Program at address 22800	
ble_state	42fa8	1	0000007	Flash Page Program at address 22c00	
ll_irq_rx_data_cb	42fac	4	00001a35	Flash Dector (4K) Erase at address 23000 Flash Page Program at address 23000	
revert_conn_crc	42fb0	4	00969996	Flash Page Program at address 23400	
Crc24Lookup	42fb4	64		Flash Page Program at address 23800	
Il_irq_systemTick_con	42ff4	4	00001921	Flash Fage Frogram at address 23cUU	-
				2	4
evk device: ok	File	e Path: E:\b	le_it_mesh\ble_lt_me	sh\ble_lt_mesh\8258_mesh\8258_mesh.bin	verion : 5.4.1

Figure 18.1: BDT tool

For wtcdb tool, click the "WF20000" button to start burning. Please refer to the following steps:

🔨 🛛 Telink

WT wtcdb		_	$\times$
D:\3.0.2\SIG_MESH_Release_V3.0.2_20191111\sdk\8258_mesh	-	BIN	Open
		DEF	Ini
BIN 5320 V OTP Program CORE V 00 1-byte V USB	Vtc	lb.ini	•
8258 mesh.bin Flash Sector (4K) Erase & Program at address 2	27000		^
8258 mesh3.0.2.bin Flash Sector (4K) Erase & Program at address 2	28000		
Flash Sector (4K) Erase & Program at address 2	29000		
Flash Sector (4K) Erase & Program at address 2	2a000		
Flash Sector (4K) Erase & Program at address 2	2Ъ000		
Flash Sector (4K) Erase & Program at address 2	20000		
Flash Sector (4K) Erse & Frogram at address 2	2a000		
Flash Sector (4K) Erase & Program at address 2	2£000		
Flash Sector (4K) Erase & Program at address 3	30000		
Flash Sector (4K) Erase & Program at address 3	31000		
Flash Sector (4K) Erase & Program at address 3	32000		
Flash Sector (4K) Erase & Program at address 3	33000		
Flash Sector (4K) Erase & Program at address 3	34000		
Flash Sector (4K) Erase & Program at address 3	35000		
Flash Sector (4K) Erase & Program at address 3	36000		
Flash Sector (4K) Erase & Program at address 3	37000		
Flash Sector (4K) Erase & Program at address 3	39000		
Flash Sector (4K) Erase & Program at address 3	3a000		
Flash Sector (4K) Erase & Program at address 3	3Ъ000		
Flash Sector (4K) Erase & Program at address 3	3c000		
Flash Sector (4K) Erase & Program at address 3	3d000		
Flash Sector (4K) Erase & Program at address 3	3e000		
Flash Sector (4K) Erase & Program at address 3	3£000		
Flash Sector (4K) Erase & Program at address 4	40000		
Flash Sector (4K) Frase & Program at address 4 Flash Sector (4K) Frase & Drogram at address 4	42000		
Flash Sector (4K) Ease & Frogram at address 4	43000		
Flash Sector (4K) Erase & Program at address 4	44000		
file dowload to 00020000: 150148 bytes			
Total Time: 11549 ms			
			~
TRACE Firware UART M S R Ltcdb.exe wf 20000 -eb -i			
tr Start USB Text 5 2			
4k VCD SRAM 4 Tdebug DelPair W20000 E256k ReadF ReadPC dor	ngleII P	owerOff	CTRL_C
33 View SWB Hex SWB SP CmdWnd E512k PktCap ReadID RstMCU ming32 mo	ouseID	gpiocfg	Close
Ready		idl	e:4

Figure 18.2: wtcdb tool

- 2) Refer to the steps (1) to (5) in Section 4.4 to establish a BLE connection between the target node and the tool.
- 3) Click the OTA button to start the OTA process. If the OTA is completed normally, the node will flash 8 times continuously.

🚯 Telink master Found	Х
<pre>CHD sig_mesh_master.ini INI BULKOUT ASCII CHD sig_mesh_master.ini INI BULKOUT ASCII LDN get_lightness LDN get_lightness LDN get_onoff lightness_get_Panel</pre>	<pre>X</pre> I Log  fastbind  2 retry Clear Save Save  K Hex  Adv Stop Scan rp_scan  OTA Rx test <6171>10:03:23:833 [INFO]: (cmd_rsp)Status Rsp: 04 00 01 00 82 4e ff ff <6172>10:03:23:874 [INFO]: (common) ExecCnd: a3 ff 00 00 00 00 20 10 ff ff 82 02 00 00 <6173>10:03:25:881 [INFO]: (common) ExecCnd: a3 ff 00 00 00 00 20 11 ff ff 82 02 00 00 <6175>10:03:26:031 [INFO]: (cmd_rsp)Status Rsp: 04 00 01 00 82 4e ff ff <6175>10:03:26:031 [INFO]: (Basic) ath mesh access tx cmd is 0x0222 : 00 02 <6175>10:03:26:031 [INFO]: (cmd_rsp)Status Rsp: 04 00 01 00 82 04 01 00 0a <6175>10:03:26:031 [INFO]: (cmd_rsp)Status Rsp: 04 00 01 00 82 04 01 00 0a <617>10:03:26:033 [INFO]: (log_win32)mesh tx_reliable_stop: op 0x0282 rsp max 1, rsp cnt 1 <617>10:03:27:138 [INFO]: (common) ExecCnd: a3 ff 00 00 00 00 00 01 ff ff 82 02 01 00 <6180>10:03:27:138 [INFO]: (Basic) ath mesh access tx cmd is 0x0222 : 01 03 <6180>10:03:27:138 [INFO]: (Basic) ath sc: 0x00004, att cst: 0x0001A, access rx cmd is 0x482 : 82 04 00 01 0a
<pre>[fw_distribution_start_02_03 fw_distribution_start_0001 fw_distribution_stop fw_distribution_detail_get</pre>	<pre>&lt;6181&gt;10:03:27:228 [INFO]:(cmd_rsp)Status Rsp: 04 00 01 00 82 04 00 01 0a <f182>10:03:27:260 [INFO]:(log win32]mesh tx reliable stop: op 0x0282 rsp_max 1, rsp_cnt 1 <f183>10:03:32:157 [INFO]:(common)GAT TOTA statt <f184>10:05:06:138 [INFO]:(common)GAT TOTA completed: 0K</f184></f183></f182></pre>



4) For the commands of the OTA part and the details of the protocol part, please refer to Chapter 6.4



of "AN\_17092701\_Telink 826x BLE SDK Developer Handbook". There is a detailed description of the command and protocol format.

## 18.2 OTA OTA where the Gateway node updates its firmware

The purpose of Gateway Gate OTA is to upgrade the firmware of gateway itself.

- 1) Open the BDT tool and download 8258\_mesh\_gw.bin to 8258 dongle.
- 2) "Found" in the upper left corner of the tool means that the 8258 Dongle and the PC tool are normally connected and can communicate normally.



3) Click the button in the lower right corner and select a different version of the 8258\_mesh\_gw.bin file;

, mash 1	▼ INI BULKOUT	ASCII V Log fastbind 2 retr	y Clear Save Save	Hex Adv	Stop Scan	rp_scan	OTA Rx tes
PN_get_lightness PN_get_onoff	art_cat_debug						
ightness_get_Panel	-						
w info get	🐉 打开				×		
_info_get_all	6 A			E12 0 21	0		
_distribution_get		PH284 / 4000200 (D.) / 5.0.2 /	♥ 0 授3	£ 3.0.2	10		
_distribution_start_0002	组织 ▼ 新建文件夹						
_distribution_start_02_03		^					
distribution_start_0001	OneDrive	名称	修改日期	类型	大小		
distribution_detail_get	-	SIG MESH Release V3.0.2 20191111	2019/11/12 10:14	文件李			
_update_get	🛄 此电脑	( 8258 mech aw hin	2010/11/26 17:50	RIN 文件	133		
update_prepare	🧊 3D 对象	C250_mean_gw.bm	2013/11/2011.33		155		
update_abort	📕 视频						
update_apply							
j transfer start							
j_transfer_abort	🗐 文档						
j_block_transfer_start	👆 下戦						
j_block_get	音乐						
j_info_get							
	某用						
heduler get	🏪 本地磁盘 (C:)						
heduler_get hed_action_get	and a laborated state of the laborated						
eduler_get ed_action_get ed_action_set_off	🔜 本地磁盘 (D:)						
heduler_get hed_action_get hed_action_set_off hed_action_set_on hed_action_set_scenel	本地磁盘 (D:)	<					
heduler_get hed_action_get hed_action_set_off hed_action_set_on hed_action_set_scenel	▲ 本地磁盘 (D:)	<					
heduler_get hed_action_get hed_action_set_off hed_action_set_con hed_action_set_scenel  me_set	本地磁盘 (D:) ▲ □···· × 文件	< 名(N): *.bin	∽ Bir	Files (*.bin)	~		
heduler_get hed_action_get hed_action_set_off hed_action_set_scenel hed_action_set_scenel heget heget heget	本地磁曲 (D:) ▲ m · · · · · · · · · · · · · · · · · ·	< 名(N): 「bin	∽ Bir	Files (*.bin)			
hedular_get hed_action_get_ hed_action_set_off hed_action_set_con hed_action_set_scenel me_set me_set me_set me_set_set me_set_set me_set_set	本地题曲 (D:) ▲ m·x × 文件	< ۲.bin	- Bir	n Files (*.bin) 打开(O)	~ 取消		
hedular_get hed_action_get hed_action_set_off hed_action_set_on hed_action_set_on mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag_et mag mag mag mag mag mag mag mag mag mag	本地磁盘 (D:) ▲ 日本	<	~ Bir	n Files (*.bin) 打开(O)			
heduler get hed_action_set hed_action_set_off hed_action_set_off hed_action_set_off heg_set heg_set heg_set heg_set heg_set heg_cong_set heg_set heg_set heg_set heg_set heg_set heg_set heg_set	本地短盘 (D:) ▲ m · · · · · · · · · · · · · · · · · ·	< \$(N): *.bin	→ Bir	n Files (*.bin) 打开(O)			
heduler_get hed_action_get hed_action_get_off hed_action_get_off hed_action_get_on me_set me_set me_set me_set me_set me_set me_set me_set me_set me_set me_set me_set me_set me_set me_set	本地經量 (D:)	<	~ Bi	n Files (*.bin) 打开(O)			

Figure 18.4: Select bin file

4) Click the Gate\_ota button to start the upgrade. LOG prompts gateway firmware load suc to indicate that the upgrade was successful. After the upgrade is successful, the gateway will automatically restart and enable the new firmware.

8 Telink sig_mesh Found	×
CMD tl_node_gateway.ini V INI BULKOUT ASCII	Log [ fastbind 2 retry Clear Save Save V Hex Adv Stop Scan rp scan OTA Rx test
mesh bulk cmd debug	
LPN get lightness	<0061>16:54:54:552 [INFO]: (common) firmware download process is 62 percent
LDN get onoff	<0062>16:54:55:430 [INFO]:(common)firmware download process is 63 percent
lightness get Panel	<0063>16:54:56:602 [INFO]:(common)firmware download process is 64 percent
Tranchess_geo_raner	<0064>16:54:57:774 [INFO]:(common)firmware download process is 65 percent
fu info get	<0065>16:54:58:934 [INFO]:(common)firmware download process is 66 percent
fu info get all	<0066>16:55:00:076 [INFO]:(common)firmware download process is 67 percent
fu distribution get	<0067>16:55:01:147 [INFO]:(common)firmware download process is 68 percent
fu distribution start all	<0068>16:55:02:274 [INFO]:(common)firmware download process is 69 percent
fw_distribution_start_all	<0069>16:55:03:430 [INFO]:(common)firmware download process is 70 percent
fw distribution start 0002	<0070>16:55:04:586 [INFO]: (common) firmware download process is 71 percent
fu distribution start 0001	<0071>16:55:05:743 [INFO]:(common)firmware download process is 72 percent
fu distribution start ovor	<0072>16:55:06:915 [INFO]:(common)firmware download process is 73 percent
fu distribution_stop	<0073>16:55:08:038 [INFO]:(common)firmware download process is 74 percent
Tw_distribution_detail_get	<0074>16:55:09:145 [INFO]:(common)firmware download process is 75 percent
rw_update_get	<0075>16:55:10:243 [INFO]:(common)firmware download process is 76 percent
rw_update_prepare	<0076>16:55:11:414 [INFO]:(common)firmware download process is 77 percent
rw_update_start	<0077>16:55:12:586 [INFO]:(common)firmware download process is 78 percent
rw_update_abort	<0078>16:55:13:743 [INFO]: (common) firmware download process is 79 percent
IW_update_appiy	<0079>16:55:14:915 [INFO]: (common) firmware download process is 80 percent
obj_transfer_get	<0080>16:55:16:044 [INFO]: (common) firmware download process is 81 percent
obj_transfer_start	<0081>16:55:17:142 [INFO]:(common)firmware download process is 82 percent
obj_transfer_abort	<0082>16:55:18:244 [INFO]: (common) firmware download process is 83 percent
obj_block_cransfer_start	<0083>16:55:19:414 [INFO]:(common)firmware download process is 84 percent
obj_cnunk_transfer	<0084>16:55:20:555 [INFO]:(common)firmware download process is 85 percent
obj_block_get	<0085>16:55:21:698 [INFO]:(common)firmware download process is 86 percent
obj_inio_gec	<0086>16:55:22:806 [INFO]:(common)firmware download process is 87 percent
cohodulor cot	<0087>16:55:23:964 [INFO]:(common)firmware download process is 88 percent
scheduler_get	<0088>16:55:25:106 [INFO]:(common)firmware download process is 89 percent
sched_action_get	<0089>16:55:26:171 [INFO]: (common) firmware download process is 90 percent
sched action set on	<0090>16:55:27:304 [INFO]:(common)firmware download process is 91 percent
sched action set on	<0091>16:55:28:477 [INFO]:(common)firmware download process is 92 percent
sched_action_set_scener	<0092>16:55:29:649 [INFO]:(common)firmware download process is 93 percent
time set	<0093>16:55:30:821 [INFO]:(common)firmware download process is 94 percent
time get	<0094>16:55:31:997 [INFO]:(common)firmware download process is 95 percent
time zone set	<0095>16:55:33:119 [INFO]:(common)firmware download process is 96 percent
time rone get	<0096>16:55:34:107 [INFO]:(common)firmware download process is 97 percent
time delta set	<0097>16:55:35:351 [INFO]: (common) firmware download process is 98 percent
time delta get	2008-1615-361508 [INFO]+(common) firmware download process is 09 percent
time role set	10099>16:55:37:695 [INFO]:(common)gateway firmware load suc
time role get	· · · · · · · · · · · · · · · · · · ·
orme_rore_geo	< >
scene store	
· · · · · · · · · · · · · · · · · · ·	ALL CNN_Set Connect GATE_RESET Pata: D:\3.0.2\02588_me search_file Mesh
e8 ff 00 00 00 00 02 00 02 00 60 41 06 00 80 00 04 10 00 00	
	VARI USB Mesh_ota Gate_ota Prov Close

Figure 18.5: Click Gate\_ota

Telink

T

# **19 Network Sharing**

For more information, you can also refer to the section 33.5.1.2 Share Export of the chapter Android and iOS APP User Guide.

# 19.1 Share Mode of App share from Gateway or GATT Master Dongle

**Step 1** Use VC master or gateway networking to make sure that the 8258 dongle nodes can be properly networked and controlled;

**Step 2** Click on "output\_db" on the homepage; (you can't copy and paste the JSON file directly because you have to remove some added fields like iv index, etc.)

**Step 3** Import the output json file into TelinkSigmesh APP.

IOS APP Steps:

**Step 1** Connect your phone to a computer with iTunes installed.

**Step 2** Click the phone icon in the upper left corner of iTunes to enter the iTunes device details interface.

**Step 3** Select "File Sharing" on the left side of iTunes, then find and click the demo APP "TelinkSigMesh" in the app, and wait for iTunes to load the file.

**Step 4** After the file is loaded, drag the json file on your computer into the "TelinkSigMesh" document on the right

**Step 5** Click the IMPORT button in the APP to select the JSON file to load. Detailed steps are in the pictures below:





Figure 19.1: Click Setting



🗉 🛛 Telink

	Setting	V3.0.0
Scenes		>
Share		>
Mesh OTA		>
Debug		>
💮 Log		>
Mesh Info		>
Choose Add Dev	ces	>
Q	Group	Setting
	0.00p	ig

Figure 19.2: Click Share

c) Click IMPORT button

✓ Share by iTunes	, ,
Import JSON:	
<ol> <li>Iphone connect to computer that install iTunes.</li> <li>Click on the iTunes phone icon in the upper left corner of iTunes into the iphone interface.</li> <li>Select "file sharing" in the left of the iTunes, then find and click on the demo APP in the application of "TelinkSigMesh", wait for iTunes load file.</li> <li>After file is loaded, drag the files on the computer "mesh.json" into the right side of the "TelinkSigMesh", replace the old file and reopen the APP, the APP will load json data file automatically.</li> <li>Click IMPORT button to choose new json file and load it.</li> </ol>	
导入 JSON 数据操作,步骤如下:	
1. 将手机连接到安装了iTunes的电脑上。 2. 点击iTunes左上角的手机图标进入iTunes设备详情界 面。 3. 选择iTunes左侧的"文件共享",然后在迎用中找到并点 击demo APP "TelinkSigMesh",等待iTunes加载文件。 4. 文件加载完成后,将电脑上的json文件拖入右侧 的"TelinkSigMesh"的文稿中。 5. APP点击 IMPORT按钮选择刚刚的 JSON 文件进行加载。	
IMPORT	

Figure 19.3: Click IMPORT

d) Click mesh.json, click IMPORT



Figure 19.4: Click mesh.json

Import file with Android APP:

**Step 1** Connect the phone to computer, import the mesh.json file into any folder on the phone, and remember the path to the folder.

a) Open APP TelinkSigmesh, click Setting button.

Share

С	D	evice	+
ALL ON	ALL OFF	CMD	LOG

Figure 19.5: Click Setting

b) Click Share button

			Setting	V3.0.J
		Scenes		>
		Share		>
		Mesh OTA		>
		Settings		>
		0	Ē	\$
		Device	Group	Setting
		Figure 1	9.6: Clio	ck Share
c) Click IMPORT button				~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
	1	<	Share	0
		EXPORT		IMPORT
		Import mesh stora	age from json	
	1em			
		Select File		
		Une of the	IMPORT	

Figure 19.7: Click IMPORT

d) Click Select File to select mesh.json imported from the computer

	< Share 🕐			
	EXPORT IMPOR	г		
	Import mesh storage from json File selected: /storage/emulated/0/tencent/ QQfile_recv/mesh.json			
[	/storage/emulated/0/tencent/QQfile_rec mesh.json	v/:>		
	IMPORT			
	PREVIEW			
	Figure 19.8: Click mesh	.json		
Т	8	SC		

e) Click Import button

**Step 2** After the import is successful, return to the main page: APP Device interface. At this time, the shared nodes will be displayed. These nodes are the nodes of the VC tool network. APP can be controlled, and both VC and APP can control the nodes.

# 19.2 Share Mode of Gateway or GATT Master Dongle share from App

- 1) Provision with iOS or Android TelinkSIGmesh APP, and works normally.
- 2) Click Setting button, then click Share button to enter share interface, click EXPORT button, generate JSON file. The path of JSON file folder will be displayed on the interface.



Figure 19.9: JSON file path

- 3) Connect the master 8269 Dongle or gateway dongle to the PC.
- 4) Open the "SIG\_MESH\_TOOL" tool, then it will show that master 8269 Dongle and PC tool are connected normally or gateway 8258 Dongle and PC tool are connected normally.
- 5) Click the "input\_db" button to import the mesh.json generated by the app.(You can't copy and paste the JSON file directly, because you have to clear the node max, etc. inside the ini file, or else the address space of the unicast address will be wasted, etc.)

Note: The Gateway dongle is generally one that has not yet been configured for the network, if it has been configured, the information in it will be deleted and the imported data will be used.

- 6) Click mesh button, the imported nodes will show in the mesh window, and can be controlled.
- 7) Not the sharing is completed, gateway/master dongle and APP can all control the nodes.



#### Figure 19.10: Complete sharing

# 20 Control Nodes via INI Demo

# 20.1 Provision Device

Device provision is slightly different for PB-GATT and PB-ADV, but it is the same for interface and operation procedure.

Step 1 Scan UNprovision\_beacon adv devices;

**Step 2** Connect according to MAC of the scanned UNprovision adv devices;

**Step 3** Provision device;

#### **Step 4** Bind model.

The following demo is to test with master dongle. Click stop, then click scan to enter scan mode, connect the scanned device, with the mac of: 112233445566.



Figure 20.1: Connect device

Log information after the device is connected:

```
29 02 01 00 00 e4 4a d8 a7 6c 7f 1f 1e ff 06 00 01 09 20 02 23 73 90 d6 fc c3 94 59 c5 fa 87 08 86 ef 1d af e2 85 c0 41 6b 83 d8 ca 14 00
<11788>10:53:00:253 [INFO]: (common) adv pkt: 29 02 01 00 00 26 29 c0 89 d2 7b 0e 02 01 1a 0a ff 4c 00 1
<11789>10:53:00:266 <11790>10:53:00:396 [INFO]:(gatt_provision)CScanDlg::OnConnect:the device uuid is
 91 2f 68 1d 5c 48 fb 3d b6 3e 11 22 33 44 55 66
[INFO]: (common) adv pkt:
 29 02 01 00 00 11 22 33 44 55 66 1d 02 01 06 03 03 27 18 15 16 27 18 91 2f 68 1d 5c 48 fb 3d b6
 3e 11 22 33 44 55 66 00 00 ca 4a 00
<11791>10:53:01:118 [INFO]:(common)Mesh Provisioning Service:
<11792>10:53:01:132 [INFO]:(common)uuid:dc 2a
<11793>10:53:01:144 [INFO]:(common)the handle:13
<11794>10:53:01:160 [INFO]:(common)Mesh Proxy Service:
<11795>10:53:01:177 [INFO]:(common)uuid:de 2a
<11796>10:53:01:191 [INFO]:(common)the handle:1c
<11797>10:53:01:201 [INFO]:(Basic)filter send cmd is 0: 00
<11798>10:53:01:222 [INFO]:(Basic)filter send cmd is 1: 4b 7e
<11799>10:53:01:242 [INFO]:(Basic)filter send cmd is 1: ff ff
<11800>10:53:01:264 [INFO]:(iv_update)app tx beacon with GATT,IV index step0: : 12 34 56 78 12 34 56 7
<11801>10:53:01:279 [INFO]:(iv_update)secure NW beacon:: 17 2b 01 00 44 cf 7a d5 44 8c f1 6e 12 34 56
<11802>10:53:01:359 [INFO]:(Basic)the filter rsp is 0: 00 00 46 6f
<11803>10:53:01:373 [INFO]:(Basic)mesh_rc_data_cfg_gatt dec suc
<11804>10:53:01:388 [INFO]:(log_win32) white list
<11805>10:53:01:405 [INFO]:(log_win32)GATT addr 0x2211, filter list status, ListSize is: 0
<11806>10:53:01:422 [INFO]:(Basic)the filter rsp is 0: 00 01 74 0e
<11807>10:53:01:432 [INFO]:(Basic)mesh_rc_data_cfg_gatt dec suc
<11808>10:53:01:444 [INFO]:(log_win32) white list
<11809>10:53:01:456 [INFO]:(log_win32)GATT addr 0x2211, filter list status, ListSize is: 1
<11810>10:53:01:467 [INFO]: (Basic) the filter rsp is 0: 00 02 85 01
<11811>10:53:01:479 [INFO]:(Basic)mesh_rc_data_cfg_gatt dec suc
<11812>10:53:01:494 [INFO]:(log_win32) white list
<11813>10:53:01:507 [INFO]:(log_win32)GATT addr 0x2211, filter list status, ListSize is: 2
```

#### Figure 20.2: log info

#### **Provision Parameter Setting and Device Provision**

- 1) Click prov button to enter provision interface, first click SetPro\_internal to assign net key to dongle.
- 2) Click provision button to provision the connected device with MAC of 112233445566, during provision, assign netkeyindex, IVindex, unicast\_addr generated by firmware to device.
- 3) Click bind\_all button to bind APPkey(based on the model reported by devicecomposition data).

provision	X
Fast prov mode         SetPro_internal         network_key         11 22 c2 c3 c4 c5 c6 c7 d8 d9 da db dc dd de df	Static         apk_id×       00 00         app_key       60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48         bind_all
key_index     00 00     iv_index     11 22 33 44       iv_update_flag     0     unicast_adr     02 00       Provision	
filter_operation filter_type white_list v filter_data 01 00 ff ff SetFilter Add_mac RM_mac	

#### Figure 20.3: Provision parameter setting and device provision

As shown in figure below, the data interaction of Provision is described as following:

- 1) Start provision, provisioner send out data
- 2) public key interaction
- 3) check confirm;
- 4) send provision data (net\_key/nkey\_index/IV\_update\_flag/IV\_index/unicast\_addr);
- 5) Add proxy white list

				.,100130110120130
1	<0000>10:54:23:620	[INFO]:(gatt_provision)Set internal provision success		
2	<0001>10:54:27:433	[INFO]:(gatt_provision)start provision for the device		
N.	<0002>10:54:27:434	[ERR]:(common)obj_adr 0x0002, not found VC node info		
4	<0003>10:54:27:438	[INFO]:(gatt_provision)SEND:provisioner send invite cmd		
5	: 00 00			
ſ	<0004>10:54:27:485	[INFO]:(gatt_provision)RCV:the provision capa data is		
ŀ	<0005>10:54:27:487	[INFO]:(gatt_provision)SEND:the provision start is	2	
l	<0006>10:54:27:499	[INFO]:(gatt_provision)SEND:provisioner send pubkey is	-	
	<0007>10:54:27:607	[INFO]:(gatt_provision)RCV:the pubkey of the device is		
¢				
	<0008>10:54:27:617	[INFO]:(gatt_provision)SEND:the provisioner's comfirm is		
ł	<0009>10:54:29:365	[INFO]:(gatt_provision)RCV:the device's comfirm is		
	<0010>10:54:29:370	[INFO]:(gatt_provision)SEND:the provisioner's random is	3	
	<0011>10:54:29:444	[INFO]:(gatt_provision)RCV:the device's random is		
	<0012>10:54:29:451	[INFO]:(gatt_provision)the device comfirm check is success		
F				
	<0013>10:54:29:459	[INFO]: (gatt_provision) SEND: the provisioner's device info is	3	
ł	: 11 22 c2 c3 c4 c5	5 c6 c7 d8 d9 da db dc dd de df 00 00 00 11 22 33 44 02 00		4
	<0014>10:54:29:465	[INFO]:(gatt_provision)the node's dev key:		
	: 97 14 da 8c 08 a4	4 5a c4 d6 f9 ea c2 8f 7a d9 e9		
	<0015>10:54:29:685	[INFO]: (gatt_provision) RCV:rcv the provision completet cmd,	provision success	
2				
5	<0016>10:54:29:699	[INFO]:(Basic)filter send cmd is 0: 00		
4	<0017>10:54:29:717	[INFO]:(Basic)filter send cmd is 1: 00 01		5
	<0018>10:54:29:737	[INFO]:(Basic)filter send cmd is 1: ff ff		3
¢	<0019>10:54:29:755	[INFO]:(iv_update)RX beacon,nk arr idx:0, new:0, pkt:		
ł	: 17 2b 01 00 56 52	2 3e be 74 5f f6 3e 11 22 33 44 f0 bf e9 8c 4e e5 9a 1f		
8	<0020>10:54:29:805	[INFO]:(Basic)the filter rsp is 0: 00 00 08 ec		
5	<0021>10:54:29:814	[INFO]:(Basic)mesh_rc_data_cfg_gatt_dec_suc		
¢	<0022>10:54:29:822	[INFO]: (log win32) white list		
1	<0023>10:54:29:833	[INFO]: (log win32)GATT addr 0x0002, filter list status, List	tSize is: 0	
	<0024>10:54:29:848	[INFO]: (Basic) the filter rsp is 0: 00 01 7e bc		
	<0025>10:54:29:859	[INFO]: (Basic)mesh_rc_data_cfg_gatt dec suc		
k	<0026>10:54:29:868	[INFO]: (log win32) white list		
	<0027>10:54:29:879	[INFO]: (log win32) GATT addr 0x0002, filter list status, List	tSize is: 1	
1				

Figure 20.4: The data interaction of Provision

Bind introduction:

Bind is to bind APP key with all device models according to the composition data of the device after the provision. This process will determine the bind time according to the number of models (only 20s). Therefore, bind has been optimized on Tmall Genies and other platforms, and fastbind will be introduced next.

Before Bind, you need to get the composition data of the device first. For detailed analysis format of Composition data, please refer to <4.2.1 Composition Data> of <Mesh\_v1.0>.

```
<0322>19:20:53:228 [INFO]: (KEYEIND) start key bind and the appkey is
: 60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48
<0323>19:20:53:239 [INFO]: (KEYEIND) SEND: get composition data
<0324>19:20:53:252 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 01 02 00 80 08 ff
<0325>19:20:53:262 [INFO]: (Basic) the mesh access tx cmd is 0x0880 : ff
<0326>19:20:53:431 [INFO]: (Basic) the mesh access tx cmd is 0x0800 : ff
<0326>19:20:53:442 [INFO]: (Basic) adr_src:0x0002, adr_dst:0x0001, access rx cmd is 0x2 :
02 00 11 02 01 00 32 39 69 00 07 00 00 00 11 01 00 00 02 00 03 00 04 00 00 fe 01 fe 00 ff 01 ff
00 10 02 10 04 10 06 10 07 10 00 13 01 13 03 13 04 13 11 02 00 00 00 00 02 00 02 10 06 13
<0328>19:20:53:454 [INFO]: (cmd_rsp) Status Rsp______:
02 00 01 00 02 00 11 02 01 00 32 39 69 00 07 00 00 07 00 00 01 10 10 00 02 00 03 00 04 00 00 fe 01 fe
00 ff 01 ff 00 10 02 10 04 10 06 10 07 10 00 13 01 13 03 13 04 13 11 02 00 00 00 02 00 03 00 04 00 00 fe 01 fe
00 ff 01 ff 00 10 02 10 04 10 06 10 07 10 00 13 01 13 03 13 04 13 11 02 00 00 02 00 03 00 04 00 00 fe 01 fe
00 ff 01 ff 00 10 02 10 04 10 06 10 07 10 00 13 01 13 03 13 04 13 11 02 00 00 02 00 03 00 04 00 00 fe 01 fe
00 ff 01 ff 00 10 02 10 04 10 06 10 07 10 00 13 01 13 03 13 04 13 11 02 00 00 00 00 02 00 02 00 02 10
06 13
```



According to the corresponding information of the obtained element and model, the bind command is subsequently issued to bind model by model.

NUG20710.20.00.400	[INTO].(IDg_WINSE/MESH_CA_ICIIGHTE_SCOP. OP GAUGUD ISP_MEA I, ISP_CHC I
<0330>19:20:53:490	[INFO]: (KEYBIND) SEND: appkey add ,0x0 is the appkey index
: 60 96 47 71 73 41	bd 76 e3 b4 05 19 d1 d9 4a 48
<0331>19:20:53:502	[INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 02 00 00 00 00 00 06 96 47 71 73 4f
<0332>19:20:53:514	[INFO]:(Basic)the mesh access tx cmd is 0x0000 : 00 00 00 60 96 47 71 73 4f bd 76 e
<0333>19:20:53:872	[INFO]:(Basic)rc data layer upper:segment tx success, map in ACK is: 03 00 00 00
<0334>19:20:53:883	[INFO]:(Basic)adr_src:0x0002,adr_dst:0x0001,access rx cmd is 0x380 : 80 03 00 00 00
<0335>19:20:53:893	[INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 80 03 00 00 00 00
<0336>19:20:53:907	[INFO]:(log_win32)mesh_tx_reliable_stop: op 0x0000 rsp_max 1, rsp_cnt 1
<0337>19:20:53:919	[INFO]:(KEYBIND)SEND: appkey bind addr: 0x0002,sig model id: 0x0002
<0338>19:20:53:932	[INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 02 00 80 3d 02 00 00 00 02 00
<0339>19:20:53:945	[INFO]:(Basic)the mesh access tx cmd is 0x3d80 : 02 00 00 00 02 00
<0340>19:20:54:032	[INFO]:(Basic)adr_src:0x0002,adr_dst:0x0001,access rx cmd is 0x3e80 : 80 3e 00 02 0
<0341>19:20:54:044	[INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 80 3e 00 02 00 00 02 00
<0342>19:20:54:063	[INFO]:(log_win32)mesh_tx_reliable_stop: op 0x3d80 rsp_max 1, rsp_cnt 1
<0343>19:20:54:073	[INFO]: (KEYBIND) SEND: appkey bind addr: 0x0002, sig model id: 0x0003
<0344>19:20:54:085	[INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 02 00 80 3d 02 00 00 03 00
<0345>19:20:54:096	[INFO]:(Basic)the mesh access tx cmd is 0x3d80 : 02 00 00 03 00
<0346>19:20:54:192	[INFO]:(Basic)adr_src:0x0002,adr_dst:0x0001,access rx cmd is 0x3e80 : 80 3e 00 02 0
<0347>19:20:54:202	[INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 80 3e 00 02 00 00 03 00
<0348>19:20:54:219	[INFO]:(log_win32)mesh_tx_reliable_stop: op 0x3d80 rsp_max 1, rsp_cnt 1
<0349>19:20:54:231	[INFO]:(KEYBIND)SEND: appkey bind addr: 0x0002,sig model id: 0x0004
<0350>19:20:54:245	[INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 02 00 80 3d 02 00 00 04 00
<0351>19:20:54:258	[INFO]:(Basic)the mesh access tx cmd is 0x3d80 : 02 00 00 04 00
<0352>19:20:54:352	[INFO]:(Basic)adr_src:0x0002,adr_dst:0x0001,access rx cmd is 0x3e80 : 80 3e 00 02 0
<0353>19:20:54:368	[INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 80 3e 00 02 00 00 04 00
<0354>19:20:54:384	[INFO]:(log_win32)mesh_tx_reliable_stop: op 0x3d80 rsp_max 1, rsp_cnt 1
<0355>19:20:54:396	[INFO]: (KEYBIND) SEND: appkey bind addr: 0x0002, sig model id: 0xfe00
<0356>19:20:54:407	[INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 02 00 80 3d 02 00 00 00 06 fe
<0357>19:20:54:418	[INFO]:(Basic)the mesh access tx cmd is 0x3d80 : 02 00 00 00 00 fe
<0358>19:20:54:512	[INFO]:(Basic)adr_src:0x0002,adr_dst:0x0001,access rx cmd is 0x3e80 : 80 3e 00 02 0
<0359>19:20:54:527	[INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 80 3e 00 02 00 00 00 06 fe
<0360>19:20:54:547	[INFO]: (log win32)mesh tx reliable stop; op 0x3d80 rsp max 1, rsp cnt 1

#### Figure 20.6: Bind

<0448>19:20:57:152 [INFO]:(Basic)adr\_src:0x0002,adr\_dst:0x0001,access rx cmd is 0x3e80 : 80 3e 00 03 0
<0449>19:20:57:166 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 02 00 01 00 80 3e 00 03 00 00 06 13
<0450>19:20:57:187 [INFO]:(log\_win32)mesh\_tx\_reliable\_stop: op 0x3d80 rsp\_max 1, rsp\_cnt 1
<0451>19:20:57:218 [INFO]:(KEYBIND)SEND: mesh keybind event success

#### Figure 20.7: Bind

#### 20.2 Configuration Operations

#### 20.2.1 Key add/bind Operation

#### APPKey add command format analysis:

CMD-cfg\_appkey\_add\_001= a3 ff 00 00 00 00 02 00 07 00 00 00 00 00 60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48

:	1	2	3	4	5	7	8	9	9 10 11		12			
			reliable eliab		eliable				para0	para1	para2	para3	para4	para5
Fl	ag	nk_idx	ak_idx retry count rsp	ak_idx retry count rsp_max dst op	<_idx retry count rsp_max dst	nt rsp_max dst	ор	Ap	pKeyInd	lex		app key		
a3	ff	0000	0000	02	00	0700	00		000000		60 96 4 e3 b4 0	7 71 73 4 5 19 d1 d	f bd 76 9 4a 48	

#### Figure 20.8: APPKey add command



Command format analysis:

- Flag: Defined by Telink, to identify the header packet of USB or UART communication, UART is E8FF, USB is A3FF.
- NK\_idx: network key index
- Ak\_idx: APP key index
- Reliable retry cnt: Application layer retry (the number of resends if the firmware cannot receive a reply after issuing a command)
- Reliable resp\_max: set the number of nodes to reply
- Dst: destination address filling
- Op: standard command code defined by sig mesh specification, please refer to <Mesh\_v1.0>, if the command code is not fixed, please refer to <4.3.4 Messages summary> in the document.

[9:12]: Transmission parameter part. Please refer to <4.3.2.37 Config AppKey Add> in <Mesh\_v1.0> for filling data.

Field	Size (octets)	Notes
NetKeyIndexAndAppKeyIndex	3	Index of the NetKey and index of the AppKey
АррКеу	16	AppKey value

#### Figure 20.9: Filling data

#### Key bind command format analysis:

CMD-cfg\_appkey\_bind\_001 = a3 ff 00 00 00 00 02 00 02 00 80 3d 02 00 00 00 00 10

Command format analysis:

- Flag: Defined by Telink, to identify the header packet of USB or UART communication, UART is E8FF, USB is A3FF.
- NK\_idx: network key index
- Ak\_idx: APP key index
- Reliable retry cnt: Application layer retry (the number of resends if the firmware cannot receive a reply after issuing a command)
- Reliable resp\_max: set the number of nodes to reply
- Dst: destination address filling
- Op: standard command code defined by sig mesh specification, please refer to <Mesh\_v1.0>, if the command code is not fixed, please refer to <4.3.4 Messages summary> in the document.

[9:12]: Transmission parameter part, light HSL control command filling data please refer to <4.3.2.46 Config Model App Bind> in <Mesh\_v1.0>.

Transmission parameter format reference:

Field	Size (octets)	Notes
ElementAddress	2	Address of the element
AppKeyIndex	2	Index of the AppKey
Modelldentifier	2 or 4	SIG Model ID or Vendor Model ID

Figure 20.10: Transmission parameter format reference

#### 20.2.2 Subscription Configuration

CMD-cfg\_sub\_add = a3 ff 00 00 00 00 00 01 02 00 80 1b 02 00 01 c0 00 10

Or please refer to group index control in 4.5.2.

#### 20.2.3 Publish configuration

CMD-cfg\_pub\_set\_sig\_2s = a3 ff 00 00 00 00 00 00 02 00 03 02 00 01 00 00 00 ff 14 15 00 10

Or please refer to GetPub\_S control button in 4.5.3

#### 20.2.4 Relay/Friend Function Configuration

Relay: a3 ff 00 00 00 00 02 01 07 00 80 27 01

```
<0000>11:44:52:572 [INFO]: (common)ExecCmd: a3 ff 00 00 00 02 01 0a 00 80 27 01
<0001>11:44:52:573 [INFO]: (Basic) the mesh access tx cmd is 0x2780 : 01
<0002>11:44:52:679 [INFO]: (Basic) adr_src:0x000a, adr_dst:0x0001, access rx cmd is 0x2880 : 80 28 01 a0
<0003>11:44:52:681 [INFO]: (cmd_rsp)Status Rsp______: 0a 00 01 00 80 28 01 a0
<0004>11:44:52:685 [INFO]: (log_win32)mesh_tx_reliable_stop: op 0x2780 rsp_max 1, rsp_cnt 1
<0005>11:44:54:880 [INFO]: (common)ExecCmd: a3 ff 00 00 00 00 02 01 0a 00 80 27 00
<0006>11:44:54:882 [INFO]: (Basic) the mesh access tx cmd is 0x2780 : 00
<0006>11:44:54:882 [INFO]: (Basic) the mesh access tx cmd is 0x2780 : 00
<0007>11:44:54:999 [INFO]: (Basic) adr_src:0x000a, adr_dst:0x0001, access rx cmd is 0x2880 : 80 28 00 c4
<0008>11:44:55:001 [INFO]: (cmd_rsp)Status Rsp______: 0a 00 01 00 80 28 00 c4
<0009>11:44:55:008 [INFO]: (log_win32)mesh_tx_reliable_stop: op 0x2780 rsp_max 1, rsp_cnt 1
```

Figure 20.11: Relay

Friend: a3 ff 00 00 00 00 02 01 07 00 80 10 01

<0000>11:44:15:977	[INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 01 0a 00 80 10 01
<0001>11:44:15:977	[INFO]:(Basic)the mesh access tx cmd is 0x1080 : 01
<0002>11:44:16:080	[INFO]:(Basic)adr_src:0x000a,adr_dst:0x0001,access rx cmd is 0x1180 : 80 11 01
<0003>11:44:16:081	[INFO]:(cmd_rsp)Status Rsp: 0a 00 01 00 80 11 01
<0004>11:44:16:086	[INFO]:(log_win32)mesh_tx_reliable_stop: op 0x1080 rsp_max 1, rsp_cnt 1
<0005>11:44:24:487	[INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 01 0a 00 80 10 00
<0006>11:44:24:488	[INFO]:(Basic)the mesh access tx cmd is 0x1080 : 00
<0007>11:44:24:599	[INFO]:(Basic)adr_src:0x000a,adr_dst:0x0001,access rx cmd is 0x1180 : 80 11 00
<0008>11:44:24:602	[INFO]:(cmd_rsp)Status Rsp: 0a 00 01 00 80 11 00
<0009>11:44:24:607	[INFO]:(log_win32)mesh_tx_reliable_stop: op 0x1080 rsp_max 1, rsp_cnt 1

#### Figure 20.12: Friend



Proxy: a3 ff 00 00 00 00 02 01 07 00 80 13 01

```
<0000>11:45:14:247 [INFO]: (common)ExecCmd: a3 ff 00 00 00 00 02 01 0a 00 80 13 01
<0001>11:45:14:248 [INFO]: (Basic)the mesh access tx cmd is 0x1380 : 01
<0002>11:45:14:358 [INFO]: (Basic)adr_src:0x000a,adr_dst:0x0001,access rx cmd is 0x1480 : 80 14 01
<0003>11:45:14:362 [INFO]: (cmd_rsp)Status Rsp_____: 0a 00 01 00 80 14 01
<0004>11:45:14:367 [INFO]: (log_win32)mesh_tx_reliable_stop: op 0x1380 rsp_max 1, rsp_cnt 1
```

#### Figure 20.13: Proxy

Or Please refer to "Relay", "Friend", "Proxy" control buttons in section 4.5.3.

#### 20.2.5 Heartbeat setting

CMD-cfg\_hb\_pub\_set\_sig = a3 ff 00 00 00 00 00 00 00 02 00 80 39 01 00 ff 02 01 07 00 00 00

## 20.3 Control Operations

#### 20.3.1 Control Generic model Demo

#### **Test Preparation**

- Provisionner dongle (burn 8258\_mesh\_gw.bin)
- Firmware tool(sig\_mesh\_tool.exe), select tl\_node\_gateway.ini
- Dongle\_2# (burn mesh.bin)
- SDK no need to modify

#### **Test Introduction**

The test is to achieve the control of the Generic model, mainly to achieve the G\_ONOFF\_SET command test.

Test and calculate the response time of CMD send and ACK.

#### **Test Step**

**Step 1** After burning gateway bin into dongle\_1 #, insert it into the computer and open the firmware software sig\_mesh\_tool.exe at the same time

**Step 2** Burn mesh node bin in dongle\_2#.

**Step 3** Firmware add mesh node into the network

**Step 4** Send, or double click in firmware int CMD bar the following command:

CMD-g\_on\_03 = e8 ff 00 00 00 00 00 00 03 00 82 02 01 00

**Step 5** The following information will show after firmware send successfully.

g on 03 <0011>15:53:49:075 [INFO]:(common)ExecCmd: e8 ff 00 00 00 00 02 00 03 00 82 02 01 00 : 03 00 02 00 82 04 00 01 0a <0012>15:53:49:199 [INFO]:(cmd\_rsp)Status Rsp\_ <0013>15:53:49:211 [INFO]: (GATEWAY) HCI\_GATEWAY\_RSP\_OP\_CODE : 91 81 03 00 02 00 82 04 00 01 0a g off 03 <0014>15:54:54:035 [INFO]: (common) ExecCmd: e8 ff 00 00 00 00 02 00 03 00 82 02 00 00 : 03 00 02 00 82 04 01 00 0a <0015>15:54:54:094 [INFO]:(cmd\_rsp)Status Rsp\_ <0016>15:54:54:107 [INFO]: (GATEWAY) HCI GATEWAY RSP OP CODE : 91 81 03 00 02 00 82 04 01 00 0a .. g\_on\_03 <0017>15:54:57:898 [INFO]: (common) ExecCmd: e8 ff 00 00 00 02 00 03 00 82 02 01 00 : 03 00 02 00 82 04 00 01 0a <0018>15:54:57:955 [INFO]:(cmd\_rsp)Status Rsp\_ <0019>15:54:57:969 [INFO]: (GATEWAY)HCI GATEWAY RSP OP CODE : 91 81 03 00 02 00 82 04 00 01 0a .. g\_off\_03 <0020>15:55:01:740 [INFO]: (common) ExecCmd: e8 ff 00 00 00 00 02 00 03 00 82 02 00 00 \_: 03 00 02 00 82 04 01 00 0a <0021>15:55:01:938 [INFO]:(cmd rsp)Status Rsp <0022>15:55:01:952 [INFO]: (GATEWAY) HCI\_GATEWAY\_RSP\_OP\_CODE : 91 81 03 00 02 00 82 04 01 00 0a

Figure 20.14: Firmware send successfully

**Step 6** As shown in above figure <0011>, the interval between CMD sending time and rsp is 199 – 075 = 124ms. Gateway and node are controlled in adv way, the ack reply time is different because of the network.

Status Rsp\_\_\_\_\_: 03 00 02 00 82 04 00 01 0a

The corresponding structure is

Telink

```
typedef struct{
    u16 len; // length
    u16 src; // source address
    u16 dst; // destination address
    u8 data[ACCESS_WITH_MIC_LEN_MAX]; // access layer(op code, parameters)
}mesh_rc_rsp_t;
```

**Step 7** If the destination address of the control is a multicast or broadcast address, the node will add a random delay before replying to the ACK after receiving the command, so that multiple device reply messages are avoided as much as possible. When broadcasting an address as shown in the figure below, the time interval between CMD and rsp is: <0023> 12:390 – 11:752 = 538 ms

```
.. g_off
<0023>16:14:11:752 [INFO]:(common)ExecCmd: e8 ff 00 00 00 02 00 ff ff 82 02 00 00
<0024>16:14:12:390 [INFO]:(cmd_rsp)Status Rsp______: 03 00 02 00 82 04 00
<0025>16:14:12:405 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE
: 91 81 03 00 02 00 82 04 00
.. g_on
<0026>16:14:14:328 [INFO]:(common)ExecCmd: e8 ff 00 00 00 02 00 ff ff 82 02 01 00
<0027>16:14:15:096 [INFO]:(cmd_rsp)Status Rsp______: 03 00 02 00 82 04 00 01 0a
<0028>16:14:15:129 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE
: 91 81 03 00 02 00 82 04 00 01 0a
```

Figure 20.15: Broadcast address

#### 20.3.2 CTL model

#### **Test Preparation**

- Provisionner dongle (burn 8269\_mesh\_gw.bin or 8258\_mesh\_gw.bin)
- Firmware tool(sig\_mesh\_tool.exe), select tl\_node\_gateway.ini
- Dongle\_2# (burn mesh.bin)
- SDK need to modify #define LIGHT\_TYPE\_SEL LIGHT\_TYPE\_CT

#### **Test Introduction**

The test is to achieve the control of the CTL model, mainly to achieve the LIGHT\_CTL\_SET command test.

#### Test Step

**Step 1** After burning gateway bin into dongle\_1 #, insert it into the computer and open the firmware software sig\_mesh\_tool.exe at the same time

**Step 2** Burn mesh node bin in dongle\_2#.

**Step 3** Firmware add mesh node into the network

**Step 4** Send, or double click in firmware int CMD bar the following command:

CMD-light\_ctl\_set = e8 ff 00 00 00 00 00 00 ff ff 82 5e 01 00 20 4e 00 00 00

The following information will show after firmware send successfully.

```
.. light_ctl_set
<0000>16:29:46:909 [INFO]:(common)ExecCmd: e8 ff 00 00 00 00 02 00 ff ff 82 5e 01 00 20 4e 00 00 00
<0001>16:29:47:513 [INFO]:(cmd_rsp)Status Rsp_____: 03 00 02 00 82 60 01 00 20 4e
<0002>16:29:47:521 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE
: 91 81 03 00 02 00 82 60 01 00 20 4e
```

Figure 20.16: Firmware send successfully

#### 20.3.3 HSL model

Hsl model will allocate 3 element addresses after provision. The main element is used for lightness, generic model control and configuration model, element2 is for hue control, element3 is for saturation control.

#### **Test Preparation**

- Provisionner dongle (burn 8269\_mesh\_gw.bin or 8258\_mesh\_gw.bin)
- Firmware tool(sig\_mesh\_tool.exe), select tl\_node\_gateway.ini
- Dongle\_2# (burn mesh.bin)
- SDK need to modify #define LIGHT\_TYPE\_SEL LIGHT\_TYPE\_HSL

#### Test Introduction:

The test is to achieve the control of the HSL model, mainly to achieve the LIGHT\_HSL\_SET command test.

#### Test Step



**Step 1** After burning gateway bin into dongle\_1 #, insert it into the computer and open the firmware software sig\_mesh\_tool.exe at the same time

Step 2 Burn mesh node bin in dongle\_2#

Step 3 Firmware add mesh node into the network

**Step 4** Send, or double click in firmware int CMD bar the following command:

CMD-light\_hsl\_set = a3 ff 00 00 00 00 00 00 ff ff 82 76 01 00 00 50 00 80 00

The following information will show after firmware send successfully

```
<0000>15:57:25:186 [INFO]:(common)ExecCmd: e8 ff 00 00 00 00 02 00 ff ff 82 76 00 20 00 50 00 80 00 00
<0001>15:57:25:210 [INFO]:(GATEWAY) gateway mesh cmd sendback src:0001 dst:ffff,opcode is 7682: 00 20
<0002>15:57:25:506 [INFO]:(cmd_rsp)Status Rsp_____: 02 00 01 00 82 78 00 20 00 50 00 80
<0003>15:57:25:515 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE
: 91 81 02 00 01 00 82 78 00 20 00 50 00 80
```

#### Figure 20.17: Firmware send successfully

#### 20.3.4 Vendor model

Self-defined OP operation

#### **Test Preparation**

- Provisionner dongle (burn 8269\_mesh\_master\_dongle.bin)
- Firmware tool(sig\_mesh\_tool.exe)
- Dongle\_2# (burn 8258\_mesh.bin)
- SDK need to modify, please refer to test introduction

#### SDK Modify Introduction

1) Configure in vendor\_model.h:

#define	VD_USER_ONOFF_GET	0xE1
#define	VD_USERONOFF_SET	0xE2
#define	VD_USERONOFF_SET_NOACK	0xE3
#define	VD_USERONOFF_STATUS	0xE4

2) Declear in vendor\_model.c



3) Add to main\_loop.

#### **Test Introduction**

Telink SDK has the following limitation based on customer usage:

- 1) A friend node can habe 16 LPN at most, the default value is 2, user may modify MAX\_LPN\_NUM to 16.
- The data of 1 LPN node cached by the Friend node can support long packets, and the maximum length of a long packet is 41 bytes. (When the APP sends data, the maximum parameter transmission is 41 bytes.)

#### 20.3.5 Gateway Transmit Long Packet to LPN

#### **Test Preparation**

- Provisionner dongle (burn 8258\_mesh\_gw.bin)
- Firmware tool(sig\_mesh\_tool.exe), select tl\_node\_gateway.ini
- Dongle\_2# (burn LPN.bin)
- SDK need to modify, please refer to test introduction

#### **Test Introduction**

Telink SDK has the following limitation based on customer usage:

- 1) A friend node can have 16 LPN at most, the default value is 2, user may modify MAX\_LPN\_NUM to 16.
- The data of 1 LPN node cached by the Friend node can support long packets, and the maximum length of a long packet is 41 bytes. (When the APP sends data, the maximum parameter transmission is 41 bytes.)

The SDK is modified as following:

- 1) Set DEBUG\_SUSPEND =1 (no Low Power mode);
- 2) Add the following information in cb\_vd\_light\_onoff\_set().

```
u8 debug_fn_reciver_data[64];
u8 debug_fn_cnt;
memset(debug_fn_reciver_data,0,sizeof(debug_fn_reciver_data));
memcpy(debug_fn_reciver_data,par,par_len);
```





```
: u8 debug_fn_reciver_data[64];
: u8 debug fn cnt;
: int Cb_vd_light_onoff_set(u8 *par, int par len, mesh_cb_fun_par_t *cb par)
:
 -{
      debug_fn_cnt++;
2
      int err = -1;
:
      int pub_flag = 0;
      //model_g_light_s_t *p_model = (model_g_light_s_t *)cb_par- >model;
5
      vd_light_onoff_set_t *p_set = (vd_light_onoff_set_t *)par;
÷
      memset(debug_fn_reciver_data, 0, sizeof(debug_fn_reciver_data));
      memcpy(debug_fn_reciver_data,par,par_len);
5
1
      . . . . .
```

Figure 20.18: cb\_vd\_light\_onoff\_set

#### **Test Step**

**Step 1** After burning gateway bin into dongle\_1 #, insert it into the computer and open the firmware software sig\_mesh\_tool.exe at the same time.

Step 2 Burn LPN node bin in dongle\_2#

**Step 3** Firmware add LPN into the network.

**Step 4** Control LPN node by existing vendor\_on/vendor\_off command, LPN can be controlled normally.

Step 5 Lengthen vendor\_on command transmission parameter to 41 byte, then send

CMD-vendor\_on = a3 ff 00 00 00 00 00 00 ff ff c2 11 02 c4 02 01 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

**Step 6** Check debug\_fn\_reciver\_data[64] with tdebug tool after firmware firmware send successfully.

😴 tdebug1.7 xiaodong.zong@	⊉telink-sem	i.com		
🔮 variable name:	🔮 addr	🔮 len	🔮 value	SeeFunctions update tables reset mcu ClrDbgInfo ClrBkpoints UsbReconnect
const_oob_static	0xa6b4	0x10	<i></i>	memory read
🖸 ct_flag	0xa154	0x01	0x01	0:CORE V 1 BYTE V Stop EVK device ok
cur_enc_keysize	0xc158	0x01	0x00	
current_connHandle	0xa178	0x02	0xffff	addr 00 data     auto   tracere   bxpointsh   autobeteet
🖸 debug_fn_cnt	0xd450	0x01	0x00	do comman
debug_fn_reciver_data	0xd50c	0x40	<i></i>	
del_node_delay_ms	0xb34e	0x02	0x0000	0030: 00 00 00 00 00 00 00 00 00 00 00 00 0
G del_node_tick	0xb358	0x04	0x00000000	0000: 01 1e 00 00 00 00 00 00 00 00 00 00 00 00 00
G delta_last.10248	0xae44	0x04	0x00000000	0020: 00 00 00 00 00 00 00 00 00 00 00 00 0
G dev_auth	0xc0f8	0x10		## Memory read Addr d50c: 0000: 00 1f 00 00 00 00 00 00 00 00 00 00 00 00 00
🖸 dev_ck	0xe304	0x10		
G dev_comfirm	0xe4e0	0x10		
🖸 dev_dpk	0xa664	0x40	۲	
🧧 dev_dsk	0xa6c4	0x20		
🧕 dev_edch	0xe560	0x20		## Memory read Addr d50c:
🧧 dev_input	0xa6e4	0x91		0000: 01 21 11 22 33 44 55 61 71 81 00 00 00 00 00 00 00 00 00 00 00 00 00
🧕 dev_mac	0xa158	0x0c		0020: 00 00 00 00 00 00 00 00 00 00 00 00 0
dev_pro_comfirm	0xe4c0	0x10		## Memory read Addr d50c: 0000: 01 22 01 02 03 04 05 06 07 08 09 10 11 12 13 14
🧧 dev_random	0xa6a4	0x10		0010: 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 0020: 31 32 33 00 00 00 00 00 00 00 00 00 00 00 00
G dev_rssi_th	0xa170	0x01	0x80	0030: 00 00 00 00 00 00 00 00 00 00 00 00 0
🖸 dev_salt	0xe4d0	0x10		0000: 01 23 01 02 03 04 05 06 07 08 09 10 11 12 13 14 0010: 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
dev_session_key	0xe2f4	0x10		0020: 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 0030: 47 00 00 00 00 00 00 00 00 00 00 00 00 00
G dev_session_nonce	0xe5c4	0x10	<i>.</i>	. <b>_</b>

# Figure 20.19: tdebug tool

# 21 Summary of mesh\_1.1\_feature

The summary content and profile of mesh V1.1 feature can be found in https://www.bluetooth.com/mesh-feature-enhancements-summary/

The corresponding mesh spec can be downloaded here:

https://www.bluetooth.com/specifications/specs/?types=specs-docs&keyword=mesh&filter=

The link contains the following spec:

(1) Mesh protocol stack spec:

#### Mesh Protocol 1.1 Specification

The mesh protocol stack, which contains data communication format definitions, foundation models, and so on. The foundation models include Configuration model, Health model, Remote Provisioning model, Directed Forwarding Configuration model, Bridge Configuration model, Mesh Private Beacon model, On-Demand Private Proxy model, SAR Configuration model, Solicitation PDU RPL Configuration model, Opcodes Aggregator model, Large Composition Data model. These models are described in later sections. When you need to see the parameters of the command codes of these models, you need to refer to the "message" section in the "4 Foundation models" chapter of this document.

(2) Mesh product model spec:

#### Mesh Model 1.1 Specification

Mesh product models, such as the Generic OnOff model, Lightness model, and so on. When you need to see the parameters of the command codes for these models, you need to refer to the "message" subsection of each model chapter in this document.

(3) Mesh OTA model spec:

Mesh Binary Large Object Transfer Model Mesh Device Firmware Update Model

- (4) NLC spec
- NLC: Networked Lighting Control

Ambient Light Sensor NLC Profile Basic Lightness Controller NLC Profile Basic Scene Selector NLC Profile Dimming Control NLC Profile Energy Monitor NLC Profile Occupancy Sensor NLC Profile

(5) Device attribute definitions

Includes sensor ID, definition of sensor data format, introduction of brightness and time attributes in light control model.

#### Device Properties

(6) Mesh definition of import and export file formats for network sharing

Mesh Configuration Database Profile

# 22 Certify\_base\_provision\_certificate Mode

# 22.1 Function

The certificate-based authentication provisioning mode can be judged by the provisioner and only nodes that meet the requirements can join the network.

The difference between this mode and the normal provisioning mode is that: when provisioning, the provisioner first obtains the certificate of the unprovisioned device, which contains the public key, uuid, and authentication information, etc. When the provisioner gets the certificate, it will judge whether it is legal or not, and if it is legal, it will start to group the nodes. When provisioning, the provisioner does not need to obtain the public key of the other party, and the other processes are the same as those in the normal provisioning mode. The judgement of whether the certificate is legal or not is made only on the provisioner side.

An overview of the functions can also be found in https://www.bluetooth.com/mesh-feature-enhancementssummary/ This SIG is described on the official website as well as https://www.bluetooth.com/bluetoothcertificate-based-provisioning-a-technical-overview/.

# 22.2 Test Using the Code's Default Certificate and Compiling It Directly into Firmware

#### 22.2.1 Code Configuration

- Open CERTIFY\_BASE\_ENABLE。
- CERTIFY\_BASE\_ENABLEcertify\_base\_crypto.c, The CERTIFY\_TYPE inside is changed to CER-TIFY\_OOB\_BY\_DEFAULT\_CERT.

#### Note:

At this point, all nodes are using the same certificate, and the same device UUID.(Mac address may be different)

The scanning interface is as follows:


#### Figure 22.1: Certificate mode

Then directly click "Add" and other operations according to the normal network operation process, you can network and control.

# 22.3 Testing Ways to Use Newly Generated Certificates

### 22.3.1 Code Configuration

- Open CERTIFY\_BASE\_ENABLE
- certify\_base\_crypto.c The CERTIFY\_TYPE inside remains the default CERTIFY\_OOB\_BY\_READING\_FLASH

Install git bash, if git is already installed on your computer. Note that the git bash version should be greater than or equal to version 2.41.0.

#### Note:

To run bash, run the macOS or Linux command or the window git bash by typing . /xxx.bash.

### 22.3.1.1 Open a Git\_bash Terminal

In the telink\_sig\_mesh\_src/sig\_mesh\_tool/bash-certifybase directory, open the git bash terminal as follows:

right mouse button - Git Bash here



#### Telink SIG Mesh SDK Developer Handbook

MINGW64:/c/Users/Admin/Desktop/git/src/telink_sig_mesh_src/sig_me – D ×	📕   🗹 📕 =	bash-cer	ifybase [ master ↓0 ↑0 ]		- 🗆 ×
Admin@DESKTOP-E9K1068 MINGw64 ~/Desktop/git/src/telink_sig_mesh_src/sig_mesh_too 1/bash-certifybase (master) \$	文件     主页       ←     →	共享 <mark>I</mark> « si	查看 g_mesh_tool > bash-certifybase	<ul><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li></ul>	~ <b>?</b> 在 bash-certifyb
	大快速访问	^	名称 ^	修改日期	类型 ^
	「「」「「」」「」」「」」「」」」	*	output-device	2023/10/23 13:57	文件夹
		- i -	📕 output-intermediate	2023/10/19 10:27	文件夹
	1:50	- <u>^</u>	📕 output-root	2023/10/17 17:47	文件夹
	🖻 又档	*	DS_Store	2023/10/20 11:21	DS_STORE 文件
	📰 图片	*	gen-device.bash	2023/10/20 11:17	BASH 文件
	📕 8258_me	sh	🔓 gen-device.config	2023/10/23 13:57	CONFIG 文件
	📕 8258_me	sh_LPI	gen-intermediate.bash	2023/10/19 13:31	BASH 文件
	ellisys bt	t	gen-intermediate.config	2023/10/11 15:18	CONFIG 文件
	周报		gen-root.bash	2023/10/8 14:31	BASH 文件 🗸
		~	<		>
	11 太雨日				9== (==)

Figure 22.2: Open git bash

### 22.3.1.2 Generate Root Certificates

We have a default certificate in our sdk, it is not recommended to regenerate the root certificate, we suggest to skip this step directly. If you want to regenerate the root certificate, you need to import the certificate file root.der from output-root into the app, and set the newly imported certificate as the root certificate. The procedure is as follows:

• To run gen-root.bash to create a root certificate, run the command . /gen-root.bash.



Figure 22.3: Generate root certificates

• Import root certificate to APP



Figure 22.4: Go to the certificates page



### Figure 22.5: Importing a new certificate



Figure 22.6: Importing a new certificate

pubKey: EC 2a:68:3c:fbi X: A 15ae7ba9d5 Y: 9 biff955aa21 Time-NotAft Alg:SHA256 (Default) pubKey: EC	Public Key [80 8e e8 9/9:07b9427b379d; 1534e47755633 10854a707b5633 10864a701b448605472353585a60046 52b947630130356995 5430537556ec46655572828094060337 5430537580eb17655 eform/Wei Oct 11140048 GMT-0800 eform/Wei Oct 11140048 GMT-0800 2035 5WITHECD5A	
a:06:02:c9:9	: Public Key [7e:42:14:9e:79:b7:3f:73:3c:f5:a 95:46:ac:8f:d1:a8] 95:026:m305ff:61:a827953454518026533	
a:06:02:c9:9 X: 3 25f7c98772	95:46.ac.8f.d1:a8] 95:026666d305ffb61d872953d5451802e533 2e27d01ca0613755224 action at: 0	

Figure 22.7: Set the newly imported certificate as root certificate

### 22.3.1.3 Run Gen-intermediate.bash to Create an Intermediate Certificate

The certificate is signed by the root certificate and the result is output in the output-intermediate directory.

Run the command: ./gen-intermediate.bash



Figure 22.8: Creating an intermediate certificate

### 22.3.1.4 Configure Device Certificate Parameters

Change the following parameters in gen-device.config:

CN(common name): This parameter is the device UUID, which should be changed first each time a certificate is generated, because the device UUID of each node cannot be the same.



BPID: This parameter is PID (Product ID) and must be equal to MESH\_PID\_SEL in the firmware sdk code.

BCID: This parameter is CID (Company ID) and must be equal to MESH\_VID inside the firmware sdk code.

Note that the CID and PID in the gen-device.config file are in big-endian byte order.

🔚 gen-device. config🛛 [req] prompt = no distinguished\_name = req\_distinguished\_name [req distinguished name] 6 #Country C=CN 8 #State 9 ST=ShangHai 10 #Organization O=Telink-Semi 11 12 #Organization Unit 13 OU=Telink +common name, need to be replaced to device uuid, CID, PID of unprovision node. 14 CN=001BDC08-1021-0B0E-0A0C-000B0E0A0C00 BCID:0211 BPID:0001 16 #emailAddress 17 emailAddress=support@telink-semi.com 18 [v3\_req] 19 20 authorityKeyIdentifier = keyid subjectKeyIdentifier = hash 21 basicConstraints = CA:FALSE 22 keyUsage = Certificate Sign, CRL Sign 23 24 #TODO: static oob #2.25.234763379998062148653007332685657680359 = DER:31:7a:6f:16:58:44:72:74:15:10:33:62:5a:fb:c4:fl 25 26 certificatePolicies = critical,@pol 27 28 [log] policyIdentifier = 2.16.840.1.101.3.2.1.48.1 29

#### Figure 22.9: Change the device UUID CID PID in gen-device.config

### 22.3.1.5 Run Gen-device.bash to Generate the Device Certificate

The certificate is signed by an intermediate certificate and the result is output in the output-device directory.

Run command: ./gen-device.bash

NINGW64:/c/Users/Admin/Desktop/git/src/telink_sig_mesh_src/sig_me — 🛛 🗙	
Admin@DESKTOP-E9K1068 MINGW64 ~/Desktop/git/src/telink_sig_mesh_src/sig_mesh_too l/bash-certifybase (master) \$ ./gen-device.bash 启动脚本,开始生成device证书 生成device证书 read EC key writing EC key Certificate request self-signature ok subject=C = CN, ST = ShangHai, O = Telink-Semi, OU = Telink, CN = 001BDC08-1021- 0B0E-0A0C-000B0E0A0C00 BCID:003F BPID:001A, emailAddress = support@telink-semi.c om 生成bin文件 start read EC key read EC key cN信息 CN=001BDC08-1021-0B0E-0A0C-000B0E0A0C00 BCID:003F BPID:001A device_uuid=001BDC08-1021-0B0E-0A0C-000B0E0A0C00 计算crc crc计算完成: 8B06	^
生成bin文件 - end 添加readme文件 添加readme文件完成! 脚本执行完成,请查看./output-device 目录	~

Figure 22.10: Generate device certificate

# 22.3.1.6 Burn the Certificate into the Device's Flash

Burn the 4Kbin file generated in the output-device file to the flash address of the device:

Location of FLASH\_ADR\_CERTIFY\_ADR (default is 0x78000)

```
[16:09:53]:
TC32 EVK : Swire OK
Flash Sector (4K) Erase at address 78000
Flash Page Program at address 78000
Flash Page Program at address 78400
Flash Page Program at address 78800
Flash Page Program at address 78c00
File Download to Flash at address 0x078000: 4096 bytes
Total Time: 310 ms
```

#### Figure 22.11: Burning certificates to flash

Once the burning is complete, reboot to perform certify base provisioning.

### 22.3.1.7 Codes Described Below

Telink

T

 <Introduction of code processing flow, non-operational steps> The programme will call the crc16 function to check the location of flash address 78000, the length of the check is f00, to confirm whether the read certificate is complete and whether it has been abnormally modified.

> static u32 val; val = crc16((u8 \*)0x78000, 3840);

#### Figure 22.12: CRC comparison

(2) <Introduction of code processing flow, non-operational steps> The result is compared with the crc value at flash address 78f00, and the expected comparison result is that both are the same.

Ŧ	Download				itä Tdebug ☷ Log windows					
Variable Name	Addr	Len	Value	^	Flash Page Program at address 78c00	^				
tick_rc24mCal	42f80	4	001a7820		File Download to Flash at address 0x078000: 4096 bytes					
tick_scan.5855	45068	4	00000000		lotal lime: 310 ms					
tick_scan.9189	4509c	4	01dcb768		[16:11:43]:					
tid_cache_idx.13211	43e60	4	00000000		TC32 EVK : Swire OK					
total_len.15403	43db8	2	00000000		2 bytes have finished!					
txPower_index	45238	1	000000bf		078f00: 06 8b					
tx_bear_extend_en	44ede	1	00000000		Total Time: 836 ms					
tx_pin_initialed	47138	4	0000001		[16.11.49].					
tx_settle_slave	4305c	4	76564b00		reset mcu					
ui_ota_is_working	43e6e	1	00000000							
update_err_cb	450a4	4	00000000		[16:12:08]: TC32 FVK : Swire OK					
use_mesh_adv_fifo_fr	44f88	1	00000000		2 bytes have finished!					
val.16665	43d6c	4	00008b06							
vd_group_g_func	42f48	24	•••	]	078f00; 06 8b					
vd_onoff_state	44f08	2	00000000		Total lime. 050 ms					
zbit_flash_flag	44f94	1	00000000			*				

Figure 22.13: CRC comparison

# 23 Remote Provision Functional Description and Development Instructions

# **23.1 Remote Provision Function Introductions**

In Sig Mesh Spec V1.0 / V1.0.1, Provisioning requires that the Provisioner and the Provisionee are within one hop of each other, because the unprovision beacon packets cannot be relayed directly, so the command interactions in the provisioning process cannot be relayed.

In order to add nodes beyond one hop to the network, Sig Mesh V1.1 adds Remote Provision function.

Remote provision also adds nodes one by one when provisioning, but there is a relay function so you can add more distant nodes into the network.

Important Application Scenario: After adopting Remote Provision, when the host (Provisioner) is not convenient to move, it is also possible to realise that the mesh nodes can be arranged according to the actual scenario in the application first, and then the network can be formed. Especially the application scenario with gateway.

In addition to remote provisioning, remote provisioning can also be used for updating Device Key, Node Address and Composition Data, as described in "3.11.8 Node Provisioning Protocol For details, see"3.11.8 Node Provisioning Protocol Interface procedures" in V1.1 spec.

Note: The above functional overview can also be found in the description on the SIG official website https://www.bluetooth.com/mesh-feature-enhancements-summary/ and https://www.bluetooth.com/mesh-remote-provisioning/.

## 23.1.1 Introduction to Remote\_provision Network Interaction Process

(The whole provisioning process is also done node by node).

- (1) First network one or more nodes within one hop of the host (Provisioner).
- (2) Scan for unprovision beacons sent from nodes further away (within the second hop range) through already networked nodes and report them to the Provisioner.
- (3) Provisioner selects a node that reports an unprovision beacon (for example, networked node A reports a scan to unprovisioned node B).
- (4) When Provisioner networks node B, it encapsulates the message to be sent to node B into a mesh network message and sends it to node A first, and then node A extracts the network information and sends it to the un-networked node B in the form of a generic provision PDU (either in the form of a PB-ADV, or a PB-GATT).
- (5) The message that node B replies to the Provisioner will be sent to node A first. Then node A encapsulates the message into a mesh network message and send to the Provisioner.
- (6) Steps 4 and 5 are executed several times until the networking is completed. During this process, for the unprovisioned device B, it can be considered that node A does not exist, and there is no difference with the normal provisioning mode.



(7) Finish provisioning the nodes in the second hop range by repeating steps 2 to 6. Then search and network the nodes in the third hop in the same way..... until the search fails to find any unprovision nodes.



Figure 23.1: The Architecture Of Remote Provisioning

### 23.1.2 Remote Provision Opcode and Flowchart

Remote provisioning opcode (see "4.3.4 Remote provisioning information" in V1.1 spec for command parameters).

т	Te	eli	n	k
310				

Element	Model Name	State	Message	Rx	Тх
Remote Remote Provisioning Main Server	Remote Provisioning	Remote Provisioning Scan Capabilities Get		-	
	Scan Capabilities	Remote Provisioning Scan Capabilities Status	-	М	
		Remote	Remote Provisioning Scan Get		-
	Provisioning	Remote Provisioning Scan Start		-	
	Scan Parameters	Remote Provisioning Scan Stop	М	-	
		Remote Provisioning Scan Status		М	
		Remote Provisioning Scan Report	-	М	
			Remote Provisioning Extended Scan Start	М	-
		Remote Provisioning Extended Scan Report	-	М	
		Remote Provisioning Link Get	М	-	
		Remote Provisioning Link Open	М	-	
			Remote Provisioning Link Close	М	-
		Remote	Remote Provisioning Link Status	-	М
		Provisioning Link	Remote Provisioning Link Report	-	М
		Parameters	Remote Provisioning PDU Send	М	-
			Remote Provisioning PDU Outbound Report	-	М
		Remote Provisioning PDU Report	-	М	

Table 4.334: Remote Provisioning Server model messages

Figure 23.2: MD\_REMOTE\_PROV\_opcode

Introduction to remote\_provision network interaction process The remote scan flowchart in step 2: (where capa get is Remote Provisioning Scan Capabilities Get, you can get the maximum number of unprovisioned nodes that can be scanned by the current mesh node and whether active-scan is supported).





Figure 23.3: MD\_REMOTE\_PROV\_scan\_flow

Introduction to remote\_provision network interaction process Flowchart of remote provision for step 3/4/5:



Figure 23.4: MD\_REMOTE\_PROV\_provision\_flow

# 23.2 Testing Remote Provisioning with the App

### 23.2.1 Test Conditions

8258 dongle greater than or equal to 2 (burn 8258\_mesh.bin), Android or iOS SIG Mesh App.

### 23.2.2 Firmware SDK Code Configuration

In the default configuration, the RPR feature is turned off. To enable it, you need to turn on the MD\_REMOTE\_PROV macro switch on the node side in the mesh\_config.h file. This is shown in the following figure:



Figure 23.5: 打开 MD\_REMOTE\_PROV\_App

### 23.2.3 App Settings

App Home Click - Settings - Preset Mode, select Remote Provision

Settings		
Enable Private Mode(Default Bound)	()	0
Provision Mode:		0
onormal(selectable)		
normal(auto)		
remote provision		
fast provision		
Enable subscription level service model ID	()	0
Extend Bearer Mode		0
No Extend		
Extend GATT Only		
Extend GATT & ADV		
Use No-OOB Automatically	()	
		0
Share Import Complete Action	_	0

### 23.2.4 Test Steps

(1) Tap the "+" sign on the home page of the App to enter the Remote Provisioning page. Then it will start the automatic provisioning.

Before starting automatic provisioning, the app will judge whether the current app is in GATT connected state with the networked node that supports Remote Provision, if not, it will carry out normal PB-GATT networking, if yes, it will carry out remote provisioning to other un-networked nodes through this networked node. As shown in the figure below, the first one (top left) is networked by normal PB-GATT, and the others (top right and bottom left) are networked by remote provision.





(2) When the timeout has not been scanned for un-networked nodes, it indicates that all nodes have been networked, and then returns to the home page to display the following:





Figure 23.8: MD\_REMOTE\_PROV\_App\_provision\_success

# 23.3 Gateway Remote Provision Host Computer Development Guide

### 23.3.1 Code and Tool Parameter Configuration for Gateway's Remote Provision

Test conditions: 1 x 8258 dongle (burning 8258\_mesh\_gw.bin), 2 x 8258 dongles (burning 8258\_mesh.bin)

(1) MD\_REMOTE\_PROV is turned on.

624: /**	
625: * MD REMOTE PROV: model of remote prov	vision.
626: * refer to "4.4.5 Remote Provisioning	Server model" and "4.4.6 Remote Provisioning Client model" of spec "MshPR
627: */	с
628: <b>#ifndef MD REMOTE PROV</b>	
629: <b>#if</b> (WIN32)	
630: #define MD REMOTE PROV 1	
631: #elif (MESH USER DEFINE MODE == MESH IR	RONMAN MENLO ENABLE)
632: #define MD REMOTE PROV	// default_disable
633: <b>#elif</b> (MI API ENABLE)	
634: #define MD REMOTE PROV 0	// must 0
635: #elif ( PROJECT MESH )	
636: #define MD_REMOTE_PROV (1)	// dufault disable
637: #elif ( PROJECT MESH PRO )	
638: #define MD REMOTE PROV	// dufault disable
639: <b>#else</b>	
640: #define MD_REMOTE_PROV 0	<pre>// must 0, other project not support due to low power application.</pre>
641: <b>#endif</b>	
642: <b>#endif</b>	
612:	

Figure 23.9: Open MD\_REMOTE\_PROV

- (2) If you want to improve the efficiency of the distribution network, you can set the macro EX-TENDED\_ADV\_ENABLE to 1, which means that you can use the extended broadcast packet mode to send remote provision messages. Note that this mode is private.
  - Enable the Macro in code.



Figure 23.10: Open EXTENDED\_ADV\_ENABLE

• Check whether the circled code below has been added, and if not, add it.



• The host computer sets the gateway to extended broadcast packet mode.

The handling of the 3 modes of the "Extend Adv" control is detailed in the is\_not\_use\_extend\_adv() function.

Telink

T

Telink sig_mesh Not Found V3.3.3.5	
CMD tl_node_gateway.ini INI BULKOUT ASCII	Log T AutoSaveLog 2 retry Clear Save Save V Hex Adv Stop Scan rp_scan ex_scan OTA Rx t
LPN_get_level LDN_get_onoff lightness_get_Panel retry count of LPN_fw_distrib_ota_start is timeout LPN_fw_distrib_ota_start_04	Tastbind     Extend Adv:     OTA Only       None     OTA     OTA       OTA     OTA     OTA
<pre>fw_update_info_get fw_update_info_get fw_distribution_getthese distribution start not for LPN. For LPN, plea fw_distribution_start_all fw_distribution_start_0002 fw_distribution_start_0001 fw_distribution_start_0001</pre>	None: no extend adv OTA Only:only mesh OTA use extend adv All: all mesh message use extend adv

scheduler_get							
sched_action_get							
sched_action_set_off							
sched_action_set_on							
sched_action_set_scenel							
time_set							
time_get							
time_zone_set							
time_zone_get	<						
time_delta_set	1			_			
time_delta_get	~		ALL	<ul> <li>chn_s</li> </ul>	et isconne	input_db	Path: E:\git_lab\telink_: OpenFil
	_						
es ff 00 00 00 02 00 01 00 b6 0b 00 c0	🗌 d	lirected	COM18	<ul> <li>UAR</li> </ul>	r USB	output_db	GwReset GwMeshOta GwOtaSelf Pro
1							

Figure 23.12: Gateway settings extended broadcast packets

#### Note:

Telink

fw\_distribution\_suspend fw\_distribution\_cancel fw\_update\_metadata\_check fw\_update\_statt fw\_update\_statt fw\_update\_apply blob\_transfer\_get blob\_transfer\_statt blob\_transfer\_cancel blob\_block\_start blob\_block\_get blob\_block\_get

fw\_distribution\_suspend

-distribution start end

T

If the gateway dongle is accidentally powered off, the gateway will be set to "GATT Only" mode by default, and you need to click "Extend Adv" again to set the gateway to Extended Broadcast Packet Mode (All), or turn off and then turn on the host computer again, which automatically refreshes the current "Extend Adv" mode of the host computer to the "Extend Adv" mode of the gateway dongle.

### 23.3.2 Phase 1 Network One or More Nodes in Normal pb\_adv Style

When the network is empty and there is no networked device, one or more nodes within one hop of the gateway need to be networked first by ordinary PB-ADV.

Provisioning steps: connect to the pc via qw dongle, open tools tool sig\_mesh\_tool.exe, select the ini file corresponding to gw as shown in the figure below, and provision one/multiple mesh nodes with pb\_adv, refer to the sig mesh handle book gateway project for the specific process: "Provisioner operation and APIs".

Mesh

Close

8 Telink sig_mesh Found V3.3.3.5	- 🗆 X
CHD <b>bl_node_gateway.ini v</b> INI BULKOUT ASCII	Log T AutoSaveLog 2 retry Clear Save Save JV Hex T Adv Stop Scan rp_scan ex_scan OTA Rx test
LPN getl_node_gateway.ini	fastbind Extend Adv: None
LPN get_onoff	
lightness get Panel	. BO AS 13 DI JE DE 33 SE 03 EO ED EL AD LE UL VA
Note:retry count field of LPN distrib start is change	-10//10/10/00/00/10/10/00/10/10/00/00/00/
LPN fw distrib ota start 04	: 3G C1 79 30 C1 84
	<pre>&gt;1000&gt;10.17.00.000 [INTO]. (common) gateway json into state</pre>
fw update info get	<pre>&gt;l000/13:17:00:324 [INFO]:(COMMON)HOGE_IGX=0, VC_ULIG=: a0 22 13 DI 96 D6 93 36 03 65 6D eF 2D F6 GI 0 &gt;l0100/13:17:00:636 [INFO]:(CATENA)/under mult makes with the first</pre>
fw update info get all	1010-10.17.00.555 [INCO]. (WILERAL) Update gw 5 netwey IIISt
fw distribution get	. OI 12 10 00 00 00 00 10001 (NETWING OF THE NAME OF DEAD NOT VOV : 46 66 20 E1 72 74 85 62 63 64 45 45
Note: these distribution start not for LPN. For LPN, p	VIOIS-13.17.00.535 [INFO]. (WITEWAI)RCT_UNTEWAI CHD_SEND_RET_RET_ET : 5 11 20 01 /2 /0 00 35 35 3C 40 40
fw distribution start all	<1012/13:17:00:333 [INFO]: (WAILART) NCL_WAILART_LCH_SLI_NOUL_PARK : 05 FT 10 00 (1012):17:00:643 [INFO]: (CAILART) NCL_WAILART LCH_SLI_NOUL_PARK : 05 FT 10 00
fw distribution start 0002	1013-13.17.00.543 [INFO]. (WAIEWAI) the gateway Extend Adv option 15.0
fw distribution_start_02_03	-1014512-17-22-161 (INFOL- (CATEWAY)HOT CATEWAY CMD SET NORE DADA - as 46 10
fw distribution start 0001	VIOLEVIAVIA VIOLEVIA VIOLI
distribution start end	>1010/13:17.32:100 [INE0]: (WAILAWAI) the gateway used is
fw distribution suspend	. BO as is billed by so so to the is billed and in the solution of the solutio
fw distribution cancel	<pre>&gt; 24 of 05 20 at a4</pre>
fw update metadata check	. 34 (1 20 30 (1 20 1)) (1 100). (normal) actions is a late that
fw update get	<pre>&gt;101/&gt;10.17.02.100 [INFO]: (common) gateway ]son into start (010/01/2017/02.100 [INFO]: (</pre>
fw update start	<pre>&gt;1016/13:17:32:170 [INFO]:(COMMON):COMMON/COMMON COMMON COM COMMON COM COMMON COM</pre>
fw update cancel	-1015-10-11-32-17-1 [INFO]: (WILEWAI) update gw 5 netwey 11150
fw update apply	: DI 12 10 00 33 33 30 46 40 DV DA DA CI GI GI GA
blob transfer get	VIOLOVIO.IT.SS.ITI [INFO]. (VALEMAI NOT UNLEMAI ALL SELLAD SELLADI ESI ESI ESI ESI A COSSISSIS E ACAC
blob transfer start	VIO21/13/17/32/17/ [INFO]: (GATEWAI) NCI GATEWAI (ID) SEI NODE MARA : 85 FF 16 00
blob transfer cancel	<1022>13:17:32:104 [INFO]: (GALEWAI) the gateway Extend Adv option 15:0
blob block start	AND AND A THE AT A CALL AND A CALL
blob chunk transfer	<1023/13:17:47:519 [INFO]: (GAIEWAT)RCL_GAIEWAT_LOT_SEL_NODE_PARK : 05 IT 10
blob block get	1024-10.11.47.027 [INFO]. (MAISMAI) the gateway und is
blob info get	: 80 22 13 D1 70 D6 73 30 03 05 05 07 20 FC 01 02
	-1020-1017.47.020 [INFO]: (WILEWAI)the gateway mac add is
scheduler get	. ON GL 20 OF GL BT
sched action get	<pre>&gt;1026&gt;10.17.47.020 [INFO]: (common)gateway json into state &lt;1026&gt;10.10.10.47.000 [INFO]: (common)gateway json into state &lt;1026&gt;10.10.10.10.000 [INFO]: (common)gateway json into state &lt;1026&gt;10.10.10.10.10.000 [INFO]: (common)gateway json into state &lt;1026&gt;10.10.10.10.10.10.10.10.10.10.10.10.10.1</pre>
sched action set off	<pre>&gt;1020/13:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 0D 01 2D 10 01 &gt;1020013:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 0D 01 2D 10 01 &gt;1020013:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 0D 01 2D 10 01 01 &gt;1020013:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 0D 01 2D 10 01 01 &gt;1020013:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 0D 01 2D 10 01 01 &gt;1020013:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 0D 01 2D 10 01 01 &gt;1020013:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 0D 01 2D 10 01 01 &gt;1020013:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 0D 01 01 01 &gt;1020013:17:47:525 [INFO]: (COMMON) HOGE LOR=), VC_ULIG=: a0 22 13 D1 90 D6 93 30 03 05 00 00 00 00 00 00 00 00 &gt;10000000000000</pre>
sched action set on	1020-10.11.47.025 [LHE0]. (WILERAL) UPLOE W S HELRY LIEL
sched action set scenel	: D1 /2 /C 00 73 73 72 46 46 DV D4 D4 C/ C1 C1 C1 C4 /1026/19.17.47.526 (THUR).//DATENAN/UNT CATENAN CAN CENT MET MEY
	-1045-10.17.47.025 [INFO]. (VALEMAI NOT UNLEMAI ALL DELUSION DEL DEL SE LE 20 01 /2 /2 00 50 50 50 80 40 40
time_set	1000-10-17-17-25 [AFC9]. (ATEMA) ACT GALERAL CLE DEL TOPE TARA . ET 11 000
time_get	1031/13.17.47.843 [INFO]. (MALEWAI) the gateway Extend Adv Option 15.0
time_zone_set	
time_zone_get	
time_delta_set	
time_delta_get v	ALL Chn set connect input db Path- C:\Users\ZB\Desktol OnenFile Mesh
e8 ff 00 00 00 00 02 00 02 00 b7 03 11 22 33 44 55 66 77 8	directed TUART USB output db Outpacet Outpace16 Prov Clase
1	GRAFAR GARARTER GARARTER CONTRICTOR

Figure 23.13: Setting the gateway mode

After successful provisioning, click "MESH" button to open the mesh control interface, and automatically send LIGHTNESS\_GET all command to get all the current nodes (you can also send ONOFF\_GET), and display in the UI interface. If you are already in the mesh control interface, and the UI interface does not show any node that supports remote provision, you need to click the "Nodes" button, and this command also sends the LIGHTNESS\_GET all command.

The LIGHTNESS\_GET all command has the following format:

HCI\_CMD\_GATEWAY\_CMD + netkey index(2 bytes) + appkey index(2 bytes) + retry cnt + rsp\_max + gateway addr + op + par

i.e.: e8 ff + netkey index + appkey index + retry cnt + rsp\_max + gateway addr + 82 02 00 00

Note: The ini format is hexadecimal throughout this document unless otherwise noted.

### 23.3.3 Stage 2 Remote Provision Add Light

(1) Click the rp\_scan button in the figure and set the limit and timeout parameters (see V1.1 spec "4.3.4.4 Remote Provisioning Scan Start" for details).

limit: Maximum number of scanned items to be reported. Value O indicates no limit.

timeout: Time limit for a scan (in seconds)

Then click Confirm and the tool will issue the scan start command on the list of nodes obtained in the previous step in the format:

HCI\_CMD\_GATEWAY\_CMD + netkey index(2 bytes) + appkey index(2 bytes) + retry cnt + rsp\_max + scan server addr + op + par

i.e. e8 ff + 00 00 + 00 00 + 02 + 01 + scan server addr + 80 52 + limit + timeout

8 Telink sig_mesh Found V3.3.3.6	>	<
CMD tl_node_gateway.ini • INI BULKOUT ASCII	Log T AutoSaveLog 2 retry Clear Save Save V Hex Adv Stop Scan rp_scan ex_scan OTA Rx tes T fastbind Extend Adv: OTA Only 🗸	t
LFN_get_onoff lightness_get_Panel Note:retry count field of LFN distrib start is change LFN_fw_distrib_ota_start_04 		^
fw_update_info_get fw_update_info_get_all fw_distribution_get	remote scan para X	
Note:these distribution start not for LPN. For LPN, p fw_distribution_start_all fw_distribution_start_0002 fw_distribution_start_0002	limit 0	
fw_distribution_star_02_04 fw_distribution_start_0001 distribution_start_end fw_distribution_suspend	timeout 4	
fw_distribution_cancel fw_update_metadata_check fw_update_get	gec_mic 确定	
fw_update_start fw_update_cancel fw_update_apply		
blob_transfer_get blob_transfer_start blob_transfer_cancel blob block start		
blob_chunk_transfer blob_block_get blob_info_get		
sched_action_get sched_action_get		
sched_action_set_scenel		
time_set time_get time_zone_set		~
time_zone_get time_delta_set time_delta_get v	ALL	
	directed UART USB output_db GwReset GwHeshOta GwOtaSelf Prov Close	•

Figure 23.14: Remote\_provision add lights

The node receives a scan start and replies with a scan status in the format of:

len(2 bytes) + TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_RSP\_OP\_CODE + scan server addr + gateway addr + op + par.

i.e.: len + 91 + 81 + scan server addr + dst addr + 80 54 + par.

Note: If the rp\_scan button is clicked with the error message shown below:

	Telink master Found V4.1.0.0	□ ×
9. Ja	CMD sig_mesh_master.ini V INI BULKOUT ASCII V Log AutoSaveLog 2 retry Clear Save Save V Hex Adv Stop Scan rp_scan ex_scan LPN_get_level LPN_get_onoff lightness_get_Panel Note:retry count field of LPN distrib start is change LPN_fw_distrib_ota_start_04	OTA Rx test



The error message means that there is not any node was found to be provisoned and supports remote provision function. so there is no way to do remote provision. it may happen when the action mentioned in the previous step, "Click the 'MESH' button to open the mesh control interface", has not been done. if that happen, click the "MESH" button to fix this error.

(2) To specify one or more nodes to SCAN and report unprovision beacon sending by unprovisoned node via the REMOTE\_PROV\_SCAN\_REPORT message in the format.

len(2 bytes) + TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_RSP\_OP\_CODE + src addr + gateway addr +
op + rssi + uuid + oob info(2 bytes) + uri hash(2 bytes, optional).

Telink



i.e.: len + 91 + 81 + src addr + gateway addr + 80 55 + rssi + uuid + oob info.

Note: In order to facilitate the upper computer to display the mac, when the device opens the remote provision function, the last 6 bytes of the device uuid in the sdk are set to the mac address as default. see the processing inside the uuid\_create\_by\_mac() for details. As shown in the figure below:

BULKOUT ASCII	Log T AutoSaveLog 2 retry Clear Save Save F Hex Adv Stop Scan rp_scan ex_scan OTA T fastbind Extend Adv: None	Rx test
tart is change	<pre>(common)VC send to gateway is: e8 ff 00 00 00 00 02 00 02 00 80 52 02 04 (GATEWAY)cmd sendback src:0x0001 dst:0x0002,op 5280(REMOTE_PROV_SCAN_START): 02 04 (RemotePro)CLIENT:REMOTE_PROV_SCAN_STS: 00 01 02 04 (GATEWAY)HCI_GATEWAY_RSP_OP_CODE 00 01 02 04</pre>	^
	<pre>(RemotePro)CLIENT:REMOTE_PROV_SCAN_REPORT(rssi:-13dB): f3 16 97 a5 b5 f7 66 51 3e b7 24 b0 be 45 (GATEWAY)HCI_GATEWAY_RSP_OP_CODE f3 16 97 a5 b5 f7 66 51 3e b7 24 b0 be 45 38 c1 a4 00 00 (RemotePro)CLIENT:REMOTE_PROV_SCAN_REPORT(rssi:-54dB): ca 9d 95 e2 e3 da c1 34 38 84 55 cf 93 f7 (GATEWAY)HCI_GATEWAY_RSP_OP_CODE ca 9d 95 c2 c3 da c1 34 38 64 55 cf 93 f7 66 bf c8 00 00</pre>	38 cl 66 bf
PN. For LPN, p	Ca 50 55 e2 e3 da CI 54 56 64 55 CI 55 I/ 66 DI ao 00 00	
end 2 Device 3	Scaned	×
16 97 9d 95	a5 b5 f7 66 51 3e b7 24 b0 be 45 38 cl a4 -13 dBm e2 e3 da cl 34 38 84 55 cf 93 f7 66 bf a8 -54 dBm	

Figure 23.16: Remote provision nodes

(3) Double-click the device to be provisioned in the scanned device list, the corresponding command is:

```
HCI_CMD_GATEWAY_CTL + HCI_GATEWAY_CMD_RP_LINK_OPEN + scan server addr + uuid
```

i.e.: E9 FF + 1A + scan server addr + uuid.

After the gateway receives HCI\_GATEWAY\_CMD\_RP\_LINK\_OPEN, it sends the Remote Provisioning Link Open command to the scan server node, and the scan server node replies Remote Provisioning Link Status to the gateway after it receives it, and the format of Remote Provisioning Link Status is as follows:

len(2 bytes) + TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_RSP\_OP\_CODE + scan server addr + gateway addr + op + par.

i.e.: len + 91 + 81 + scan server addr + gateway addr + 80 5B + par.

As shown below, double-click on the device that needs provisioning:



2 De	vice	Scar	ned																					×	^
16 9d	97 95	a5 e2	b5 e3	f7 da	66 cl	51 34	3e 38	b7 84	24 55	b0 cf	be 93	45 £7	38 66	cl bf	a4 a8	-13 -54	dBm dBm	]			_				
																				dou	ble	clic	k		
																									ľ
	Con	nec	t																				Cance	1	
										$\overline{}$															

Figure 23.17: Double click nodes

(4) Click the "Prov" button to enter the Provision interface, the corresponding command for the "Prov" button is:

HCI\_CMD\_GATEWAY\_CTL + HCI\_GATEWAY\_CMD\_GET\_PRO\_SELF\_STS.

i.e.: e9 ff + Oc.

Click "Prov" button to enter the Provision screen as shown below:



🛞 Telink sig\_mesh -- Found V4.1.0.0

CMD tl_node_gateway.ini  INI BULKOUT ASCII  Log AutoSaveLog 2	retry Clear Save Save V Hex Adv Stop Scan rp_scan ex_scan OTA Rx test
LPN_get_level , Level	00 80 52 02 04 ^
LEN function in Fast prov mode	Static appkey_idx(little) 00 00
CDTP_GTS_GAX SetPro_internal	app_key 60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48
Image: second	bind_all b5 f7 66 51 3e b7 2
fv_distribut     newcy_lux(nue)     00 00     nv_index(big end)       fv_distribut     fv_distribut     nv_index(big end)     00 00	<pre>s_prov_link_open) :</pre>
fw_distribut     Provision     KEY_REFRESI     DKRI_RP       fw_update_m	
tw_update_st fw_update_st fw_update_stfilter_operation	
blob_transf blob_transf blob_transf blob_transf	17 66 £5 96 30 cl a
blob_chunk t blob_block c blob_info_ge	
sched_action_set off	2
sched_action_set_on sched_action_set_scenel	1
time_set <	thn_set connect input_db Path:MenFile Mesh
directed	UART USB output_db GwReset GwMeshOta GwOtaSelf Prov Close

Figure 23.18: Click prov

The gateway receives this command and returns whether there is already configuration information and the number of elements address occupied by gateway itself. The gateway configuration information command format is.

len(2 bytes) + TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_CMD\_PRO\_STS\_RSP + provision\_flag +
pro\_net\_info.

i.e.: len + 91 8b + provision\_flag+ pro\_net\_info.

A provision\_flag of 1 indicates that the gateway is provisioned. O indicates that network information needs to be set, this case please refer to "Add Light via Provisioner".

The number of elements command format for the gateway itself:

len(2 bytes) + TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_CMD\_SEND\_ELE\_CNT.

i.e.: len + 91 + 8C + gateway element count.

(5) Clicking the "provision" button to trigger the addition of a light, corresponding to the command:

HCI\_CMD\_GATEWAY\_CTL + HCI\_GATEWAY\_CMD\_RP\_START + provision data.

i.e.: E9 FF + 1B + provision data.

 $\times$ 

provision

Fast prov mode SetPro_internal	Static           appkey_id×(little)         00 00           app_key         60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48
network_key ae f9 03 03 0d 17 21 21 2e 37 37 41 41 4b 4b 55	bind_all
netkey_idx(little) 00 00 iv_index(big end) 00 00 01	
iv_update_flag 00 unicast_addr(little)04 00	
Provision     KEY_REFRESI     DKRI_RP       NODE ADD     02.00     condition	
filter_operation filter_type white_list v filter_data 01 00 ff ff SetFilter Add_mac RM_mac	

# Figure 23.19: Trigger adding light

(6) The device provisioning status is reported when provisioning is complete. The format is:

TSCRIPT\_GATEWAY\_DIR\_RSP+HCI\_GATEWAY\_CMD\_PROVISION\_EVT+ gateway\_prov\_event\_t.

i.e.: 91 + 89 + gateway\_prov\_event\_t.

✓       Log       AutoSaveLog       2       retry       Clear       Save       ✓       Hex       Adv       Stop       Scan       rp_scan       ex_scan       OTA       Rx       test
▲ fastbind Extend Adv: None ▼
4c 17 c2 5b 10 4d 0d a6 4f 60          <0039>14:58:17:781 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 5d80(REMOTE_PROV_PDU_SEND): 0         <0040>14:58:18:900 [INFO]: (RemotePro) REMOTE_PROV_PDU_REPORT, inbound num is 5: 05         <0041>14:58:18:900 [INFO]: (GATEWAY) HCI_GATEWAY_RSP_OP_CODE         : 91 81 02 00 01 00 80 5e 05         <0042>14:58:19:118 [INFO]: (common) =============== GATEWAY >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
: 91 81 02 00 01 00 80 5f 04 06 e0 3b f6 f1 0f f9 68 22 de b6 16 8d 62 fe aa 52 11 4f 34 02 b4 ce 4b f7 b7 17 64 19 02 8f 1b 78 <0049>14:58:19:180 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 5d80(REMOTE_PROV_PDU_SEND): 0 <0050>14:58:20:413 [INFO]: (RemotePro)REMOTE_PROV_PDU_REPORT, inbound num is 6: 06 <0051>14:58:20:413 [INFO]: (GATEWAY)HCI_GATEWAY_RSP_OP_CODE : 91 81 02 00 01 00 80 5e 06 <0052>14:58:20:613 [INFO]: (common) ========= GATEWAY >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<pre>&lt;0056&gt;14+59:20:660 [INFO]:(GATEWAY)HCI_GATEWAY_CMD_PROVISION_EVT : 91 89 01 04 00 b0 be 45 38 c1 a4 16 97 a5 b5 f7 66 51 3e b7 24 b0 be 45 38 c1 a4 &lt;0057&gt;14:50:30:C00 [INFO]:(log_win02)json_add_nct_info_doc &lt;0058&gt;14:58:20:705 [INFO]:(RemotePro)REMOTE_PROV_PDU_OUTBOUND_REPORT data is : 05 08 &lt;0059&gt;14:58:20:705 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE</pre>
: 91 81 02 00 01 00 80 5f 05 08 <0060>14:58:21:109 [INFO]:(RemotePro)CLIENT:REMOTE_PROV_LINK_STS: 00 04 <0061>14:58:21:109 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE : 91 81 02 00 01 00 80 5b 00 04 <0062>14:58:21:328 [INFO]:(RemotePro)CLIENT:REMOTE_PROV_LINK_REPORT: 08 00 00 <0063>14:58:21:328 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE : 91 81 02 00 01 00 80 5c 08 00
<
ALL     chn_set     connect     input_db     Path:     OpenFile     Mesh
directed     ▼     UART     USB     output_db     GwReset     GwMeshOta     GwOtaSelf     Prov     Close

Figure 23.20: Provisioning status

```
typedef struct{
    u8 eve;//1 indicates success
    u16 adr;
    u8 mac[6];
    u8 uuid[16];
}gateway_prov_event_t;
```

(7) Bind app\_key

After Provisioning is complete, you also need to bind the app\_key for the model. click "bind\_all" button to bind the app\_key for the model. the process is the same as for non-remote provisioning modes.

a. The corresponding command for bind\_all is: HCI\_CMD\_GATEWAY\_CTL+ HCI\_GATEWAY\_CMD\_START\_KEYBIND
 + fast\_bind + app\_key index(2 byte)+app\_key(16 bytes).



🚷 Telink sig_mesh F	ound	2
CMD [l_node_gat gateway_provision gateway_provision gateway_set_appk gateway_set_appk gateway_set_appk gateway_set_appk gatl_off g_mac_off_unrel g_mac_off_unrel g_all_level_32767 g_all_level_32767 g_all_level_32767 g_all_level_32767 g_all_level_32767 g_all_level_32767 g_all_level_32767 cfg_sec_nw_bc_get cfg_sec_nw_bc_get cfg_sec_nw_bc_get cfg_sec_nw_bc_get cfg_ttl_get cfg_nw_transmit_s cfg_relay_set cfg_relay_set cfg_fined_set cfg_fined_set cfg_fined_set cfg_gatt_proxy_se cfg_gpub_get_sig cfg_pub_get_sig	provision           SetPro_internal         test_cmd           provision         send           network_key         22 22 c2 c3 c4 c5 c6 c7 d8 d9 da db dc dd de df           key_index         00 00         iv_index           iv_update_flag         00         unicast_adr           filter_operation         filter_type         white_list v           filter_type         Add_mac           RM_mac	Static aky_idx 00 00 app_key 60 96 47 71 73 4f bd 76 e3 b4 05 19 d1 d9 4a 48 bind_all start app_key bind, button disable after success
cfg_sub_dd_sig cfg_sub_del_sig cfg_sub_del_all_s cfg_sub_ow_sig cfg_sub_ow_sig cfg_set_level 	ig : 04 00 00 00 11 02 01 0 <1357715:10:53:940 [INFO : 04 00 01 00 80 20 20 <1357715:10:53:940 [INFO : 04 00 01 00 80 3e 02 0 <1357715:10:53:942 [INFO : 01 <1357715:10:53:922 [INFO : 01 <1357715:10:54:005 [INFO : 00	USB UART Connect Prov Mesh Close

i.e.: e9 ff + Ob + fast\_bind + app\_key index(2 byte)+app\_key(16 bytes).

Figure 23.21: Bind app\_key

When fast\_bind is 1: the gateway will only issue appkey add, the provisioned device needs to turn on the default binding function (PROVISION\_FLOW\_SIMPLE\_EN is set to 1).

When fast\_bind is 0: the gateway binds all the model ids by default, in order to save time, users can choose the model ids that need to be bound. open the macro MD\_BIND\_WHITE\_LIST\_EN on the gateway side, and the model ids that need to be bound can be seen in the master\_filter\_list in the Mesh\_common.c file. [], users can modify it as needed.

b. App\_key bind process gateway will call u8 gateway\_model\_cmd\_rsp(u8 \*para,u8 len ) to return the status information of the bound model, the format is:TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_RSP\_OP\_CODE + Parameters.

i.e.: 91 + 81 + appkey bind status

c. App\_key bind completes and returns HCI\_GATEWAY\_CMD\_KEY\_BIND\_EVT indicating success or time\_out. in the format of:

TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_CMD\_KEY\_BIND\_EVT +result.

i.e.: 91 + 8a + result.(1:success 2:time\_out).

(8) Repeat steps (1)~(7) to perform remote provision for other nodes one by one until all nodes are networked.

# 24 Mesh OTA and Guide for Host Computer Development

# 24.1 Mesh OTA Introduction

### 24.1.1 Mesh OTA Features and Modes

The mesh OTA, termed mesh DFU (device firmware update) in the mesh spec, is a function that performs firmware upgrades for nodes. This function specifies how OTA is implemented and enables simultaneous firmware upgrades for nodes beyond the RF multi-hop distance.

### 24.1.2 Introduction to Mesh OTA Modes and Reference Rates

The following modes are supported:

(1) Support to upgrade multiple nodes at the same time by mesh ADV relay. Currently, the upgrade time for Demo SDK 160k firmware is around 60 minutes, and if Telink's Extended Broadcast Packet mode is enabled, the upgrade time is around 4 minutes.

(2) mesh OTA for LPN: the current Demo SDK 130k firmware upgrade time is around 70 minutes, if Telink's extended broadcast packet mode is enabled, the upgrade time is around 4 minutes.

(3) GATT OTA mode for single node (including LPN node): APP will disconnect the current connection, go to connect the upgraded node, and then execute OTA of SIG MESH, the sending and receiving process is the same as the process of upgrading multiple nodes, only the sending and receiving packet interactions are faster, the OTA time is basically the same as that of the OTA defined in the Telink gerneric BLE SDK, and the upgrading time is about 1 minute.

An overview of the functions can also be found in https://www.bluetooth.com/mesh-feature-enhancementssummary/

## 24.1.3 Mesh OTA Firmware Distribution Method

- (1) The host computer or app acts as both Initiator and distributor: The host computer is involved in the process of distributing firmware data to multiple nodes that need to be upgraded. The host has to stay connected to the nodes at all times and cannot be disconnected, a process that can take several tens of minutes.
- (2) "GATT master dongle + host computer" or app as Initiator, GATT directly connected node as distributor: in this mode, the host computer only needs to download the new firmware to the directly connected node through GATT in the front stage, and then the directly connected node acts as distributor to manage and execute the later work: distributing the new firmware to other nodes to be upgraded. At this time, the App can disconnect with the node if needed. Gateway does not support this mode for the time being, but directly adopts the mode of the host computer plus the gateway dongle as the Initiator and distributor, because the gateway doesn't need to keep the GATT connection with the nodes.

### 24.1.4 Three Role Profiles of Mesh OTA

- (1) Initiator: A node whose DISTRIBUTOR\_UPDATE\_CLIENT\_EN is set to 1 in the SDK functions as an OTA initiator, e.g., an PC tool sig\_mesh\_tool.exe, an app, etc. The Initiator fetches new firmware from the local or cloud via http and generates a receiver list, which specifies the nodes to which the OTA update is to be initiated. Then it sends the new firmware and receiver list to the Distributor, the corresponding process is in mesh\_ota\_initiator\_proc ().
- (2) Distributor: Nodes with DISTRIBUTOR\_UPDATE\_SERVER\_EN set to 1, such as gateway dongle and Mesh Light nodes with distributor turned on (not supported by default, need to enable DISTRIBU-TOR\_UPDATE\_SERVER\_EN). Function: Receive the firmware from Initiator, or download the new firmware through the http URL sent by Initiator, and store it temporarily. Then distribute the new firmware to the nodes belonging to the receiver list. Distributor corresponds to the sending process in mesh\_ota\_master\_proc () and receiving process in mesh\_ota\_master\_rx ().
- (3) Updating Node: The node that receives the OTA new firmware and updates the old firmware to the new firmware. That is, the node belongs to the receiver list.

### 24.1.5 Mesh OTA Silent Upgrade Mode

After distributing the firmware completely, the upgraded node will not apply the new firmware until receive firmware update apply command. So app or host can choose a suitable time to send firmware update apply command after transmitting firmware to the upgraded node to achieve silent upgrade mode.

### 24.1.6 Mods for Mesh OTA

- (1) BLOB Transfer(Binary Large Object Transfer) server: this model is used to receive large block data transfer, including but not limited to receiving firmware data and contains commands such as BLOB\_INFO\_GET, BLOB\_TRANSFER\_START, BLOB\_BLOCK\_START, BLOB\_CHUNK\_TRANSFER and so on. BLOB\_INFO\_GET, BLOB\_TRANSFER\_START, BLOB\_BLOCK\_START, BLOB\_CHUNK\_TRANSFER, and so on. There are two modes, Push mode and Pull mode (see definition of MESH\_OTA\_TRANSFER\_MODE\_SEL), Pull mode is only used for LPN nodes.
  - Push mode is where the host actively sends firmware data to the upgraded node, which has the advantage of being fast, but requires the node to be listening for mesh messages all the time.
  - Pull mode is when the upgraded node is in receive mode, requesting firmware data from the host. applicable to LPN devices, the OTA time is relatively long, and only one node is being upgraded at the same time.

The LPN supports both Push mode and Pull mode by default. the LPN can use Push mode only when it is in the GATT connection state.

- (2) Firmware update server: This mod is used for firmware process control and contains commands such as FW\_UPDATE\_METADATA\_CHECK, FW\_UPDATE\_START, FW\_UPDATE\_APPLY and so on.
- (3) Firmware Distribution server: This model is mainly used to receive new firmware and receiver list from Initiator. It downloads the new firmware to local storage by receiving commands such as FW\_DISTRIBUT\_UPLOAD\_START, FW\_DISTRIBUT\_RECEIVERS\_ADD, BLOB\_TRANSFER\_START, etc.; and distributes the new firmware by receiving commands such as FW\_DISTRIBUT\_START,



BLOB\_TRANSFER\_START, etc. BLOB\_TRANSFER\_START commands to download the new firmware to local storage; distribute the new firmware by receiving FW\_DISTRIBUT\_START, BLOB\_TRANSFER\_START commands.

- (4) BLOB Transfer client: Corresponds to the sender side of the server model.
- (5) Firmware update client : Corresponds to the sender side of the server model.
- (6) Firmware Distribution client : Corresponds to the sender side of the server model.

## 24.2 Test Mesh OTA with App

To test the mesh OTA using the App, please refer to the "4.4 Mesh OTA" section of the chapter Android and iOS APP User Guide.

# 24.3 Gateway Mesh OTA

The node to be upgraded is a non-LPN node.

### 24.3.1 Test and Command Sending and Receiving Process

#### 24.3.1.1 Code Configuration

Code configuration test conditions: 8258 dongle 1 (burn 8258\_mesh\_gw.bin), 8258 dongle 2 (burn 8258\_mesh.bin)

(1) Open MD\_MESH\_OTA\_EN.

```
      contig_82/8.h (vendor\mesh_provision)
      app_mesh.h (proj_lib\sig_mesh)
      mesh_contig.h (vendor\common) % mesh

      559:
      #define
      CERTIFY_BASE_ENABLE
      0

      560:
      #else
      0
```

```
#define CERTIFY_BASE_ENABLE
                                           0
         #endif
563: #endif
565: #ifndef MD_MESH_OTA_EN
566: #if (DEBUG_CFG_CMD_GROUP_AK_EN)
567: #define MD_MESH_OTA_EN
                                           1 // just for internal test.
568: #elif (_
              _PROJECT_MESH_PRO__)
                                      // app & gateway
         #if WIN32
570: #define MD_MESH_OTA_EN
                                           1
         #else // gateway
                                                // default disable before released by SIG.
572: #define MD_MESH_OTA_EN
                                           1
         #endif
574: #else
         #if ((MESH_USER_DEFINE_MODE == MESH_MI_ENABLE) || (LIGHT_TYPE_SEL == LIGHT_TYPE_PANEL) || SPIRIT_PI
576: #define MD_MESH_OTA_EN
                                           0
                                               // must 0
         #elif (MESH USER DEFINE MODE ==
                                           MESH IRONMAN MENLO ENABLE)
578: #define MD_MESH_OTA_EN
                                           0
                                               11
         #elif (AIS ENABLE)
580: #define MD_MESH_OTA_EN
                                               // default disable
                                           0
         #else
582: #define MD_MESH_OTA_EN
                                           1
                                               // dufault disable before released by SIG.
         #endif
                                                 mesh
584: #endif
585: #endif
```



(2) To shorten the mesh ota time, the macro EXTENDED\_ADV\_ENABLE can be set to 1 to support the extended broadcast packet mode, which should be noted is not a spec-defined mode.



Figure 24.2: Open EXTENDED\_ADV\_ENABLE

(3) The host computer can set Extend Adv to enable the extended broadcast packet mode.

Telink sig_mesh Not Found V3.3.3.5	
CHD tl_node_gateway.ini 💌 INI BULKOUT ASCII 🔽	Log 🗆 AutoSaveLog 2 retry Clear Save Save 🗗 Hex 🗆 Adv Stop Scan rp_scan ex_scan OTA Rx te
mesh_bulk_cmd_debug	fastbind Extend Adv: OTA Only
LPN_get_ievei	None
lightness get Danel	OTA Only
rightness_get_ranei	All
LDN fy distrib ota start 04	
fw update info get	
fw update info get all	None: no extend adv
fw_distribution_get	
these distribution start not for LPN. For LPN, plea	OTA Only: only mash OTA use extend adv
fw_distribution_start_all	ora only only mesh ora use extend duv
fw_distribution_start_0002	Ally all mach massage use extend adv
fw_distribution_start_02_03	All, all mesh message use exteriu adv
fw_distribution_start_0001	
fu distribution sugmend	
fy distribution cancel	
fw update metadata check	
fw update get	
fw_update_start	
fw_update_cancel	
fw_update_apply	
blob_transfer_get	
blob_transfer_start	
blob_transfer_cancel	
blob_block_start	
blob block get	
blob info get	
scheduler_get	
sched_action_get	
sched_action_set_off	
sched_action_set_on	
scned_action_set_scenei	
time set	
time get	
time zone set	
time_zone_get	
time_delta_set	
time_delta_get v	ALL  chn_set isconned input_db Path: E:\git_lab\telink_ OpenFile Mesh
e8 ff 00 00 00 00 02 00 01 00 b6 0b 00 c0	directed COM18 V UART USB output db CuReset CuMeshots Currented Brow Class
	Garbert Garbert Garbert And Crose

**Figure 24.3**: The host computer opens the extend\_adv

## 24.3.1.2 Networking Nodes

Connect the pc through gw dongle, open tools tool sig\_mesh\_tool.exe, select the ini file corresponding to gw as following figure 1-1, and then network the 2 8258 dongles sequentially. After successful networking, click "MESH" button to open the mesh control interface, and automatically send LIGHTNESS\_GET all command to get all the current nodes (you can also send ONOFF\_GET).

HCI\_CMD\_GATEWAY\_CMD + netkey index(2 bytes) + appkey index(2 bytes) + retry cnt + rsp\_max + gateway addr + op + par

i.e.: e8 ff + netkey index + appkey index + retry cnt + rsp\_max + gateway addr + 82 02 00 00

Note: Unless otherwise specified, the ini format is hexadecimal in this document.

Make sure that all nodes are displayed in the UI, because if you want to do mesh OTA via fw\_distribution\_start\_all later, the unicast address of the corresponding node is obtained from inside this UI, which is the source of the address list in the parameter area of FW\_DISTRIBUT\_START.

Telink sig_mesh Found V3.3.3.5	- L X
CHD <b>bl_node_gateway.ini</b> INI BULKOUT ASCII	Log 🗆 AutoSaveLog 💈 retry Clear Save Save 📈 Hex T Adv Stop Scan rp_scan ex_scan   OTA   Rx test
sig mesh master.ini md_debug ^	fastbind Extend Adv: None 💌
LPN get onoff	
lightness get Panel	: as 22 13 bl 9e be 93 3e 83 e5 eb er 2b re dl 02
Note:retry count field of LPN distrib start is change	<1007>13117:08:331 [INFO]: (GAIEWAI) the gateway mac adr 1s
LPN fw distrib ota start 04	: 3d cl 95 3U cl a4
	<1008>13:17:08:331 [INSO]: (common) gateway ]son init start
fw update info get	<1009>13:17:08:932 [INFO]: (COMMON) HODE 14X=0, VC uuld=: a8 22 13 b1 9e b6 93 3e 83 e5 6b ef 2b fe d1 0
fw update info get all	<1010>13:17:08:939 (INFO): (WAILWAI) Update gw's netkey first
fw distribution get	
Note: these distribution start not for LPN. For LPN, p	<1011>13:17:08:333 [INEQ]: (WAIEWAI)HCI_WAIEWAI WIT SANWINEI ALI : 09 IT 20 01 /2 /2 00 93 93 92 40 40
fw distribution start all	1012/13/17/00/335 [INFO]: (GAIEWAI ACT GAIEWAI CHU SAI NODE MARA : 05 FF 10 00
fw distribution start 0002	sitissis. 1.100.545 [Into]. (Withwaithing gateway should Adv option 15:0
fw distribution start 02 03	AND AND AN AND AND AND AND AND AND AND A
fw distribution start 0001	<1014/13:17/32:161 (INFO): (GAIERA) RCI GAIERAI COL SAI NODE FARA : 65 EF 10
distribution start end	<1015>13:17/32:168 [INFO]: (GAIEWA) the gateway utila is
fw distribution suspend	: 80 44 13 D1 90 D6 93 30 03 05 6D 01 4D 10 01 04 4
fw distribution cancel	1 of 1 of 1 of 1 of 1 of
fw update metadata check	: 30 C1 75 30 C1 89
fw update get	<pre><li></li></pre>
fw update start	<pre><li><li><li><li><li><li><li><li><li><li< td=""></li<></li></li></li></li></li></li></li></li></li></pre>
fw update cancel	51 2 2 2 6 62 62 62 6 6 6 6 6 6 6 6 7 7 7 1 d) d
fw update apply	: DI /2 /C 05 33 33 5C 46 46 DV DA DA C/ GI GI GA 21000-10119-17-09-171 INTENI//CATEVANULAT CATEVANU CANE OF MET VEV , a6 46 30 51 73 74 65 63 69 69 60 60 50 50
blob transfer get	<1010-10.17.30.171 [INCO]. (WILEWAI / CUI_GALEWAI / CUI /
blob transfer start	<1002101017.02.104 [INED] (GATUMAT ACT GATUMAT THE ONLY ALL OF A CONTRACT OF A CONTRAC
blob_transfer_cancel	(1022/13.17.32.104 [INEO]. (WAISHAI) the gateway Extend Adv option 15.0
blob block start	-1000-10-17-47-516 (TNEOL) (CATENAN) WOT CATENAN CHE GET NORE DADA : -6 66 10
blob_chunk_transfer	<pre><li></li></pre>
blob block get	-2027 - 2013 - 11 - 50 (ansolutional vie gatematy und as
blob_info_get	. Bo as 19.17.47 F200 [INTO] (CITENNI) the determined add is
	- 3d of 165 38 of ad
scheduler_get	<pre><li>&lt;1026&gt;13:17:47:528 [INFO]: (common) gateway ison init start</li></pre>
sched_action_get	-1020-12-17-47-529 [INFO]: (common) gode idyed) you uside: a8 22 13 h1 9e h6 93 3e 83 e5 fb ef 2h fe d1 0
sched_action_set_off	<pre><li>&lt;1028&gt;13:17:47:529 [INFO]: (GATEWAY) undate my's netter first</li></pre>
sched_action_set_on	51 72 7c 85 93 93 9c af af h0 ha ha c7 d1 d1 da
sched_action_set_scenel	(1029>13:17:47:529 [INFO]: (GATEWAY)HCI GATEWAY (MD SEND NET KEY : e9 ff 20 51 72 7c 85 93 93 9c a6 a6
	<1030>13:17:47:529 (INFO): (GATEWAY) HCI GATEWAY CMD SET NODE PARA : e9 ff 16 00
time_set	<1031>13:17:47:543 [INFO]: (GATEWAY) the gateway Extend Adv option is:0
time_get	
time_zone_set	~
time_zone_get	< > >
time_delta_set	
time_delta_get v	ALL  chn_set connect input_db Path: C:\Users\ZB\Desktoj OpenFile Mesh
e8 ff 00 00 00 00 02 00 02 00 b7 03 11 22 33 44 55 66 77 8	directed VART USB output_db GwReset GwMeshOta GwOtaSelf Prov Close

Figure 24.4: Network node

### 24.3.1.3 Select New Firmware

Click on the open file button in the figure, select the target firmware, and click confirm.

Telink master Not Found V3.3.3.5		-	- ×
CMD sig_mesh_master.ini V INI BULKOUT ASCII	Vog 🗆 AutoSaveLog 2 retry Clear Save Save 🔽 Hex 🗆 Adv Stop Scan rp_scan ex_s	can OTA	Rx tes
mesh bulk cmd debug			
PN get level	V Fastbind Extend Adv: All		
PN get onoff	10713-11-12-00-400 (THEOL (CREENV)	17	04.74
ightness get Panel	(0712)1116(19:9400 [INFO].(0812841) CHI Sendback Stc.0001 dst.0000, op 0074(2): as 01 0	2 05 51 0	00 -2
Note:retry count field of LPN distrib start is change	<pre>collactice.la.ast [info]: (driskr) cmd sendback sic.dool dsc.eddd, op do/d(); as of s collactice.la.ast [info]: (driskr) cmd sendback sic.dool dsc.eddd, op do/d(); as of s</pre>	0 21	- 66 67
PN fw distrib ota start 04	collective: (area): (areas and areas are	6 42 51 et	C #2 42
PN_initiator_start_v_apply4	(0716-11-16-21-16) [THEO]-(but under a manager action of a second of the second s		
PN initiator_start_v_only4	- 00 00 00 01		
	(0175)11(5)24:705 [THEOL: (mt us log)(073 block sum: 1 sum: 0 shunk sum:60] sum:425	Drograss	718 107
w update info get	(0719-11:16:24:000 [INFO]:(0710-110) The sound conducts sum: 1, cut. 0, chunk sum: cut. 0, cut. 4, cut	9 ec 02 2	02 60
<pre>w_update_info_get_all</pre>	<pre>construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c0000, op 007d(7); ad 01 a construction (international sendback stc:0001 dst:c000</pre>	a 06 0b 40	68 06
#_distribution_get	(0720)11:16:32:322 [INFO] (GATEMAY) and sandback src:0001 dst: c000, op 007d(2): af 01 3	a 60 90 00	30 99 9
Note: these distribution start not for LPN. For LPN, p	(0721)11:16:36:107 [INFO] (GITENEY) and sendback src:0001 dst: c000, op 007d(2) - b0 01 0	2 33 03 30	5 01 a2
distribution_start_all	(0722)11:16:39:892 [THEO] (GATENIA) and sendback src:0001 dst: d000, op 0074(2) bl 01 0	5 60 00 00	00 43
distribution_start_0002	(0722)11:16:42-710 [INFO] (GATENA) and sendback src:0001 det: g000 ap 007d(2) - b2 01 o	2 40 20 -	2 22 41
_distribution_start_02_03	<0724>11:16:47:528 [INFO1: (my ve log)QTA block sum: 1 cur: 0, chunk sum:60] cur:435	Progress	725 M
_distribution_start_0001	<0725>11:16:47:544 [INFO] (GTENAY) and sendback src:0001 dst:c000 op 007d(2) - b3 01 1	3 40 05 70	38
distribution start end	<0726>11:16:48:325 [INFO]: (iv undate) gate value dongle report TUT-		
_distribution_suspend			
distribution_cancel	<pre>&lt;0227&gt;11:16:51:288 [INFO1:(CATEWAY) and sandback src:0001 dst:c000 op 007d(2): b4 01 6</pre>	b 06 44 97	7 bb Sk
update_metadata_check	(0728-11-16-55-102 [INFO] (GATENAY) and sandhack src:0001 dst: c000, op 007d(2) - 55 01 4	0 06 49 00	E 15 #2
update_get	(0729-11-16-59-929 [INFO]-(CATENAY) and sendback src-0001 dst-c000, op 007d(2)- b6 01 9	1 =2 ## 97	7 09 90
_update_start	<0730>11:17:02:718 [INFO]: (GATEWAY) and sendback src:0001 dst: c000 op 007d(2): b7 01 0	5 40 01 b	11 00
_update_cancel	<0731>11:17:06:551 [INFO]: (GATEWAY) and sendback src:0001 dst: c000 on 007d(2): b8 01 0	2 41 43 4	f0 87
update_apply	<0732>11:17:10:430 [INFO]: (my vc log)OTA block sum: 1 cur: 0, chunk sum:60] cur:441	Progress	735 M
ob_transfer_get	<0733>11:17:10:445 [INFO]: (GTENAY) and sandback src:0001 dst: c000 op 007d(2): b9 01 f	0 65 57 00	6 46 06
ob_transfer_start	<0734>11:17:14-188 [INFO]-(GATENAY) and sandback src:0001 dst:c000 op 007d(2): ba 01 a	3 4b 1b fr	1 1 0 3
ob_transfer_cancel	<0735>11:17:15:665 [INFO]: (iv update)gateway dongle report IVI:		
ob_block_start	- 00 00 00 01		
ob_chunk_transfer	<0736>11:17:17:972 [INFO]: (GATEWAY) and sendback src:0001 dst:c000.op 007d(2): bb 01 0	0 86 00 8	e3 f0
ob_block_get	<0737>11:17:21:811 [INFO]: (GATEWAY) cmd sendback src:0001 dst:c000.op 007d(?); bc 01 5	3 15 28 er	0 C 60
ob_info_get	<0738>11:17:25:598 [INFO]: (GATEWAY) cmd sendback src:0001 dst:c000.op 007d(?): bd 01 d	8 03 01 00	80 03
_initiator_start_verify_only_all	<0739>11:17:29:441 [INFO]: (GATEWAY) and sendback src:0001 dst:c000.op 007d(?); be 01 9	9 03 0b er	13 03
_initiator_start_verify_apply_all	<0740>11:17:33:254 [INFO]: (gw vc log)OTA, block sum: 1.cur: 0. chunk sum:601.cur:447.	Progress	745 N
	<0741>11:17:33:270 [INFO]: (GATEWAY) and sendback src:0001 dst:c000.op 007d(?): bf 01 0	1 7e 30 er	29 ec
_initiator_start_verify_apply_02_04	<0742>11:17:36:966 [INFO]: (GATEWAY) cmd sendback src:0001 dst:c000.op 007d(7): c0 01 0	0 a0 00 at	54 c0
_initiator_start_VC_test	<0743>11:17:40:762 [INFO]: (GATEWAY) and sendback src:0001 dst:c000.op 007d(2): c1 01 e	3 9f ea 8'	7 3e ed
_distribution_cap_get	<0744>11:17:42:851 [INFO]: (iv update)gateway dongle report IVI:		
_distribution_recv_get	: 00 00 00 01		
_distribution_recv_add	<0745>11:17:44:550 [INFO]: (GATEWAY) cmd sendback src:0001 dst:c000.op 007d(?): c2 01 e	0 ъб 02 31	21 b3
_distribution_recv_del_all	c		
<pre>/_distribution_upload_get</pre>			7
<pre>#_distribution_upload_start </pre>	ALL      chn_set connect input_db Path: C:\Users\ZB\Desktop	OpenFile	Mesh
		-	-
8 ff 00 00 00 00 02 00 02 00 b7 03 11 22 33 44 55 66 77 8	directed V UART USB output db GwReset GwMeshOta GwOtaSe	1f Prov	Clos

Figure 24.5: Select new firmware

### 24.3.1.4 Download New Firmware to Local Gateway Dongle

Click the "GwMeshOta" button, the host computer loads the selected new firmware from the target path to the gateway dongle for storing locally, so as to prepare for the subsequent mesh OTA. After the button is clicked, the host computer calls OnBnClickedGatewayOta to send firmware to the gateway, the corresponding ini commands are as follows:

(1) Set the ota type to GATEWAY\_OTA\_MESH in the format: HCI\_CMD\_MESH\_OTA + MESH\_OTA\_SET\_TYPE + GATEWAY\_OTA\_MESH. i.e.: eb ff + 01 + 00.

Note: ota type GATEWAY\_OTA\_SLEF (01) is the upgrade gateway itself.

- (2) Send the ota start command to wait for the ota area erase to complete. the default wait is 5 seconds. in the format: HCI\_CMD\_GATEWAY\_OTA + len + CMD\_OTA\_START. i.e.: ea ff 02 01 ff.
- (3) Send the firmware package in the format: HCI\_CMD\_GATEWAY\_OTA + len + ota\_index(2 bytes) + ota\_payload(16 bytes) + crc16. i.e.: ea ff + 14 + ota\_index + ota\_packet(16 bytes) + crc16. Where pkt index is the index value of the ota packet, the size of each ota payload is 16 bytes, and the crc16 value is the crc16 checksum of ota\_index and ota\_payload.
- (4) After the firmware is transferred, send the ota end command in the format: HCI\_CMD\_GATEWAY\_OTA + len + CMD\_OTA\_END + index\_max + ~(index\_max). i.e.: ea ff + 6 + 02 ff + index\_max + ~(index\_max). Where index\_max = (firmware\_total\_len + 15)/16 1, is the maximum index value of the ota packet, which is used to receive the ota end command whether to collect all or not.

#### Note:

New firmware is temporarily stored in the pending OTA area of the gateway dongle (0x0000 or 0x40000), once the gateway dongle is rebooted, it will be cleared, and you need to perform the loading action again. Otherwise, the mesh ota cannot be performed subsequently.



Figure 24.6: Download new firmware to local cache

### 24.3.1.5 Get the Version Information of the Nodes Currently on the Network

Users can query the version information of the nodes currently on the network via fw\_updata\_info\_get, as shown in the following figure 02 The version number of the address device is displayed 41 00 (i.e., the ASCII code of 4.1 version "4", "1").

HCI\_CMD\_GATEWAY\_CMD + netkey index(2 bytes) + appkey index(2 bytes) + retry cnt + rsp\_max + dst addr + op + par

#### i.e.: e8 ff + netkey index + appkey index + retry cnt + rsp\_max + dst addr + 83 08 00 01

🛞 Telink sig\_mesh -- Found V4.1.0.0

CMD tl_node_gateway.ini 💌 INI BULKOUT ASCII 🔽	Log T AutoSaveLog 2 retry Clear Save Save V Hex T Adv Stop Scan rp_scan ex_scan OTA Rx test
LDN get level	fastbind Extend Adv: None
LPN get_nooff	
lightness get Panel	fw_update_info_get
Note:retry count field of LPN distrib start is change	<0000>14:52:29:859 [INFO]: (common) ExecCmd: e8 ff 00 00 00 00 02 00 02 00 83 08 00 01
LPN fw distrib ota start 04	<pre>&lt;0001&gt;14:52:29:893 [INFO]: (GATEWAY) cmd sendback src:0x0001 det 0x00002, op 0883 (FW UPDATE INFO GET): 00</pre>
	<pre>&lt;0002&gt;14:52:29:908 [INFO]: (cmd rsp)Status Rsp :: 02:00 01:00:83:09:01:00:04:01:00 04:00:00</pre>
CDTP OTS GATT ADV ON	<0003>14:52:29:908 [INFO]: (GATEWAY)HCI GATEWAY RSP OP CODE
CDTP OTS GATT ADV OFF	: 91 81 02 00 01 00 83 09 01 00 04 01 00 00
fw update info get	
fw update info get all	
fw distribution get	
Note: these distribution start not for LPN. For LPN, p	
fw distribution start all	
fw distribution start 0002	
fw distribution start 02 04	
fw distribution start 0001	
distribution start end	
fw distribution suspend	
fw_distribution_cancel	
fw_update_metadata_check	
fw_update_get	
fw_update_start	
fw_update_cancel	
fw_update_apply	
blob_transfer_get	
blob_transfer_start	
blob_transfer_cancel	
blob_block_start	
blob_chunk_transfer	
blob_block_get	
blob_info_get	
scheduler_get	
sched_action_get	
sched_action_set_off	
sched_action_set_on	
sched_action_set_scenel	
time_set	< > >
time_get	
time_zone_set V	ALL Chn_set connect input_db Path: OpenFile Mesh
e8 ff 00 00 00 02 00 02 00 83 08 00 01	
	directed UART USB output_db GwReset GwMeshOta GwOtaSelf Prov Close
,	

Figure 24.7: Get node information

## 24.3.1.6 Send fw\_distribution\_start\_all Command

Double-click the fw\_distribution\_start\_all command in the ini list, the host computer(SIG Mesh Tool on PC) will automatically read the unicast address of all current nodes from inside the UI list at vc\_distribute\_all\_proc(), command format:

HCI\_CMD\_GATEWAY\_CMD + netkey index(2 bytes) + appkey index(2 bytes) + retry cnt + rsp\_max + gateway addr + op + par

i.e.: e8 ff + 00 00 + 00 00 + 02 + 00 + gateway\_addr + 83 19 + group\_addr + device\_addr \_list.

where gateway\_addr is the gateway address, and the gateway will do mesh ota to the device specified by device\_addr\_list after adding it to group group\_addr.

### 24.3.1.7 OTA Progress Reporting

The gateway receives distribution\_start and starts the mesh ota process, mesh\_cmd\_sig\_fw\_distribute\_start()->mesh\_ota\_master\_proc(), which sends the firmware that is temporarily stored in the gateway to the updating node.

Gateway ota progress is currently reported in string format through the APP\_RefreshProgressBar interface shown as below:

CMD tl_node_gateway.ini V INI BULKOUT ASCII	Log T AutoSaveLog 2 retry Clear Save Save 🔽 Hex T Adv Stop Scan rp_scan ex_scan OTA Rx +
mesh_bulk_cmd_debug	fastbind Extend Adv: None
LPN_get_level	
LPN_get_onoff	[INFO]: (GATEWAY) cmd sendback src:0001 dst:c000, op 007d(?): 48 02 82 68 00 00 82 6a 00 00 05 13 00 0
lightness_get_Panel	[INFO]: (GATEWAY) cmd sendback src:0001 dst:c000, op 007d(?): 49 02 bc 57 84 00 00 00 00 00 08 00 01 0
Note:retry count field of LFN distrib start is change	[INFO]: (iv_update) gateway dongle report IVI:
LPN_fw_distrib_ota_start_04	
de undere inde ann	[INFO]: (GATEWAY) cmd sendback src:0001 dst:c000, op 007d(?): 4a 02 e4 5d 84 00 00 00 00 00 b8 00 01 0
rw_update_info_get	[INFO]:(GATEWAY) cmd sendback src:0001 dst:c000,op 007d(?): 4b 02 00 d2 00 d6 00 da 00 de 00 e2 00 e
Iw_dpdace_inito_get_all	[INFO]: (gw_vc_log)OTA, block sum: 1, cur: 0, chunk sum:601, cur:588, Progress:575 NULL
IN distribution_get	[INFO]: (GATEWAY) cmd sendback src:0001 dst:c000, op 007d(?): 4c 02 04 00 00 00 05 00 00 06 00 00 0 0
fu distribution start all	[INFO]: (GATEWAY) cmd sendback src:0001 dst:c000,op 007d(?): 4d 02 2c 5d 84 00 c0 2f 84 00 28 02 00 0
fu distribution start 0003	[INFO]: (GATEWAY) cmd sendback src:0001 dst:c000, op 007d(?): 4e 02 00 10 03 00 01 00 00 00 02 00 00 0
fw distribution start 02.02	[INFO]:(GATEWAY)cmd sendback src:0001 dst:c000,op 007d(?): 4f 02 43 44 45 46 00 00 00 01 100 00 0
fy distribution start 0001	[INFO]: (GATEWAY) cmd sendback src:0001 dst:c000, op 007d(?): 50 02 ff ff ff ff 00 00 00 00 00 00 00 00 00
distribution start end	[INFO]: (iv_update)gateway dongle report IVI:
fy distribution suspend	
fy distribution cancel	[INFO]: (GATEWAY) cmd sendback src:0001 dst:c000, op 007d(?): 51 02 67 00 00 00 67 65 74 20 75 74 20 7
fy update metadata check	[INFO]: (gw vc log) OTA, block sum: 1, cur: 0, chunk sum: 601, cur: 594, Progress: 904 NULL
fy update get	[INFO]: (GAIEWAI) cmd sendback src:0001 dst:c000, op 007d(?): 52 02 65 65 79 00 67 65 74 20 74 68 65 2
fy update start	[INFO]: (CATEWAX) cmd sendback src:0001 dst:c000, op 007d(7): 53 02 64 20 64 65 76 69 63 65 20 65 65 7
fw update cancel	[INFO]: (GAIEWAR) cmd sendback src:0001 dst: c000, op 007d(2): 54 02 az b7 01 00 az b7 01 00 32 b5 01 0
fy update apply	[INFO]: (GRIENAT) cha sendback src:0001 dst:000, 00 007d(7): 55 02 a2 57 01 00 a2 57 01 00 a2 57 01 0
blob transfer get	[INFO] (GRIERAT) chi senabaci sic.0001 dsc.000, dp 007d(5), 56 02 00 00 e0 03 03 04 20 06 13 00 00 0
blob transfer start	[THEO]: (AN LAWAR) CHA SHADARCK STC: 0001 ASC: 2000, 00 00 4(7): 57 02 11 02 01 00 33 35 65 00 07 00 00 0
blob_transfer_cancel	[INFO]: (gr=vc_toy) of A, block sum: 1, cut: 0, chair sum sol, cut: coo, rioglessiss with b
blob block start	(TWP): (on intervision set about a site coop of a site of a coop o
blob_chunk_transfer	(into). (iv_update/yateway donyie report ivi.
blob block get	(TUPO) (CATENAY) and conducate era:0001 det:0002 on 07b7/2) NULL
blob_info_get	[INFO]. (on take) Status Ben
	(INFO) - (datemay) Hot (datemay BSD OD CODE
scheduler_get	
sched_action_get	[INFO] (GATEWAY) and sendback src 0001 dst 0004 op 07b7(2) NULL
sched_action_set_off	[INFO]: (cmd rsp) Status Rsp : 04 00 01 00 7e 40 00 00 d0 00
sched_action_set_on	(INFO) - (GATEWAY) HCI GATEWAY BSP OF CODE
sched_action_set_scenel	7 <b>e</b> 40 00 00 d0 00
	[INFQ]: (GATEWAY) cmd sendback src:0001 dst:0002.op 05b6(2) NULL
time_set	[INFO]: (cmd rsp)Status Rsp : 02 00 01 00 b6 09 80 ff 01 00 00 11 22 33 44 55 66 77 88
time_get	(INFO): (GATEWAY)HCI GATEWAY RSP OF CODE
time_zone_set	be 09 80 ff 01 00 00 11 22 33 44 55 66 77 88 00
time_zone_get	6
time_delta_set	
time delte ant	

Figure 24.8: OTA progress reporting

### 24.3.1.8 Mesh OTA Completion Display Page

After ota has finished, report ota results via gateway\_upload\_mesh\_ota\_sts in the format:

TSCRIPT\_GATEWAY\_DIR\_RSP + HCI\_GATEWAY\_CMD\_SEND\_MESH\_OTA\_STS + fail\_num + fail\_list.

i.e.: 91 + 98 + fail\_num + fail\_list.

Telink

T

Where a fail\_num of 0 indicates the number of failed upgrades, and 0 indicates all successes, the print page will show mesh OTA success as shown below.



B Telink sig\_mesh -- Found V4.1.0.0

CMD tl_node_gateway.ini 💌 INI BULKOUT ASCII	Log 🗆 AutoSaveLog 2 retry Clear Save Save 🔽 Hex 🗆 Adv Stop Scan rp_scan ex_scan OTA Rx test
<pre>mesh_bulk_cmd_debug</pre>	fastbind Extend Adv: All
LPN_get_level	
LPN_get_onoff	<0208>15:27:07:475 [INFO]: (cmd rsp)Status Rsp : 04 00 01 00 67 c0 00 00 d0 00 07 A
lightness_get_Panel	<0209>15:27:07:475 [INFO]: (GATEWAY) HCI GATEWAY RSP OP CODE
Note:retry count field of LPN distrib start is change	: 91 81 04 00 01 00 67 c0 00 00 d0 00 07
LPN_fw_distrib_ota_start_04	<0210>15:27:07:614 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0xc000.op 0066(BLOB CHUNK TRANSFER): 07
	<0211>15:27:07:833 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002.op 0583(BLOB BLOCK GET) NULL
CDTP_OTS_GATT_ADV_ON	<0212>15:27:07:864 [INFO]: (cmd rsp) Status Rsp : 02 00 01 00 67 40 00 00 d0 00
CDTP_OTS_GATT_ADV_OFF	<0213>15:27:07:864 [INFO]: (GATEWAY) HCI GATEWAY RSP OP CODE
	: 91 81 02 00 01 00 67 40 00 00 40 00
fw_update_info_get	<0214>15:27:08:052 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0004.op 0583(BLOB BLOCK GET)NULL
fw_update_info_get_all	<0215>15:27:08:209 [INFO]: (cmd rsp) Status Rsp : 04 00 01 00 67 40 00 00 d0 00
fw_distribution_get	<0216>15:27:08:209 [INFO]: (GATEWAY) HCI GATEWAY RSP OP CODE
Note:these distribution start not for LPN. For LPN, p	: 91 81 04 00 01 00 67 40 00 00 d0 00
fw_distribution_start_all	<0217>15:27:08:271 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002.op 0c83(FW UPDATE GET) NULL
fw_distribution_start_0002	<0218>15:27:08:582 [INFO]: (cmd rsp) Status Rsp : 02 00 01 00 83 10 80 ff 01 00 00 11 22 33
fw_distribution_start_02_04	<0219>15:27:08:582 [INFO]: (GATEWAY) HCI GATEWAY RSP OP CODE
fw_distribution_start_0001	: 91 81 02 00 01 00 83 10 80 ff 01 00 00 11 22 33 44 55 66 77 88 00
distribution start end	<0220>15:27:08:754 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0004,op 0c83(FW UPDATE GET)NULL
fw_distribution_suspend	<0221>15:27:09:035 [INFO]: (cmnd_rsp)Status Rsp: 04 00 01 00 83 10 80 ff 01 00 00 11 22 33
fw_distribution_cancel	<0222>15:27:09:035 [INFO]: (GATEWAY) HCI GATEWAY RSP OP CODE
fw_update_metadata_check	: 91 81 04 00 01 00 83 10 80 ff 01 00 00 11 22 33 44 55 66 77 88 00
fw_update_get	<0223>15:27:09:219 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0002,op 0f83(FW_UPDATE_APPLY)NULL
fw_update_start	<0224>15:27:09:530 [INFO]: (cmd_rsp)Status Rsp: 02 00 01 00 83 10 c0 ff 01 00 00 11 22 33
fw_update_cancel	<0225>15:27:09:530 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE
IW_update_apply	: 91 81 02 00 01 00 83 10 c0 ff 01 00 00 11 22 33 44 55 66 77 88 00
blob_transfer_get	<0226>15:27:09:717 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0004,op 0f83(FW_UPDATE_APPLY)NULL
blob_transfer_start	<pre>&lt;0227&gt;15:27:09:999 [INFO]:(cmd_rsp)Status Rsp: 04 00 01 00 83 10 c0 ff 01 00 00 11 22 33</pre>
blob_transfer_cancel	<0228>15:27:09:999 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE
blob_block_start	: 91 81 04 00 01 00 83 10 c0 ff 01 00 00 11 22 33 44 55 66 77 88 00
blob_chunk_transfer	<0229>15:27:10:186 [INFO]:(GATEW; Y)mesh OTA success
blob_block_get	
blob_inio_get	<0230>15:27:10:201 [INFO]:(GATEWAY) cmd sendback src:0x0001 dst:0x0001,op 1b83(FW_DISTRIBUT_CANCEL) NULL
	<pre>&lt;0231&gt;15:27:10:217 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0001,op 1d83(FW_DISTRIBUT_STATUS): 00</pre>
scheduler_get	<0232>15:27:10:232 [INFO]:(cmd_rsp)Status Rsp: 01 00 01 00 83 ld 00 00
sched_action_get	<0233>15:27:10:232 [INFO]:(cmd_name)mesh OTA completed or get info ok!
sched_action_set_orr	<0234>15:27:10:232 [INFO]: (GATEWAY) HCI_GATEWAY_RSP_OP_CODE
sched_action_set_on	: 91 81 01 00 01 00 83 14 00 00
sched_action_set_scener	<pre>&lt;0235&gt;15:27:10:279 [INFO]:(gw_vc_log)OTA, block sum: 0,cur: 0, chunk sum: 0,cur: 0, Progress:100% NULL</pre>
time set	V
time get	< >
time zone set	
, · · · · · · · · · · · · ·	And the connect input_db Path: A:/test_Din/0200_m OpenFile Mesh
e8 ff 00 00 00 00 02 00 01 00 83 0e	directed V UART USB output_db GwReset GwMeshOta GwOtaSelf Prov Close
1	

Figure 24.9: Mesh OTA completed

### 24.3.1.9 Device Flashes 6 Seconds Slowly

All devices will flash slowly for 6 seconds, and then reboot automatically to take effect new firmware. after reboot, you can query the version through step 5 above to confirm whether the version is upgraded successfully again, as shown below.

Telink sig_mesh Not Found V3.3.3.5	X
CHD tl_node_gateway.ini 💌 INI BULKOUT ASCII	Log T AutoSaveLog 2 retry Clear Save Save 🔽 Hex T Adv Stop Scan rp_scan ex_scan OTA Rx test
mesh_bulk_cmd_debug	fastbind Extend Adv: None
LVN_get_level	
LVN_get_onorr	[1:30:51:721 [INFO]: (cmd_rsp)Status Rsp: 02 00 ff ff 82 4e ff ff A
lightness_get_Panel	L1:30:51:732 [INFO]: (GATEWAY)HCI_GATEWAY_RSP_OP_CODE
Note:retry count field of LWN distrib start is change	L 02 00 ff ff 82 4e ff ff
LVN_TW_distrib_ota_start_04	[1:30:54:007 [INFO]:(common)VC send to gateway is: e8 ff 00 00 00 00 02 02 ff ff 82 02 00 00
· · · · · · · · · · · · · · · · · · ·	<pre>L1:30:54:026 [INFO]:(GATEWAY)cmd sendback src:0001 dst:ffff.op 0282(G_ONOFF_SET): 00 01</pre>
IW_update_info_get	[1:30:54:322 [INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 82 04 01 00 0a
fw_update_info_get_all	L1:30:54:330 [INFO]: (GATEWAY) HCI_GATEWAY_RSP_OP_CODE
fw_distribution_get	L 02 00 01 00 82 04 01 00 0a
Note: these distribution start not for LPN. For LPN, p	[1:30:54:337 [INFO]: (cmd_rsp)Status Rsp: 04 00 01 00 82 04 01 00 0a
fw_distribution_start_all	11:30:54:346 [INFO]: (GATEWAY) HCI GATEWAY RSP OP CODE
fw_distribution_start_0002	L 04 00 01 00 82 04 01 00 0a
fw_distribution_start_02_03	L1:30:55:872 [INFO]: (common)VC send to gateway is: e0 ff 00 00 00 00 02 02 ff ff 02 02 01 00
fw_distribution_start_0001	11:30:55:895 [INFO]: (GATEWAY) cmd sendback src:0001 dst:ffff.op 0282(G ONOFF SET): 01 02
distribution start end	11:30:56:145 [INFO]: (cmd rsp)Status Rsp : 02 00 01 00 82 04 00 01 0a
fw_distribution_suspend	(1:30:56:153 (INFO): (GATEWAY) HCI GATEWAY RSP OP CODE
fw_distribution_cancel	L 02 00 01 00 82 04 00 01 0a
fw_update_metadata_check	L1:30:56:255 [INFO]: (cmd rsp)Status Rsp : 04 00 01 00 02 04 00 01 0a
fw_update_get	11:30:56:263 (INFO): (GATEWAY) HCI GATEWAY RSP OP CODE
fw_update_start	L 04 00 01 00 82 04 00 01 0a
fw_update_cancel	11:31:02:514 [INFO]: (common)Exected: e8 ff 00 00 00 02 00 02 00 b6 01 00 01
fw_update_apply	11:31:02:552 [INFO]: (GATEWAY) and sendback src:0001 dst:0002 op 01b6(2): 00 01
blob_transfer_get	11:31:02:677 [INFO]: (cmd rsp)Status Rsp : 02 00 01 00 b6 02 01 00 04 01 00 33 35 00
blob_transfer_start	11-31-02-677 (INFO)- (GATEWAY) HCT GATEWAY DED OD CODE
blob_transfer_cancel	
blob_block_start	11:31:45:131 (INFO): (common) ExerCit: #8 ff 00 00 00 02 00 04 00 b6 01 00 01
blob chunk transfer	11:21:46:172 (TNFO): (CATENAY) and condback erg:0001 det:0004 op 01b6(2): 00.01
blob_block_get	11:91:46:200 [TNEO]:(omd ven) Cratter Den
blob_info_get	11-31-46-20 (INFO) (GATWAY) HCT GATWAY DED OD CODF
scheduler get	19170.00.617 / THEOLI / CATOLY LAT CATOLY CAN OF HONE BARE
sched_action_get	19:17:00:091/[INFO]:(CATEMAI/AUL_OALEMAI_GED_GEL_MODE_FARE : EFIL IV
sched_action_set_off	1 13 1 6 b 6 2 3 3 3 4 5 b 4 7 b 6 d 1 0 1
sched action set on	: 13 DI 76 DE 73 30 C3 65 ED ET 2D TO C1 04
sched_action_set_scenel	1 05 00 01 44
	1 25 35 CL RY
time set	13.17.00.591 [INFO].(Common/gateway json into state
time get	la 17.00.000 (MEV), (CHERNIN, NOVE ANA ", VC WARA", 46 22 13 DI 96 D6 93 36 63 65 6D 6T 2D TE GI 02
time zone set	1 25 02 02 02 02 05 05 05 05 05 05 05 05 05 05 05 05 05
time zone get	: /C 09 73 73 7C 86 86 DU DA DA C/ GL GL GA
time delta set	¢ >
time_delta_get v	ALL  Chn_set connect input_db Path: C:\Users\ZB\Desktoj OpenFile Mesh
e9 ff 00 00 00 00 02 00 01 00 b6 0b 00 c0 02 00 04 00	directed VLART USB output_db GwReset GwHeshOta GwOtaSelf Prov Close

Figure 24.10: ota reboot

#### 24.3.2 OTA Code Flow Summary

- (1) The host computer loads fireware to the gw node.
- (2) The host computer sends FW\_DISTRIBUT\_START to notify gw of the start of ota. The unicast address list which need to be OTA and the group address for OTA is included in parameters of FW\_DISTRIBUT\_START.
- (3) gw receives the FW\_DISTRIBUT\_START, then callback to the corresponding function: mesh\_cmd\_sig\_fw\_distribut\_s -> mesh\_ota\_master\_proc() -> (Start to send firmware data)

### 24.3.3 Gateway OTA Flowchart

Gateway's processing flowchart in mesh\_ota\_master\_proc() is shown below:


Figure 24.11: Gateway ota flowchart

## 24.3.4 Mesh OTA Related Commands

The structure corresponding to the INI command is mesh\_bulk\_ini\_vc\_t:

Telink

T

```
typedef struct{
    u16 nk_idx;
    u16 ak_idx;
    u8 retry_cnt; // only for reliable command // for op "distribute start" of gateway mesh
    OTA, it is reliable retry interval for LPN. // retry_intv_for_lpn_100ms
    u8 rsp_max; // only for reliable command
    u16 adr_dst;
    u8 op;
    u8 par[MESH_CMD_ACCESS_LEN_MAX];
}mesh_bulk_cmd_par_t;
typedef struct{
```

u16 flag; mesh\_bulk\_cmd\_par\_t cmd; }mesh\_bulk\_ini\_vc\_t;

The following is a description of the commands used in the flowchart Gateway ota flowchart. Please refer to the "messages" section in the spec "MshDFU\_v1.0.pdf" and "MshMBT\_v1.0.pdf" for the parameters of each command.

## 24.3.4.1 FW\_DISTRIBUT\_START

The initiator (host) sends this command to the distributor, which receives it and starts executing the distribution of firmware.

#### Note

These distributor starts in the INI are private commands.

```
fw_distribution_start_all =a3 ff 00 00 00 00 00 00 01 00 83 19 00 c0
```

The opcode "83 19" is the spec-defined distributor start opcode, but when the first two bytes of the following argument are a group address, it is recognised in private format. See is\_par\_distribute\_start\_tlk() in mesh\_cmd\_sig\_fw\_distribut\_start() for more details, in order to be compatible with the earlier command format. When is\_par\_distribute\_start\_tlk() returns ture, the corresponding parsing format of the parameter area is:

```
typedef struct{
    u16 adr_group; // Destination address to be used when sending firmware data.
    u16 update_list[MESH_OTA_UPDATE_NODE_MAX]; // The unicast address list of the node to be
    upgraded
}fw_distribut_start_tlk_t;
```

The update\_list of fw\_distribution\_start\_all is empty, which means it needs to be auto-populated with the node list of the host computer's "Mesh" window.

If the update\_list for fw\_distribution\_start is not empty, the example is as follows:



CMD-fw\_distribution\_start\_0002 =a3 ff 00 00 00 00 00 00 01 00 83 19 00 c0 02 00 CMD-fw\_distribution\_start\_02\_04 =a3 ff 00 00 00 00 00 00 01 00 83 19 00 c0 02 00 04 00

The unicast address list of the node to be upgraded is specified by the INI command and does not need to be auto-populated. However, it should be noted that when the number of addresses to be populated exceeds the default value of MESH\_OTA\_UPDATE\_NODE\_MAX, you need to modify MESH\_OTA\_UPDATE\_NODE\_MAX.

Also, since when doing a Mesh OTA for an LPN, only one LPN node can be upgraded at a time, it is used:

LPN\_fw\_distrib\_ota\_start\_04 =a3 ff 00 00 00 00 32 00 01 00 83 19 00 00 04 00

The corresponding parsing format of the parameter area is also fw\_distribut\_start\_tlk\_t, except that the adr\_group should be set to 0, and update\_list[0] is the unicast address of the LPN. In addition, the reliable retry count is "32". "For LPN\_fw\_distrib\_ota\_start, it is not the reliable retry count, but the reliable retry interval in unit 100ms, so 0x32 here means 5000ms, please see the comment of member retry\_cnt of mesh\_bulk\_cmd\_par\_t. \_cnt.

#### 24.3.4.2 FW\_UPDATE\_METADATA\_CHECK

Distributor sends this command to the node to be upgraded, which contains the firmware id, which reads the contents of the second to fifth bytes of the new firmware, corresponding to pid (product id) and vid (version id), see get\_fw\_metadata() for details. When the node to be upgraded receives this command, it checks the received firmware id and its own pid vid for comparison, when the pid is the same, it replies with the value of METADATA\_CHECK status as success, which suggests that the distributor can carry out OTA, and if the pid is not the same, it replies with the value of not allowed to carry out OTA, for more details, please refer to mesh\_ cmd\_sig\_fw\_update\_metadata\_check() of mesh\_ota\_slave\_need\_ota() for more details, if you want to change to other rules, please modify this function ota\_is\_valid\_pid\_vid().

```
/**
 * @brief
                This function check if new firmware has a valid PID(product ID) and VID(Version
\leftrightarrow IS).
 * @param[in] p_fw_id
                             - firmware ID
 * @param[in]
               gatt_flag - 1: it is GATT OTA.
 * @return
 * @note
                for both GATT and MESH ADV OTA
 */
_USER_CAN_REDEFINE_ int ota_is_valid_pid_vid(fw_id_t *p_fw_id, int gatt_flag)
{
    #if (OTA_ADOPT_RULE_CHECK_PID_EN)
   // user can change this policy
    int accept = 0;
   if(p_fw_id->pid == fw_id_local.pid){
        #if OTA_ADOPT_RULE_CHECK_VID_EN
        sw_version_big_endian_t *p_new = (sw_version_big_endian_t *)p_fw_id;
```

```
sw_version_big_endian_t *p_local = (sw_version_big_endian_t *)&fw_id_local.pid;
u16 ver_new_little = get_little_end_version(p_fw_id->pid);
u16 ver_local_little = get_little_end_version(fw_id_local.pid);
if(ver_new_little > ver_local_little){
    accept = 1;
}
#else
accept = 1;
#endif
}
return accept;
#else
return 1;
#endif
```

#### 24.3.4.3 CFG\_MODEL\_SUB\_ADD

Telink

}

Corresponding flowchart Gateway ota flowchart The comment inside is "subscription set".

Distributor sends this command to the pending upgrade. When the pending upgrade node receives this command, it subscribes to the group number for SIG\_MD\_BLOB\_TRANSFER\_S, so that when the OTA sends the firmware data in the future, it can use the group address as the destination address, and send the firmware data to all of the pending upgrade nodes at the same time.

#### 24.3.4.4 FW\_UPDATE\_INFO\_GET

Distributor gets the firmware information of the node to be upgraded, which mainly contains information such as firmware id.

#### 24.3.4.5 FW\_UPDATE\_START

Distributor sends this command to the node to be upgraded to indicate that firmware update is to be started.

#### 24.3.4.6 BLOB\_INFO\_GET

Distributor sends this command to get information of the node to be upgraded. include block size, chunk size, transfer mode, etc.

#### 24.3.4.7 BLOB\_TRANSFER\_START

Distributor sends this command to the node to be upgraded to inform the node of the size of the new firmware and the block size and chunk size parameters to be used, and to start the BLOB data sending process.



#### 24.3.4.8 BLOB\_BLOCK\_START

The distributor sends this command to the node to be upgraded, informing the node which block data is about to be sent, etc.

#### 24.3.4.9 BLOB\_CHUNK\_TRANSFER

Distributor sends this command to the node to be upgraded to transmit firmware data.

#### 24.3.4.10 BLOB\_BLOCK\_GET

After the distributor transmitting firmware data is completed, send this command to query all the nodes to be upgraded to see if there is any packet loss, if there is, it will re-send BLOB\_BLOCK\_START and BLOB\_CHUNK\_TRANSFER to make up for the packet loss until all the nodes have finished receiving.

#### 24.3.4.11 FW\_UPDATE\_GET

After the distributor confirming that all the nodes to be upgraded have received the firmware data, send this command and the nodes to be upgraded will perform CRC checksums and return the checksum value.

## 24.3.4.12 FW\_UPDATE\_APPLY and FW\_UPDATE\_CANCEL

If the returned result is successful, the distributor sends FW\_UPDATE\_APPLY to the node to notify it to reboot and enable the new firmware. If the result is a failure, the distributor sends FW\_UPDATE\_CANCEL to the node, informing it to reboot and discarding the firmware data it just received.

## 24.4 Gatt master dongle mode mesh OTA (kma\_dongle)

The node to be upgraded is a non-LPN node.

Test conditions: 1 x 8269 dongle (burn 8269kma\_master\_dongle), 2 x 8258 dongles.

#### 24.4.1 Code Configuration

The default mesh ota function is not enabled on the node side, the way to enable it:change MD\_MESH\_OTA\_EN from 0 to 1, as follows:



Figure 24.12: Open MD\_MESH\_OTA\_EN

If ota upgrade selects the directly connected node as the distributor's mode, you need to change DISTRIB-UTOR\_UPDATE\_SERVER\_EN from 0 to 1, as shown below:



## 24.4.2 Networking Nodes

Through the master dongle connection pc, open tools tool for two 8258 dongle in turn for networking, networking success, connect one of them, click on the "MESH" button to open the mesh control interface, and automatically get all the current nodes, to confirm that all the nodes are displayed in the UI (because the subsequent mesh OTA node's unicast addr is based on the UI inside the access, that is, the source of the address list in the parameter area of the FW\_DISTRIBUT\_START).

B Telink master Found			
CMD sig_mesh_master.ini v INI Iightness get LDN lightness Mesh	BULKOUT ASCII V Log C fastbind 2 retr C2652>15:40:08:291 [INTO]:	y Clear Save Save	Scan rp_scan OTA Px test p model id: 0x1304
fw_info_g fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri fw_discri	Nodes       reliable       All       On       Off       Sur       Chat         get/set       0       On       Off       Sur       Chat       I       0       On       Off       Sur       Chat       I       I       On       Off       Sur       Chat       I       I       I       On       Off       Sur       Chat       I       I       On       Off       Sur       Chat       I       I       I       On       Off       Sur       Chat       I       I       I       On       Off       Sur       Chat       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I       I	schedule       Day         Year       © any         Custom       © custom         Jan       Feb         Jan       Feb         Month       Jan         Jual       Aug         Sep       Oct         Nov       Dec         Hour       © any hour         © any minute       Second         © every 15 minute       © every 15 seco         © every 20 minute       © custom         © custom       ©         Week       Mon         Mon       Tue         Week       Sat         Sun       Action         © Off <no action<="" td="">       Recall 0         set       set</no>	Time get time set time add 1 id scene_number delete load

Figure 24.14: mesh\_UI

## 24.4.3 Select New Firmware

Close the mesh page to return to the home page, as shown in the following figure, click on the search file button in the figure, select the target firmware, click on the confirmation, after the confirmation as shown in mark 2.



#### Figure 24.15: Select new firmware

## 24.4.4 Get Version

T

Click the ini command : fw-update-info-get-all to get the version of the devices currently on the network, as shown in the following figure, the version number of the two devices shows 32 38 (i.e., version 2.8), for details, please refer to the corresponding description of the gateway mesh ota.

Telink	SIG Mest		Developer	Handbook
ICIIIIK	210 1.1621	1 201	Developer	TIGHUUUUUK

🚯 Telink master Found V4.1.0.0	- D X
CHD sig_mesh_master.ini  INI BULKOUT ASCII	Log T AutoSaveLog 2 retry Clear Save Save V Hex T Adv Stop Scan rp_scan ex_scan OTA Rx test T fastbind Extend Adv: All 🗸
LPN_get_onoff lightness_get_Panl Note:retry count field of LPN distrib start is change LPN_fw_distrib_ota_start_04 LDN_initiator_start_v_apply4	<pre> fw update info get_all</pre>
LPN_initiator_start_v_only4 	<pre>&lt;0004&gt;16:07:51:412 [INFO]:(common)mesh_ota_master_proc state: 0x05, wait flag:0 &lt;0005&gt;16:07:51:412 [INFO]:(common)mesh_ota_master_proc state: 0x05, wait flag:0 &lt;0006&gt;16:07:51:412 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 10 2 00 83 08 00 01 &lt;0006&gt;16:07:51:441 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 10 2 08 30 80 00 01 &lt;0006&gt;16:07:51:443 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 10 2 00 10 2 00 30 80 00 01 &lt;0006&gt;16:07:51:443 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 10 2 00 10 2 00 00 10 &lt;0006&gt;16:07:51:443 [INFO]:(Basic)access r_mod 0x08083(FW UPDATE INFO GHT): 83 00 001</pre>
Note:these distribution start not for LPN. For LPN, p fw_distribution_start_all fw_distribution_start_0002 fw_distribution_start_02_04 fw_distribution_start_0001	<pre>&lt;0012&gt;16:07:51:582 [INFO]:(Basic)src:0002,dst:0001,ac RX:0983(FM_UPDATE_INFO_STATUS): 83 09 01 00 04 0 &lt;0013&gt;16:07:51:582 [INFO]:(cmd_rsp)Status Rsp 02 00 01 (0 83 05 01 00 04 01 00 41 00 00 &lt;0014&gt;16:07:51:582 [INFO]:(log win32)mesh tx reliable_stop: op 0x0808 sp_max 1, rsp_cnt 1 &lt;0015&gt;16:07:51:589 [LIB]:(Basic)access_cnd_fw_update_info_get &lt;0015&gt;16:07:51:589 [LIB]:(Basic)access_cnd_fw_update_info_get</pre>
distribution start end fw_distribution_suspend fw_distribution_cancel fw_update_metadata_check fw_update_act	<pre><pre>&lt;0017&gt;6:07:51:625 [INFO]: [Common/PARCCMM1 as if 00 00 00 00 00 00 00 00 00 00 00 00 00</pre></pre>
fu_pdate_start fu_pdate_start fu_pdate_ancel fu_pdate_apply blob_transfer_get	
blob_transfer_start blob_transfer_cancel blob_block_start blob_block_unk_transfer blob_block_get	
<pre>blob_info_get fw_initiator_start_verify_only_all fw_initiator_start_verify_apply_all fw_initiator_start_verify_only_02_04</pre>	
rv_initiator_start_verify_apply_02_04 fw_initiator_start_VC_test fw_distribution_cap_get fw_distribution_recw_get fw_distribution_recw_get	
fw_distribution_recv_del_all fw_distribution_upload_get fw_distribution_upload_start v	<  ALL  Chn_set connect input_db Path: C:\Users\Admin\Desi OpenFile Mesh
	directed  UART USB output_db GwReset GwMeshOta GwOtaSelf Prov Close

#### Figure 24.16: Get version

#### 24.4.5 OTA Start

Telink

T

When testing, choose one of the following 3 ways to test.

(1) Upper computer as distributor mode

The host will transmit firmware data directly to the target node. This mode requires the master dongle and the GATT connected node to be in GATT connection state at all times.

Double-click the ini command :CMD-fw\_distribution\_start\_all to start the mesh OTA.



Telink

CMD sig_mesh_master.ini V INI BULKOUT ASCII	Log   fastbind 2 retry Clear Save Save   Hex   Adv Stop Scan rp scan OTA Rx test
mesh bulk cmd debug	
lightness get LPN	<2827>15:53:08:970 [INFO]: (common) ExecCmd: a3 ff 00 00 00 02 00 00 c0 7d 07 00 13
lightness get Panel	<2028>15:53:14:483 [INFO]: (common) ExecCmd: a3 ff 00 00 00 02 00 00 c0 7d 08 00 c4
	<2829>15:53:19:999 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 00 c0 7d 09 00 f2
fw info get	<2830>15:53:25:515 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 00 c0 7d 0a 00 ed
fw info get all	<2831>15:53:31:037 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 00 c0 7d 0b 00 14
fw distribution get	<2832>15:53:36:553 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 00 c0 7d 0c 00 10
fy distribution start all	<2833>15:53:42:068 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 00 c0 7d 0d 00 5b
fw distribution start 0002	<2834>15:53:47:583 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 00 c0 7d 0e 00 e6
fw distribution start 02 03	<2835>15:53:53:098 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 00 c0 7d 0f 00 09
fw distribution start 0001	<2836>15:53:58:613 [INFO]:(cmd_name)access_cmd_obj_block_get
fy distribution stop	<pre>&lt;2837&gt;15:53:58:627 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 01 02 00 7e 11 22 33 44 55 66 77 88 00</pre>
fy distribution detail get	<pre>&lt;2838&gt;15:53:58:641 [INFO]:(Basic)the mesh access tx cmd is 0x007e : 11 22 33 44 55 66 77 88 00 00</pre>
fy undate get	<pre>&lt;2839&gt;15:53:59:010 [INFO]:(Basic)adr_src:0x0002,adr_dst:0x0001,access rx cmd is 0x9b7 : b7 09 00</pre>
fy undate prenare	<2840>15:53:59:023 [INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 b7 09 00
fr undate start	<2841>15:53:59:040 [INFO]:(log_win32)mesh_tx_reliable_stop: op 0x007e rsp_max 1, rsp_cnt 1
fw_update_start	<2842>15:53:59:060 [INFO]:(cmd_name)access_cmd_obj_block_get
fw_update_abort	<pre>&lt;2843&gt;15:53:59:075 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 01 04 00 7e 11 22 33 44 55 66 77 88 00</pre>
iw_update_appiy	<2844>15:53:59:090 [INFO]:(Basic)the mesh access tx cmd is 0x007e : 11 22 33 44 55 66 77 88 00 00
obj_transfer_get	<2845>15:53:59:169 [INFO]:(Basic)adr_src:0x0004,adr_dst:0x0001,access rx cmd is 0x9b7 : b7 09 00
bbj_transfer_start	<2846>15:53:59:186 [INFO]:(cmd_rsp)Status Rsp: 04 00 01 00 b7 09 00
obj_transfer_abort	<2847>15:53:59:205 [INFO]:(log_win32)mesh_tx_reliable_stop: op 0x007e rsp_max 1, rsp_cnt 1
obj_block_transfer_start	<2848>15:53:59:217 [INFO]:(cmd_name)access_cmd_obj_block_transfer_start
bbj_chunk_transfer	<pre>&lt;2849&gt;15:53:59:231 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 01 02 00 b7 05 11 22 33 44 55 66 77 88</pre>
OD] DIOCK get	<pre>&lt;2850&gt;15:53:59:242 [INFO]:(Basic)the mesh access tx cmd is 0x05b7 : 11 22 33 44 55 66 77 88 01 00 00 0</pre>
obj_inro_get	<2851>15:53:59:889 [INFO]:(Basic)rc data layer upper:segment tx success, map in ACK is: 07 00 00 00
	<pre>&lt;2852&gt;15:54:00:210 [INFO]:(Basic)adr_src:0x0002,adr_dst:0x0001,access rx cmd is 0x6b7 : b7 06 00</pre>
scheduler_get	<pre>&lt;2853&gt;15:54:00:225 [INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 b7 06 00</pre>
sched_action_get	<2854>15:54:00:247 [INFO]:(log_win32)mesh_tx_reliable_stop: op 0x05b7 rsp_max 1, rsp_cnt 1
sched_action_set_orr	<2855>15:54:00:263 [INFO]:(cmd_name)access_cmd_obj_block_transfer_start
sched_action_set_on	<pre>&lt;2856&gt;15:54:00:279 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 01 04 00 b7 05 11 22 33 44 55 66 77 88</pre>
sched_action_set_scenel	<pre>&lt;2857&gt;15:54:00:297 [INFO]:(Basic)the mesh access tx cmd is 0x05b7 : 11 22 33 44 55 66 77 88 01 00 00 0</pre>
	<pre>&lt;2858&gt;15:54:00:930 [INFO]:(Basic)rc data layer upper:segment tx success, map in ACK is: 07 00 00 00</pre>
time_set	<pre>&lt;2859&gt;15:54:00:944 [INFO]:(Basic)adr_src:0x0004,adr_dst:0x0001,access rx cmd is 0x6b7 : b7 06 00</pre>
time_get	<pre>&lt;2860&gt;15:54:00:955 [INFO]:(cmd_rsp)Status Rsp: 04 00 01 00 b7 06 00</pre>
time_zone_set	<pre>&lt;2861&gt;15:54:00:973 [INFO]:(log win32)mesh tx reliable ston: on 0x05b7 rsp max 1 rsp cnt 1</pre>
time_zone_get	<pre>&lt;2862&gt;15:54:00:985 [INFO]:(cmd name)Progress, block total: 39, surrent: 1, DTA progress: 6%</pre>
time_delta_set	<2863>15:54:00:998 [INFO]: (common) ExecCmd: a3 11 00 00 00 00 02 66 60 c0 10 10
time_delta_get	<2864>15:54:06:511 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 00 c0 7d 01 00 1b
time_role_set	<pre>&lt;2865&gt;15:54:12:026 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 00 00 00 c0 7d 02 00 42</pre>
time_role_get	-
scene store	۲
scene del	TAR DIST. DIST. MESH Dalas
Januar 11	USB Connect Fath: D. (JIG_ASA_RETER search_file
a3 ff 00 00 00 00 02 00 01 00 b6 0a 11 02 21 00 00 ff 00 c0	UART CATE_RESET Mesh_ota Gate_ota Prov Mesh Close

Figure 24.17: start mesh OTA

(2) The directly connected node acts as distributor and selects verify and apply mode.

Instead of directly transmitting the firmware data to the node to be upgraded, the host computer first quickly transmits the firmware to the directly connected node through GATT mode, and then the GATT connected node transmits the firmware to the node to be upgraded. during the transmission to the node to be upgraded, the host computer can disconnect the GATT connection.

Since verify and apply mode is selected, the directly connected node will send the firmware update apply command to the node to be upgraded after distributing the firmware and confirming that it has been received, so that the node to be upgraded can reboot and take effect of the new firmware.

Double-click the ini command:CMD-fw\_initiator\_start\_verity\_apply\_all to start the mesh OTA. Command Format:(refer to structure fw\_initiator\_start\_t)

HCI\_CMD\_MESH\_OTA\_INITIATOR\_START + distribute adress + distribute appkey index + distribute ttl + timeout + distribute transfer mode + group + upload ttl + upload timeout + upload blob id

i.e.: ab ff + 00 00 + ff ff + ff + ff ff + 05 + 00 00 + 00 c0 + ff + ff ff + 61 62 63 64 65 66 67 68

🛞 Telink master Found 🛛 V4.1.0.0	- 🗆 X
CMD sig_mesh_master.ini V INI BULKOUT ASCII	Log   AutoSaveLog   2 retry Clear Save Save   Hex  Adv Stop Scan rp_scan ex_scan   OTA Rx test
mesh_bulk_cmd_debug	
LPN_get_level	rastbind Extend Adv: All
LPN_get_onoff	<pre>c0297&gt;19-25-49-959 [INFO1-(common)FracCmd- a3 ff 00 00 00 00 02 00 04 00 66 17 00 11</pre>
lightness_get_Panel	(303)19:25:49:336 [INFO]: (common)Exected: a3 ff 00 00 00 00 02 00 04 00 66 18 00 90
Note:retry count field of LPN distrib start is change	(3309)19:25:49:714 [INFO]: (common)EverCmd: a3 ff 00 00 00 00 02 00 04 00 66 19 00 0e
LPN_fw_distrib_ota_start_04	(0315)19:25:50:087 [INFO]: (common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 la 00 62
LPN_initiator_start_v_apply4	<0321>19:25:50:462 [INFO]: (common) ExecOmd: a3 ff 00 00 00 00 02 00 04 00 66 1b 00 2a
LPN_initiator_start_v_only4	<0327>19:25:50:836 [INFO]: (common) ExecOmd: a3 ff 00 00 00 00 02 00 04 00 66 1c 00 56
	<0333>19:25:51:209 [INFO]: (cmd name)block sum: 1.cur: 0.chunk sum:719.cur:29. upload Progress:5%
fw_update_info_get	<0334>19:25:51:209 [INFO]: (common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 1d 00 08
fw_update_info_get_all	<0340>19:25:51:582 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 1e 00 15
fw_distribution_get	<0346>19:25:51:957 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 1f 00 00
Note: these distribution start not for LPN. For LPN, p	<0352>19:25:52:330 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 20 00 1b
fw_distribution_start_all	<0358>19:25:52:703 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 21 00 02
fw_distribution_start_0002	<0364>19:25:53:082 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 22 00 0a
fw_distribution_start_02_04	<0370>19:25:53:476 [INFO]: (common]ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 23 00 64
fw_distribution_start_0001	<0376>19:25:53:851 [INFO]:(cmd_name)block_sum: 1,cur: 0,chunk_sum:719,cur:36, upload Progress:6%
distribution start end	<0377>19:25:53:851 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 24 00 70
fw_distribution_suspend	<0383>19:25:54:223 [INFO]: (common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 25 00 29
fw_distribution_cancel	<0389>19:25:54:603 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 26 00 91
fw_update_metadata_check	<0395>19:25:54:973 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 27 00 23
fw_update_get	<0401>19:25:55:348 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 28 00 02
rw_update_start	<0407>19:25:55:725 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 29 00 28
fw_update_cancel	<0413>19:25:56:112 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 2a 00 19
rw_update_appiy	<0419>19:25:56:491 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 2b 00 2b
blob_transfer_get	<0425>19:25:56:865 [INFO]:(cmd_name)block sum: 1,cur: 0,chunk sum:719,cur:44, upload Progress:7%
blob_transfer_start	<0426>19:25:56:865 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 2c 00 ff
blob block start	<0432>19:25:57:239 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 2d 00 13
blob_block_start	<0438>19:25:57:618 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 2e 00 00
blob block get	<0444>19:25:58:005 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 2f 00 1a
blob_block_get	<0447>19:25:58:099 [INFO]:(Basic)src:0002,dst:ffff,ac RX:4e82(LIGHTNESS_STATUS): 82 4e ff ff
fy initiator start verify only all	<0448>19:25:58:099 [INFO]:(cmd_rsp)Status Rsp: 02 00 ff ff 82 4e ff ff
fw initiator start verify apply all	<0454>19:25:58:393 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 30 00 08
fw initiator start verify only 02 04	<0460>15:25:58:769 [INFO]: (common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 31 00 18
fw initiator start verify apply 02 04	<pre>&lt;0466&gt;19:25:59:148 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 32 00 e0</pre>
fw initiator start VC test	<pre>c0d72519:25:59:518 [INFU]:(cmd_name)block sum: 1, cur: 0, chunk sum:719, cur:51, upload Progress:84</pre>
fw distribution cap get	-0473-19:25:59:515 [INFU]: (COMMON) EXECUME: 33 ff 00 00 00 00 02 00 04 00 66 33 00 66
fw distribution recv get	<pre>sup(3513:25:53:592 [INFU]:(COMMON)ExeCuma: as if 00 00 00 00 02 00 04 00 66 34 00 02 sup(3513:25:00)</pre>
fw distribution recv add	-0405/15:26:00:201 [1N:0]:(Common)Executa: as II 00 00 00 02 00 04 00 66 35 00 30
fw distribution recv del all	
fw_distribution_upload_get	s
fw_distribution_upload_start ~	ALL T ohn sat connect input db Bath: C:\Users\Admin\Desi OnenFile Mach
ab ff 00 00 ff ff ff ff ff 05 00 00 c0 ff ff ff 61 62 6	directed T UART USB output db CuBocot CuMochOta CuBocot Sale
	GWKESEC GWRESHOLA GWOLASEIT Prov Close
	17 HILL COV



(3) The directly connected node acts as distributor and selects verify only mode.

The difference between this mode and the "select verify and apply mode" is that the directly connected node will not send the distribution apply command to the node to be upgraded after distributing the firmware and confirming that it has been received. Instead, it waits for the host to send the distribution apply command before the node to be upgraded takes effect with the new firmware.

Currently, for ease of use, when the host computer receives a report from the distributor (directly connected node) that the distribution is complete, the host computer will automatically trigger the distribution apply command to be sent to the distributor. if you want to cancel this automatic sending, you can modify the code of " mesh\_ota\_initiator\_proc() -> case INITIATOR\_OTA\_ST\_DISTR\_PRE\_APPLY". For example, change it to nothing, and wait until the apply command is sent manually, and then set it to INITIA-TOR\_OTA\_ST\_DISTR\_GET state.

Double-click the ini command:CMD-fw\_initiator\_start\_verity\_only\_all to start mesh OTA. Command Format: (The difference from verify apply mode is that the value of distribute transfer mode is not the same.)

HCI\_CMD\_MESH\_OTA\_INITIATOR\_START + distribute adress + distribute appkey index + distribute ttl + timeout + distribute transfer mode + group + upload ttl + upload timeout + upload blob id

i.e.: ab ff + 00 00 + ff ff + ff + ff ff + 01 + 00 00 + 00 c0 + ff + ff ff + 61 62 63 64 65 66 67 68

Telink

8 Telink master Found V4.1.0.0	- 🗆 X
CMD   sig_mesh_master.ini 💌 INI   BULKOUT ASCII 🔽	log 🗆 AutoSaveLog 2 retry Clear Save Save 📈 Hex 🗆 Adv Stop Scan rp_scan ex_scan OTA Rx test
mesh_bulk_cmd_debug ^	fastbind Extend Adv: All
LPN_get_level	
LPN_get_onoff	<0213>19:26:58:219 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 0b 00 00 A
lightness_get_Panel	<0219>19:26:58:594 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 0c 00 ad
Note:retry count field of LPN distrib start is change	<0225>19:26:58:968 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 0d 00 03
LPN_fw_distrib_ota_start_04	(231)19:26:59:341 [INFO]:(cmd name)block sum: 1 cur: 0 chunk sum:719 cur:14 upload Progress:34
LPN_initiator_start_v_apply4	(2232)16:26:56:341 [INFO]: (common ) zero(md: a3 ff 00 00 00 00 02 00 04 00 66 0a 00 48
LPN initiator_start_v_only4	(2238)15:26:55:716 [INFO]: (common) Precomd: a3 ff 00 00 00 00 00 00 00 00 00 00 00 00
fw update info get	(224) 15.27.00.04(1) [INFO] (Common) Execute: as 11 00 00 00 00 02 00 04 00 06 10 00 56
fw update info get all	<pre>&lt;0250-15.27.00.461 [INFO] (Common) Execute: as if 00 00 00 00 00 00 00 00 00 00 00 00 00</pre>
fw distribution get	<pre>&lt;0256*15:27:00:035 [INFO]: (Common) ExeCUEL: a3 FF 00 00 00 00 02 00 04 00 66 12 00 FB</pre>
Note: these distribution start not for LPN. For LPN. p	<0262-13:27:01:210 [INFO]: (Common) Execute: a3 FF 00 00 00 00 02 00 04 00 66 13 00 34
fw distribution start all	<pre>&lt;0268&gt;19:27:01:564 [INFO]:(Common)ExeCuta: a3 FF 00 00 00 00 02 00 04 00 66 14 00 EB</pre>
fw distribution start 0002	<0274>19:27:01:959 [INFO]: (common)ExecCmd: a3 ff 00 00 00 02 00 04 00 66 15 00 0b
fw distribution start 02 04	<0280>15:27:02:330 [INFO]: (cmd_name)block sum: 1, cur: 0, chunk sum:719, cur:22, upload Progress:44
fy distribution start 0001	<0281>19:27:02:330 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 16 00 1d
iw_distribution_start_ovdi	<0287>19:27:02:705 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 17 00 11
fu distribution suspend	<0293>19:27:03:078 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 18 00 90
fw_distribution_suspend	<0299>19:27:03:451 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 19 00 0e
fw unders metodete ebeeb	<0305>19:27:03:822 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 1a 00 62
rw_update_metadata_check	<0311>19:27:04:196 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 lb 00 2a
rw_update_get	<0317>19:27:04:573 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 1c 00 56
rw_update_start	<0323>19:27:04:941 [INFO]:(cmd_name)block sum: 1,cur: 0,chunk sum:719,cur:29, upload Progress:5%
fw_update_cancel	<0324>19:27:04:941 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 1d 00 08
fw_update_apply	<0330>19:27:05:320 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 le 00 15
blob_transfer_get	<0336>19:27:05:710 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 1f 00 00
blob_transfer_start	<0342>19:27:06:097 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 20 00 1b
blob_transfer_cancel	<0348>19:27:06:472 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 21 00 02
blob_block_start	<0354>19:27:06:849 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 22 00 0a
blob_chunk_transfer	<0360>19:27:07:229 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 04 00 66 23 00 64
blob_block_get	166>19:27:07:604 [INFO]:(cmd name)block sum: 1.cur: 0.chunk sum:719.cur:36, upload Progress:68
blob_info_get 截图(Alt	A) 67>19:27:07:604 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 00 04 00 65 24 00 70
<pre>fw_initiator_start_verify_only_all</pre>	<0373>19:27:07:983 [INFO]: (common) ExecCmd: a3 ff 00 00 00 02 00 04 00 66 25 00 29
<pre>fw_initiator_start_verify_apply_all</pre>	<0379>19:27:08:358 [INFO]: (common) ExecOmd: a3 ff 00 00 00 00 02 00 04 00 66 26 00 91
fw_initiator_start_verify_only_02_04	(1285)19:77:08:732 [INFO]: (common) Evec(md: a2 ff 00 00 00 00 02 00 04 00 65 27 00 23
fw_initiator_start_verify_apply_02_04	0291315-27-08-105 [INFO]: (common Execute: a2 ff 00 00 00 00 00 02 00 04 00 65 29 00 02
fw initiator start VC test	039713.27.05.479 [INFO]: (Common Execute: a2 ff 00 00 00 00 02 00 04 00 06 20 00 02
fw distribution cap get	CA103:115.27.05.451 [INFO]: (Common Execute: a2 ff 00 00 00 00 00 02 00 04 00 06 22 00 18
fw distribution recv get	CANOS-15.27.15.232 [INFO] (Common) Execute: as 16 00 00 00 00 00 00 00 00 00 00 00 00 00
fw distribution recv add	10405/15.27.10.225 [INPO]. (Common) Execute: as II 00 00 00 00 02 00 04 00 06 25 00 25
fw distribution recy del all	
fw distribution upload get	< >
fw_distribution_upload_start v	ALL  Chn_set connect input_db Path: C:\Users\Admin\Des: OpenFile Mesh
ab ff 00 00 ff ff ff ff ff 01 00 00 c0 ff ff ff 61 62 6	directed VART USB output_db GwReset GwMeshOta GwOtaSelf Prov Close
1	

Figure 24.19: verify only

The red box in the above figure is the progress reference value: cur:1 indicates that the current is the 1st bolck; every 256k is 1 block, 147k firmware is 147/256 (such as block total:1 in the figure), total chunk:719 indicates that a total of 719 chunks are to be transmitted, cur:4 indicates that the current transmitted chunk is the 4th, OTA process:7% indicates that the current progress of ota is 7%.

## 24.4.6 OTA Finish

Telink

OTA is completed, the two devices will be applied, the print page will appear distribution completed; flow completed, the VC tool page to stop printing, the tick in front of the log will be removed to close the log printing, at the same time will pop up a small prompt box log disable now, indicating that the OTA is complete. log off is In order to avoid the log being flushed out during the upgrade process, it is convenient to check the log and save it, as shown in the following figure.

😵 Telink master Found V4.1.0.0	-
CMD sig mesh master.ini	Log - AutoSaveLog 2 vetry Class Save Save W Hex - Adv Stop Scap vp scap ev scap OTA Dv test
	Log F monoreney [ reary oreat bave ave men boop loan 1p_scal ex_scal
<pre>~ ^ </pre>	fastbind Extend Adv: All
LPN_get_level	
LPN_get_onoff	<4876>09:41:12:822 [INFO]:(common)ExecCmd: a3 ff 00 00 00 00 02 01 02 00 83 14 00 00 01 00
Note: wetry court field of IDM distrib start is change	<4877>09:41:12:853 [INFO]: (Basic) access tx cmd 0x1483(FW_DISTRIBUT_RECEIVERS_GET): 83 14 00 00 01 00
LDN fw distrib ota start 04	<pre>&lt;4881&gt;09:41:13:023 [INFO]:(Basic)src:0002,dst:0001,ac RX:1583(FW_DISTRIBUT_RECEIVERS_LIST): 83 15 02 0</pre>
LPN initiator start v applv4	<pre>&lt;4882&gt;09:41:13:023 [INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 83 15 02 00 00 04 00 01 c0</pre>
LPN initiator start v onlv4	<4803>09:41:13:023 [LIB]:(Basic) node addr:0x0004,update phase:2, distribute progress: 964
	<4884>09:41:13:023 [INFO]: (log win32)mesh tx reliable stop: op 0x1483 rsp max 1, rsp_cnt 1
fw update info get	<pre><doi:10.00 00="" 01.00="" 01.00<="" td=""></doi:10.00></pre>
fw_update_info_get_all	<pre></pre>
fw_distribution_met	<pre><deploy-dil17-978 02.dt:0001.ac="" 2000)<="" [intel:(act)stress="" distribut="" intels="" pre="" receivers="" rx:1583(fm=""></deploy-dil17-978></pre>
Note:these d sig mesh tool X : LPN, p	<4892209:41:17:978 [INFO]: (cmd rsp)Status Rsp : 02 00 01 00 83 15 02 00 00 00 44 00 1 c8
fw_distribution	<4893>09:41:17:978 [LIB]: (Basic) node addr:0x0004.update phase:2. distribute progress: 100%
fw_distribution	<4894>09:41:17:978 [INFO]:(log win32)mesh tx reliable stop: op 0x1483 rsp max 1, rsp cnt 1
fw_distribution	<4895>09:41:22:838 [LIB]:(Basic)access cmd fw distr receivers get
fw_distribution Log disable now	<4896>09:41:22:838 [INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 01 02 00 83 14 00 00 01 00
Mesh OTA flow finished!	<4897>09:41:22:870 [INFO]:(Basic)access tx cmd 0x1483(FW_DISTRIBUT_RECEIVERS_GET): 83 14 00 00 01 00
fw_distribution	<pre>&lt;4901&gt;09:41:23:026 [INFO]:(Basic)src:0002,dst:0001,ac RX:1583(FW_DISTRIBUT_RECEIVERS_LIST): 83 15 02 0</pre>
TW_distribution	<pre>&lt;4902&gt;09:41:23:026 [INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 83 15 02 00 00 04 00 04 c8</pre>
fu update metad	<pre>&lt;4903&gt;09:41:23:026 [LIB]:(Basic) node addr:0x0004,update phase:8, distribute progress: 100%</pre>
fw update start 确定	<4904>09:41:23:026 [INFO]:(log_win32)mesh_tx_reliable_stop: op 0x1483 rsp_max 1, rsp_cnt 1
fw update cance	<4905>09:41:25:840 [LIB]: (Basic) access cmd fw_distribut_get
fw update apply	<4905>05:41:25:840 [INFO] (Common) Executa: as IF 00 00 00 00 00 00 20 02 00 83 18
blob transfer get	<pre>&lt;40(E)0(4)(2)(0)(1)(E)(E)(E)(C)(C)(C)(E)(E)(E)(E)(E)(E)(E)(E)(E)(E)(E)(E)(E)</pre>
blob_transfer_start	<pre>&lt;416-09-41-26-040 [INFO]: (Basic) sr sequence at rough access received &lt;416-09-41-26-040 [INFO]: (Basic) sr concert at rough access received</pre>
blob_transfer_cancel	<pre>&lt;4917&gt;09:41:26:040 [INFOl:(md rsn)Status Ren : 02 00 01 00 83 1d 00 04 00 c0 ff ff ff ff</pre>
blob_block_start	<4918>09:41:26:040 [LIB] (Basic) distribution completed !
blob_chunk_transfer	<4919>09:41:26:040 [INFO]:(log_win32)mesh_tx_reliable_stop: op 0x1883 rsp_max 1, rsp_cnt 1
blob_block_get	<4921>09:41:26:117 [LIB]: (Basic) access_cmd_fw_distribut_cancel
DIOD_INTO_get	<4922>09:41:26:117 [INFO]:(common)ExecOnd: a3 ff 00 00 00 00 02 01 02 00 83 lb
<pre>imitiator_start_verify_only_all fu initiator_start_verify_only_all</pre>	<4923>09:41:26:147 [INFO]:(Basic)access tx cmd 0x1b83(FW_DISTRIBUT_CANCEL): 83 1b
fw initiator_start_verify_appry_arr	<4925>09:41:27:108 [LIB]:(Basic)mesh_tx_reliable_proc:retry cnt 2
fw initiator start verify apply 02 04	<4926>09:41:27:108 [INFO]: (Basic) access tx cmd 0x1b83(FW_DISTRIBUT_CANCEL): 83 lb
fw initiator start VC test	<4928>09:41:28:083 [LIB]: (Basic)mesh tx reliable proc:retry cnt 1
fw distribution cap get	<pre>vij22/05:11:20:000 [INSU]:[BaSIC]ACCESS tx cmmd UXIDSS(FW_DISIKIBUI_CANCED): 83 lb c/d01008:41:20:040 [INSU]:[Common transformed ty volumble creation on UNL02: cmm mark 0.</pre>
fw_distribution_recv_get	<pre>c403209-11-29-079 [INFO]: (company flow completed)</pre>
fw_distribution_recv_add	And the complete of the complete of
fw_distribution_recv_del_all	c
fw_distribution_upload_get	
fw_distribution_upload_start V	ALL _ chn_set connect input_db Path: C:\Users\Admin\Desl OpenFile Mesh
02 80 20 0b 00 07 00 04 00 08 01 00 ff ff 03 28	directed VUART USB output db GwReset GwMeshOta GuOtaSelf Drov Close
3	

#### Figure 24.20: kma ota finish

## 24.4.7 Recover Log

Telink

All devices will flash slowly for 6 seconds, and then it will reboot automatically, after reboot, confirm the log, if necessary, you can tap the "Save" button to save the log, VC tools "log" control check box, so that the print is back to normal.

							×
	-						
	$\frown$						
$\frown$							
THE HOUSE A DESCRIPTION OF A DESCRIPTION	-		I 🖬 Tan 🗖 Ada		-		
BULKOUT ASCII V Log I fastbing 2 retry Clear	Save	. Save	Mex   Adv	Stop	Scan	rp_scan	OTA RX test
	$\sim$	<u> </u>	3				



#### 24.4.8 Check for Success

Reconnect to the network, you can check the version through the above step 3 to confirm whether the version is upgraded successfully, as shown below.

Telink

			ALL
CMD sig_mesh_master.ini 💌 INI BULKOUT ASCI	1   <b>F</b>	Log	T fastbind 2 retry Clear Save Save V Hex Adv Stop Scan rp_scan OTA Rx test
mesh bulk cmd debug		120	(INFO) (Paris) the mast second by and is 000756 - 11 02 01 00 00 55
lightness get LPN		130	[INFO] (Basic) the mesh access to child is 0x0/b6. If 02 21 00 00 If
lightness get Panel		1000	[INFO]: (Common) mean tx reliable proceetry cht i
		200	[INFO]: (Basic) the mesh access tx cmd is 0x0/b6 : II 02 21 00 00 Fr
fw info get	E	202	[INFO] (Basic)mesh segments all packet received
fw info get all		222	[INFO] (Paci ) Pack and a la packat received
fw distribution get		324	[INFO] (Basic) add as on our out of a second of a seco
fw_distribution_start_all		247	[INFO] (and ran) Status Days - 04 00 01 00 b0 00 01 10 2 21 00 00 ff 01 22 20 44 55
fw distribution start 0002		1000	[INFO] (Ind_159)20005 sp of 0
fw distribution start 02 03		270	[INFO] (IDg_win52)mesh_ct_leliable_scop. op okonst isp_max i, isp_cht i
fw_distribution_start_0001		1000	[INFO] (Basic)mesh segments all packet received
fw_distribution_stop		106	[INFO] - (cmd ren)Status Ban - 04 00 01 00 b6 08 00 00 11 02 21 00 00 ff 11 22 33 44 55
fw_distribution_detail_get		118	[INFO] (common/mesh of master proc state 13
fw_update_get		140	[INFO]. (cmd name)access cmd fw distribut ston
fw_update_prepare		152	[INF0]: (common) ExecOnd: a3 ff 00 00 00 00 01 01 00 b6 0b 11 02 21 00 00 ff
fw_update_start		166	[INFO] (cmd nma) Pretomi, as if of total of current of of boods if
fw_update_abort		178	[INFO] (Basic)the mash access ty cmd is Oylbh6 - 11 02 21 00 00 ff
fw_update_apply		190	INFO: (Basic) the mesh access the cmain is 0x0006 - 00 11 02 21 00 00 ff
obj_transfer_get		105	[INFO]: (cmd rep) Status Rep
obj_transfer_start		118	[INFO]:(and name)mesh OTB completed!
obj_transfer_abort		111	· · · · · · · · · · · · · · · · · · ·
obj_block_transfer_start		571	[INFO]: (common)ExecCmd: a3 ff 00 00 00 00 02 00 01 00 b6 0a 11 02 21 00 00 ff 00 00
obj_chunk_transfer		586	[INFO]: (Basic) the mesh access tx cmd is 0x0ab6 : 11 02 21 00 00 ff 00 00
obj_block_get		599	[INFO]: (Basic) the mesh access tx cmd is 0x0cb6 : 01 11 02 21 00 00 ff
obj_info_get		513	[INFO]: (emd rsp)Status Rsp : 01 00 01 00 b6 0c 01 11 02 21 00 00 ff
		525	[INFO]: (cmd name)mesh OTA completed!
scheduler_get		544	[INFO]: (common) mesh ota master proc state: 3
sched_action_get		557	[INFO]: (cmd name)access cmd fw info get
sched_action_set_off		570	[INFO]: (common)ExecCmd: a3 ff 00 00 00 00 02 01 02 00 b6 01
sched_action_set_on		582	[INFO]: (Basic) the mesh access tx cmd is 0x01b6 NULL
sched_action_set_scene1		168	[INFO]: (Basic)adr src:0x0002,adr dst:0x0001,access rx cmd is 0x2b6 : b6 02 11 02 01 00 32 39
		383	[INFO]: (cmd rsp)Status Rsp : 02 00 01 00 b6 02 11 02 01 00 32 39
time_set		105	[INFO]: (log win32)mesh_tx_reliable_stop: op 0x01b6 rsp_max 1, rsp_cnt 1
time_get		121	[INFO]: (cmd_name)access_cmd_fw_info_get
time_zone_set		L37	[INFO]: (common) ExecCmd: a3 ff 00 00 00 00 02 01 04 00 b6 01
time_zone_get		154	[INFO]: (Basic) the mesh access tx cmd is 0x01b6 NULL
time_deita_set		268	[INFO]:(Basic)adr_src:0x0004,adr_dst:0x0001,access rx cmd is 0x2b6 : b6 02 11 02 01 00 32 39
time_deita_get		283	[INFO]: (cmd_rsp)Status Rsp: 04 00 01 00 b6 02 11 02 01 00 32 39
time_role_set		304	[INFO]:(log_win32)mesh_tx_reliable_stop: op 0x01b6 rsp_max 1, rsp_cnt 1
time_roie_get			
			III
scene_store		1	
	1	-	

Figure 24.22: Check for success

## 24.5 LPN Mesh OTA

## 24.5.1 LPN Mesh OTA Gateway Mode Operation Procedure

Test conditions: 1 x 8258 dongle (burning 8258\_mesh\_gw.bin), 2 x 8258 dongles (burning 8258\_mesh\_LPN.bin)

## 24.5.1.1 Code Configuration

- (1) Open MD\_MESH\_OTA\_EN
- (2) To shorten the mesh ota time, the macro EXTENDED\_ADV\_ENABLE can be set to 1 to support the extended broadcast packet mode, which should be noted is not a spec-defined mode.
- (3) If the gateway node is not selected as a friend, the gateway node should have FEATURE\_FRIEND\_EN set to 0.

<pre>#define FEATURE_LOWPOWER_EN #define FEATURE_PROV_EN #define FEATURE_RELAY_EN #define FEATURE_PROXY_EN #else</pre>	0 1 (0    SWITCH_ALWAYS_MODE_GATT_EN) 1
<pre>#if DU_ENABLE #define FEATURE_FRIEND_EN</pre>	0
#define FEATURE_FRIEND_EN	0 // WIN 32 should be support disable
#endlt #dafing FEATURE LOWDOWER EN	0





#### 24.5.1.2 Networking Nodes

Refer to the Gateway mesh ota networking process.

#### Note:

If the LPN node is not shown in the UI after the networking is completed, you can click this INI command to get it (you need to pay attention to whether the destination address is correct, the default is 0x0004), CMD-LPN\_get\_onoff. Another way is that the power down the LPN node and then re-powerup operation, the LPN will actively send the current status once, and the UI interface will display the LPN node.

#### 24.5.1.3 Select New Firmware

Refer to the Gateway mesh ota selecting new firmware process.

#### 24.5.1.4 Get Version

Refer to the Gateway mesh ota selection to obtain the version process.

#### 24.5.1.5 OTA Start

Click LPN\_fw\_distrib\_ota\_start\_04, then modify the last two bytes of the command window below to the unicast address of the LPN node. and then click "Enter" in keyboard of PC to send this INI command.

i.e. 02 00, and hit enter to send this command.



 $\times$ 

B Telink sig\_mesh -- Found V4.1.0.0

	Mesh			
CMD CI_node_gateway.ini V INI BOLKOUT ASCII	Mark		C	schedule
mesh_bulk_cmd_debug	Mesh	– Nodes reliable 🔻	Group	Year Dav
LPN_get_level	001 0002 On Off 🔾 100 🗹		All On Off Svr Clnt	G anu G anu
LPN_get_onoff		gevset		
lightness_get_Panel		0003	0 On Off	Custom Coust
Note:retry count field of LPN distrib start is change		Group S		Month
LPN_fw_distrib_ota_start_04		circup_5	1 On Off	I los I Fab I Mar Aar
		Group C	0.00	V Jall V FED V Mar V Apr
CDTP_OTS_GATT_ADV_ON			2 On Off	V Jul V Aug V Sep V Oct
CDTP_OTS_GATT_ADV_OFF		GrpDeIAII S	2 00 05	
			3 01 01	Hour
IW_update_info_get		GrpDeIAII_C	4 On Off	<ul> <li>any hour</li> <li>once a day</li> </ul>
IW_update_inro_get_all		0 10 L 0		Minute
IW_distribution_get		GetPub_S	5 On Off	C any minute C any
Note: these distribution start not for LPN. For LPN, p		Cashk Da		C every 15 minute C eve
IW_distribution_start_all		SECIAMBC	6 On Off	C eveny 20 minute
fw_distribution_start_0002		TTI		C Every 20 minute
fw_distribution_start_02_04		112	7 On Off	once an <u>nour</u> one
fw_distribution_start_0001		transmit	0.00	• custom  0   • cus
distribution start end			8 On Off	
Tw_distribution_suspend		Relay	0.0-0#	Week
TW_distribution_cancel			9 00 00	Mon V lue V Wed
IW_update_metadata_check		Friend	10 On Off	M Thu M Fri M Sat M S
IW_update_get				
IW_update_start		Proxy	11 On Off	Action
IW_update_cancel				On O Off O No action O
IW_update_appiy		Lightness	12 On Off	cet
blob_transfer_get		СЛ		301
blob_transfer_start		- CAT	13 On Off	tel antina III
blob_transfer_cancel		BELL		id action
blob block start		140	14 On Off	
blob block oct		GetCPS	11 0. 01	
blob_block_get			IS On Off	
prop_into_get		DelNode	16 On Off	
scheduler get				
sched action get			17 On Off	
sched_action_get				
sched_action_set_on			18 On Off	
sched action set scenel		/		
Sched_action_set_scener				
time set				
time get	<			>
time_zone_set v	ALL T	t connect innut an	Dath: C:\Users\Ac	min\Desi OpenFile
			Path:  0. (0.5215 (At	openrice Mesh
e8 ff 00 00 00 00 32 00 01 00 83 19 00 00 02 00	directed VART	USB output db	GwReset GwMesh0	a GwOtaSelf Prov Close
1	,			

Figure 24.24: LPN mesh ota start

## 24.5.1.6 OTA Finish

The log after successful ota is shown below.

B Telink sig\_mesh -- Found V4.1.0.0

CMD tl node gateway, ini	Log AutoSaveLog 2 retry Clear Save Save V Hex Adv Ston Scan rn scan ex scan OTA Py t	est
A The second sec	fastbind Extend Adv: All	
LPN_get_level		
LPN_get_onorr	<pre>&lt;1108&gt;17:28:15:103 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0002,op 0066(BLOB_CHUNK_TRANSFER):</pre>	b3 🔺
lightness_get_Panel	<1109>17:28:15:292 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 0066 (BLOB CHUNK TRANSFER):	b4
Note:retry count field of LPN distrib start is change	<1110>17:28:15:461 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 0066 (BLOB CHUNK TRANSFER) :	b5
LPN_fw_distrib_ota_start_04	<1111>17:28:15:646 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 0066 (BLOB CHUNK TRANSFER):	be
	<1112>17:28:15:832 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002.op 0066(BLOB CHUNK TRANSFER):	b7
CDTP_OTS_GATT_ADV_ON	<1113>17:28:17:780 [INFO]: (cmd rsp) Status Rsp : 02 00 01 00 68 ca b8 ca b9	
CDTP_OTS_GATT_ADV_OFF	<1114>17:28:17:780 [INFO]: (GATEWAY) HCI GATEWAY RSP OF CODE	
	: 91 81 02 00 01 00 68 ca b8 ca b9	
fw_update_info_get	<1115>17.28.17.983 [INFO]. (GATEWAY) and sendback src:0x0001 dst:0x0002 op 0066 (BLOB CHINK TRANSFER).	<b>b</b> 8
fw_update_info_get_all	<1116>17:28:18:182 [INFO]: (av up log)073 block sum: 1 cur: 0 chunk sum:698 cur:697 Progress:998 N	ITT.
fw_distribution_get	<1117-117-12-10-108 [INFO] (grt_ley] and sendback src:0.0001 det-0.0002 on 0.066 (BLOB CUINK TEANSFED).	hg
Note: these distribution start not for LPN. For LPN, p	<pre>clilips17:00-10-200 [INEO] (and replate site and set of the s</pre>	22
fw distribution start all	<pre>&gt;III0&gt;II.20.019.000 [INEQ].(CATEWAY DED OD CODE</pre>	
fw distribution start 0002		
fw distribution start 02 04	(1) 20 00 00 00 00 00 00 00 00 00 00 00 00	
fw distribution start 0001		
distribution start end	<1121>17:28:19:724 [INFO]: (cm FSp) Status RSp	
fw distribution suspend	<1122>17:28:19:724 [INFO]: (GATEWAY) HCL_GATEWAY_RSP_OP_CODE	
fw distribution cancel	: 91 81 02 00 01 00 67 40 00 00 00	
fw update metadata check	<1123>17:28:19:740 [INFO]: (GAIEWAY) cmd sendback src:0x0001 dst:0x0002, op 0c83 (FW OPDATE GET) NOLL	
fw undate get	<1124>17:28:20:719 [INFO]: (emd rsp)Status Rsp: 02 00 01 00 83 10 80 ff 00 00 00 11 22	33
fy undate start	<1125>17:28:20:719 [INFO]: (GATEWAY) HCI_GATEWAY_RSP_OP_CODE	
fy undate cancel	: 91 81 02 00 01 00 83 10 80 ff 00 00 01 1 22 33 44 55 66 77 88 00	
fy undate apply	<pre>&lt;1126&gt;17:28:20:734 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 0f83(FW_UPDATE_APPLY) NULL</pre>	
blob transfer get	<pre>&lt;1127&gt;17:28:22:445 [INFO]: (cmd_rsp)Status Rsp: 02 00 01 00 83 10 c0 ff 00 00 00 11 22</pre>	33
blob transfer start	<1128>17:28:22:445 [INFO]: (GATEWAY) HCI_GATEWAY_RSP_OP_CODE	
blob_transfer_spare	: 91 81 02 00 01 00 83 10 c0 ff 00 00 00 11 22 33 44 55 66 77 88 00	
blob_block_start	<1129>17:28:22:460 [INFO]: GATEWAY)mesh OTA success	
blob_block_start		
blob_chunk_transfer	<pre>&lt;1130&gt;17:28:22:476 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0001,op 1b83(FW_DISTRIBUT_CANCEL)NU</pre>	LL
blob_block_get	<pre>&lt;1131&gt;17:28:22:492 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0001,op 1d83(FW_DISTRIBUT_STATUS):</pre>	00
blob_inio_get	<1132>17:28:22:507 [INFO]: (cmd_rsp) Status Rsp 01_00_01_00_83_1d_00_00	
	<1133>17:28:22:507 [INFO]:(cmd_name)mesh OTA completed or get info ok!	
scheduler_get	<1134>17:28:22:507 [INFO]: (GATEWAY) HCI_GATEWAY_RSP_OP_CODE	
sched_action_get	: 91 81 01 00 01 00 83 14 00 00	
sched_action_set_off	<1135>17:28:22:553 [INFO]:(gw_vc_log)OTA, block sum: 0,cur: 0, chunk sum: 0,cur: 0, Progress:100% NU	LL
sched_action_set_on	<1136>17:28:35:419 [INFO]: (cmd_rsp) Status Rsp: 02 00 ff ff 82 08 ff 7f	
sched_action_set_scenel	<1137>17:28:35:442 [INFO]: (GATEWAY) HCI_GATEWAY_RSP_OP_CODE	
	: 91 81 02 00 ff ff 82 08 ff 7f	~
time_set	5	>
time_get		- ,
time_zone_set V	ALL Chn_set connect input_db Path: C:\Users\Admin\Des! OpenFile Me	sh
28 ff 00 00 00 32 00 01 00 83 19 00 00 04 00		_
	directed VART USB output_db GwReset GwMeshOta GwOtaSelf Prov Clo	se
,		

Figure 24.25: ota success

## 24.5.2 LPN Mesh OTA Gatt Master Dongle Mode

Test condition: 8269 dongle 1 (burning 8269kma\_master\_dongle), 8258 dongle 3 (two burning 8258\_mesh.bin, one burning 8258\_mesh\_LPN.bin)

## 24.5.2.1 Code Configuration

- (1) Open MD\_MESH\_OTA\_EN.
- (2) To shorten the mesh ota time, the macro EXTENDED\_ADV\_ENABLE can be set to 1 to support the extended broadcast packet mode, which should be noted is not a spec-defined mode.
- (3) If the ota upgrade selects the directly connected node as the distributor mode, you need to change DISTRIBUTOR\_UPDATE\_SERVER\_EN from 0 to 1.

## 24.5.2.2 Networking Nodes

Refer to the gatt master dongle mesh ota Selecting a Networking Node Procedure.

#### Note:

If the LPN node is not shown in the UI after the networking is completed, you can click this INI com-



mand to get it (you need to pay attention to whether the destination address is correct, the default is 0x0004):

CMD-LPN\_get\_onoff

Another way is that the power down the LPN node and then re-power-up operation, the LPN will actively send the current status once, and the UI interface will display the LPN node.

#### 24.5.2.3 Select New Firmware

Refer to gatt master dongle mesh ota select new firmware process.

#### 24.5.2.4 Get Version

Refer to gatt master dongle mesh ot selection to get the version process.

#### 24.5.2.5 OTA Start

(1) Upper computer as distributor mode

Click on LPN\_fw\_distrib\_ota\_start\_04, then modify the last two bytes of the command window below to be the unicast address of the LPN node, i.e., 02 00, and then click on the Enter key to send this INI command.



Figure 24.26: LPN mesh ota start

(2) The directly connected node acts as a distributor and the verify apply mode is selected.



Click LPN\_initiator\_start\_v\_apply4, then modify the last two bytes of the command window below to be the unicast address of the LPN node, i.e., 02 00, and then hit enter.

😵 Telink master Found V4.1.0.0	_ @	×
CMD sig_mesh_master.ini INI BULKOUT ASCII	Log T AutoSaveLog 2 retry Clear Save Save V Hex Adv Stop Scan rp_scan ex_scan OTA	Rx test
LPN get_onoff lightness_get_Panel Note:retry count field of LPN distrib start is change LPN fw distrib ota start 04 LPN initiator start v.app1y4 LDN initiator start v only4		^
Tw_update_info_get fw_update_info_get_all fw_distribution_get Note:these distribution start not for LPN. For LPN, p		
fw_distribution_start_all fw_distribution_start_0002 fw_distribution_start_02_04 fw_distribution_start_0201 		
<pre>fuldistribution_cancel fu_update_metadata_check fu_update_get fu_update_get fu_update_start fu_update_start</pre>		
fw_update_apply blob_transfer_get blob_transfer_start blob_transfer_cancel blob_block_start		
blob_chunk_transfer blob_block_get blob_info_get fw_initiator_start_verify_only_all fw_initiator_start_verify_apply_all		
<pre>fw_initiator_start_verify_only_02_04 fw_initiator_start_verify_apply_02_04 fw_initiator_start_VC_test fw_distribution_cap_get fw_distribution_recw_get fw_distribution_recw_ded</pre>		
fw_distribution_recv_del_all fw_distribution_recv_del_all fw_distribution_upload_get fw_distribution_upload_statt v	ALL     chn_set     connect     input_db     Path:     C:\Users\Admin\Dest     OpenFile	Mesh
	directed VART USB output_db GwReset GwMeshOta GwOtaSelf Prov	Close

#### Figure 24.27: LPN mesh ota verify apply mode

(3) Directly connected node as distributor and verify only mode selected.

Click LPN\_initiator\_start\_v\_only4, then modify the last two bytes of the command window below to be the unicast address of the LPN node, i.e., O2 OO, and then click Enter.

	Telink SIG Mesh SDK Developer Handbook
Talink matter Found V4100	- 0
CMD sig_mesh_master.ini 💌 INI BULKOUT ASCII 🗆 Log 🗆 AutoSaveLog 2 re	try Clear Save Save 🔽 Hex 🗆 Adv Stop Scan rp_scan ex_scan OTA Rx test
mesh_bulk_cmd_debug	All V
LPN get level	
lightness get Davel	^
LDN fy distrib ota start 04	
LDN initiator start v apply4	
LPN initiator start v onlv4	
fw update info get	
fw update info get all	
fw distribution get	
Note:these distribution start not for LPN. For LPN, p	
fw_distribution_start_all	
fw_distribution_start_0002	
fw_distribution_start_02_04	
fw_distribution_start_0001	
distribution start end	
fw_distribution_suspend	
fw_distribution_cancel	
IW_update_metadata_cneck	
IW_update_get	
fw undate apply	
blob transfer get	
blob transfer start	
blob transfer cancel	
blob block start	
blob_chunk_transfer	
blob_block_get	
blob_info_get	
fw_initiator_start_verify_only_all	
fw_initiator_start_verify_apply_all	
fw_initiator_start_verify_only_02_04	
fw_initiator_start_verify_apply_02_04	
fw_initiator_start_VC_test	
IW distribution cap get	
Tw_distribution_recv_get	
fur distribution recv_adu	V
fw distribution unload get	>
fw_distribution_upload_start v ALL v chu	a_set connect input_db Path: C:\Users\Admin\Desl OpenFile Mesh
'f ff 01 00 00 00 00 ff ff ff 61 62 63 64 65 66 67 68 02 00 directed	ART USB output_db GwReset GwMeshOta GwOtaSelf Prov Close

# Figure 24.28: LPN mesh ota verify only mode

## 24.5.2.6 OTA Finish

Telink

The log after successful ota is shown below.



Figure 24.29: ota success

# 24.6 QA

Telink

тI

# 24.6.1 What's the Best Way to Distinguish Between Different Equipment Types for OTA?

To initiate a mesh OTA on the VC host, you click the distribute start command, which has the parameter format:

```
typedef struct{
    u16 adr_group;
    u16 update_list[MESH_OTA_UPDATE_NODE_MAX];
}fw_distribut_start_t;
```

Figure 24.30: fw\_distribution\_start

- (1) Method 1: Normally, you need to put all the node addresses of all the nodes that need to be upgraded inside the update\_list array. Those that are not in the update list will not be OTA.
- (2) Method 2: By default, the device side will judge whether the PID(Product ID) of the new firmware and the current PID are the same or not, if not, it will reject the current OTA request. The corresponding function to judge is mesh\_ota\_slave\_need\_ota()->ota\_is\_valid\_pid\_vid().



So you can modify the update list parameter of the distribution start command to determine which node to initiate OTA for, for example, "fw\_distribution\_start\_02\_04" is to initiate OTA for 0x0002 and 0x0004:

CMD-fw\_distribution\_start\_02\_04 =e8 ff 00 00 00 00 00 00 01 00 83 19 00 c0 02 00 04 00

Introduction to the fw\_distribution\_start\_all Command:

fw\_distribution\_start\_all = =e8 ff 00 00 00 00 00 00 01 00 b6 0b 00 c0

#### Figure 24.31: fw\_distribution\_start\_all

On the VC UI, we don't want to have to manually add the node address to the distribute start command in order to initiate an OTA for ease of operation, but rather have a common command to start executing the OTA, so we have made a special marking, i.e., when the length of the update list is 0, we assume that all nodes displayed in this UI are added to the update list. As shown in the figure below, clicking "fw\_distribution\_start\_all" will perform OTA on nodes 0x0004 and 0x0007.

-Me	sh			_		_
001	0007	On	Off	$\bigcirc$	100	^
002	0004	On	Off	0	100	
Fig	gure 2	4.32	: sta	rt_a	II_UI	

#### 24.6.2 Ways to Differentiate between Different Devices?

We are currently using the PID (Product ID) to determine them.

## 24.6.3 Is it Possible to Confirm the Version before OTA?

On the originating side (master side), it is identified by reading the PID and CID of the composition data inside the Jason file.

On the node side (the upgraded side), mesh\_ota\_slave\_need\_ota() is used to judge the pid cid from the meta data to determine whether to upgrade or not, and returns O if no upgrade is needed. By default, our Demo SDK is to let the master decide which to upgrade, and the node side only check PID, if equal, return 1 to allow OTA.

#### 24.6.4 Can I Revert to a Previous Version?

The default is yes, which means that OTA downgrades are allowed. If you wish to disallow downgrades, just set OTA\_ADOPT\_RULE\_CHECK\_VID\_EN to 1.

# 24.6.5 What Needs to Be Done in FW in order to Differentiate between Device Types for Separate OTAs?

On the node side (upgraded side), ota\_is\_valid\_pid\_vid() is used to judge whether the product ID passed from the meta data is the same as the product ID of the new firmware to judge whether to upgrade or not, if the product ID is not the same, it returns 0 to indicate that it cannot be upgraded; if it is the same, it returns 1 to indicate that it can be upgraded.

# 24.6.6 What Needs to Be Done in FW in order to Distinguish FW Version Information for OTA?

On the node side (the upgraded side), ota\_is\_valid\_pid\_vid() is used to judge the version information passed from the meta data to determine whether to upgrade or not, and returns 0 if the upgrade is not needed, or 1 if the upgrade is needed.

# 24.7 Appendix Log

Gateway Mesh OTA Upgrade for Nodes with Unicast Addresses 0x0002 and 0x0004

(To save OTA time, the following log is based on enabling the private extended advertising packet mode, i.e., EXTENDED\_ADV\_ENABLE is set to 1.)

<0106>15:26:57:370 [INFO]:(common)ExecCmd: e8 ff 00 00 00 00 02 00 01 00 83 19 00 c0

<0107>15:26:57:418 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0001, op 1983(FW\_DISTRIBUT\_START): 00 c0 02 00 04 00

<0108>15:26:57:433 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0001,op 1d83(FW\_DISTRIBUT\_STATUS): 00 00

<0109>15:26:57:449 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 01 00 01 00 83 1d 00 00

<0110>15:26:57:449 [INFO]:(cmd\_name)mesh OTA completed or get info ok!

<0111>15:26:57:449 [INFO]: (GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 01 00 01 00 83 1d 00 00

<0112>15:26:57:480 [INFO]:(gw\_vc\_log)OTA, block sum: 0,cur: 0, chunk sum: 0,cur: 0, Progress: 0% NULL

<0113>15:26:57:496 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 0a83 (FW\_UPDATE\_METADATA\_CHE 00 01 00 32 38 00 00 00 00

<0114>15:26:57:512 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 02 00 01 00 83 0b 08 00

<0115>15:26:57:512 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 02 00 01 00 83 0b 08 00

<0116>15:26:57:620 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0004, op 0a83 (FW\_UPDATE\_METADATA\_CHE 00 01 00 32 38 00 00 00 00

<0117>15:26:57:666 [INFO]: (cmd\_rsp)Status Rsp\_\_\_\_\_: 04 00 01 00 83 0b 08 00

<0118>15:26:57:666 [INFO]: (GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 04 00 01 00 83 0b 08 00



<0119>15:26:57:823 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0002,op 1b80(CFG_MODEL_SUB_ADD): 02 00 00 c0 00 14
<0120>15:26:57:963 [INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 80 1f 00 02 00 00 c0 00 14
<0121>15:26:57:965 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE: 91 81 02 00 01 00 80 1f 00 02 00 00 c0 00 14
<0122>15:26:58:043 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0004,op 1b80(CFG_MODEL_SUB_ADD): 04 00 00 c0 00 14
<0123>15:26:58:244 [INFO]:(cmd_rsp)Status Rsp: 04 00 01 00 80 1f 00 04 00 00 c0 00 14
<0124>15:26:58:249 [INFO]: (GATEWAY)HCI_GATEWAY_RSP_OP_CODE: 91 81 04 00 01 00 80 1f 00 04 00 00 c0 00 14
<0125>15:26:58:260 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0002,op 0883(FW_UPDATE_INFO_GET): 00 01
<0126>15:26:58:306 [INFO]: (cmd_rsp)Status Rsp: 02 00 01 00 83 09 01 00 04 01 00 41 00 00
<0127>15:26:58:306 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE: 91 81 02 00 01 00 83 09 01 00 04 01 00 41 00 00
<0128>15:26:58:461 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0004,op 0883 (FW_UPDATE_INFO_GET): 00 01
<0129>15:26:58:538 [INFO]:(cmd_rsp)Status Rsp: 04 00 01 00 83 09 01 00 04 01 00 41 00 00
<0130>15:26:58:538 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE: 91 81 04 00 01 00 83 09 01 00 04 01 00 41 00 00
<0131>15:26:58:661 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0002,op 0d83(FW_UPDATE_START): ff 00 00 11 22 33 44 55 66 77 88 00 01 00 32 38 00 00 00 00
<0132>15:26:58:942 [INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 83 10 40 ff 01 00 00 11 22 33 44 55 66 77 88 00
<0133>15:26:58:942 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE: 91 81 02 00 01 00 83 10 40 ff 01 00 00 11 22 33 44 55 66 77 88 00
<0134>15:26:59:112 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0004,op 0d83(FW_UPDATE_START): ff 00 00 11 22 33 44 55 66 77 88 00 01 00 32 38 00 00 00 00
<0135>15:26:59:419 [INFO]:(cmd_rsp)Status Rsp: 04 00 01 00 83 10 40 ff 01 00 00 11 22 33 44 55 66 77 88 00
<0136>15:26:59:419 [INFO]:(GATEWAY)HCI_GATEWAY_RSP_OP_CODE: 91 81 04 00 01 00 83 10 40 ff 01 00 00 11 22 33 44 55 66 77 88 00
<0137>15:26:59:591 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002,op 0083 (BLOB_TRANSFER_GET) NULL
<0138>15:26:59:885 [INFO]:(cmd_rsp)Status Rsp: 02 00 01 00 83 03 00 01 11 22 33 44 55 66 77 88





<0161>15:27:02:785 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 0, Progress: 5% NULL

<0162>15:27:02:800 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 00 00 26 80 01 00 32 38 5d 01 4b 4e 4c 54 60 00 88 00 ae 80

<0163>15:27:03:000 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 1, Progress: 9% NULL

<0164>15:27:03:015 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 01 00 08 58 10 50 04 b1 04 b2 f8 87 00 a0 1a 09 1b 0a 91 02

<0165>15:27:03:217 [INFO]: (gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum: 22,cur: 2, Progress: 14% NULL

<0166>15:27:03:232 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 02 00 70 07 c0 46 00 65 00 f6 00 fe 07 0b 18 40 07 0b 18 40

<0167>15:27:03:448 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 3, Progress:18% NULL

<0168>15:27:03:463 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 03 00 10 6d c0 46 43 06 80 00 b8 00 80 00 ba 00 80 00 06 65

<0169>15:27:03:649 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 4, Progress:23% NULL

<0170>15:27:03:665 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 04 00 f9 c9 01 a2 04 0b 1a 40 34 a0 80 a1 ff 97 78 9f 01 60

<0171>15:27:03:867 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 5, Progress:27% NULL

<0172>15:27:03:882 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 05 00 02 a1 ff 97 15 9f 02 a0 a2 a1 ff 97 11 9f 27 a0 00 a1

<0173>15:27:04:084 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 6, Progress:32% NULL

<0174>15:27:04:099 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 06 00 c1 87 01 a3 63 40 d1 87 60 00 80 00 04 04 04 04 00 c

<0175>15:27:04:284 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 7, Progress:36% NULL

<0176>15:27:04:314 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 8, Progress:41% NULL

<0177>15:27:04:330 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 08 00 d1 87 bd a0 ff 97 bc 9d 01 ec a1 03 09 f6 09 fe bd a0

<0178>15:27:04:486 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur: 9, Progress:45% NULL

<0179>15:27:04:502 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 09 00 9c 02 5c c1 0a a9 00 c1 45 82 28 a9 00 c1 29 83 0d a9

<0180>15:27:04:701 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:10, Progress:50% NULL



<0181>15:27:04:716 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0xc000,op 0066 (BLOB\_CHUNK\_TRANSFER): 0a 00 54 e9 af 0a a4 e8 22 48 13 00 0b 03 23 40 30 6d 02 ac

<0182>15:27:04:903 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:11, Progress:54% NULL

<0183>15:27:04:919 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 0b 00 15 82 81 a5 ad f0 cf a3 00 a1 8b 87 05 a9 00 c1 69 81

<0184>15:27:05:120 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:12, Progress:59% NULL

<0185>15:27:05:135 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 0c 00 00 c1 01 82 28 a9 00 c1 5e 81 08 a5 3f a3 00 a1 20 87

<0186>15:27:05:320 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:13, Progress:63% NULL

<0187>15:27:05:336 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 0d 00 fc a3 00 a1 be 86 09 a9 00 c1 df 80 04 a9 00 c1 a3 81

<0188>15:27:05:521 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:14, Progress:68% NULL

<0189>15:27:05:537 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 0e 00 40 a1 57 86 82 a5 6d f0 cf a3 10 a1 52 86 a7 0d fc a3

<0190>15:27:05:742 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:15, Progress:72% NULL

<0191>15:27:05:757 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 0f 00 00 a1 ef 85 25 ec f3 a3 00 a1 eb 85 74 0d fc a3 00 a1

<0192>15:27:05:946 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:16, Progress:77% NULL

<0193>15:27:05:961 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 10 00 88 85 43 0d fc a3 02 a1 84 85 c4 a5 ad f0 fc a3 02 a1

<0194>15:27:06:149 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:17, Progress:81% NULL

<0195>15:27:06:164 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 11 00 c1 a5 ad f0 cf a3 10 a1 1c 85 25 ec f3 a3 04 a1 18 85

<0196>15:27:06:367 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:18, Progress:86% NULL

<0197>15:27:06:382 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 12 00 cf 99 f2 a0 08 a1 ff 97 cb 99 04 a0 00 a1 ff 97 7f 9b

<0198>15:27:06:586 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:19, Progress:90% NULL

<0199>15:27:06:602 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 13 00 67 99 88 a0 ff 97 44 99 04 ec 87 a0 ff 97 40 99 1f a1

<0200>15:27:06:789 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:20, Progress:95% NULL



<0201>15:27:06:804 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 14 00 00 a3 23 50 3b 40 42 06 13 40 04 6c 90 06 f0 6d 66 00

<0202>15:27:06:991 [INFO]:(gw\_vc\_log)OTA, block sum: 1,cur: 0, chunk sum:22,cur:21, Progress:99% NULL

<0203>15:27:07:007 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0xc000,op 0066(BLOB\_CHUNK\_TRANSFER): 15 00 11 50 a5 a1 19 0a 11 40 01 b2 13 40 10 6d 18 08 ff 97

<0204>15:27:07:195 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0x0002, op 0583 (BLOB\_BLOCK\_GET) NULL

<0205>15:27:07:226 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 02 00 01 00 67 c0 00 00 d0 00 07

<0206>15:27:07:226 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 02 00 01 00 67 c0 00 00 d0 00 07

<0207>15:27:07:396 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0x0004, op 0583 (BLOB\_BLOCK\_GET) NULL

<0208>15:27:07:475 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 04 00 01 00 67 c0 00 00 d0 00 07

<0209>15:27:07:475 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 04 00 01 00 67 c0 00 00 d0 00 07

<0210>15:27:07:614 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0xc000, op 0066 (BLOB\_CHUNK\_TRANSFER): 07 00 10 65 0c f6 24 fe a1 f0 23 f1 19 03 21 03 09 f6 09 fe

<0211>15:27:07:833 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 0583 (BLOB\_BLOCK\_GET) NULL

<0212>15:27:07:864 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 02 00 01 00 67 40 00 00 d0 00

<0213>15:27:07:864 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 02 00 01 00 67 40 00 00 d0 00

<0214>15:27:08:052 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0004,op 0583 (BLOB\_BLOCK\_GET) NULL

<0215>15:27:08:209 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 04 00 01 00 67 40 00 00 d0 00

<0216>15:27:08:209 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 04 00 01 00 67 40 00 00 d0 00

<0217>15:27:08:271 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 0c83 (FW\_UPDATE\_GET) NULL

<0218>15:27:08:582 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 02 00 01 00 83 10 80 ff 01 00 00 11 22 33 44 55 66 77 88 00

<0219>15:27:08:582 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 02 00 01 00 83 10 80 ff 01 00 00 11 22 33 44 55 66 77 88 00

<0220>15:27:08:754 [INFO]:(GATEWAY)cmd sendback src:0x0001 dst:0x0004,op 0c83(FW\_UPDATE\_GET)NULL

<0221>15:27:09:035 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 04 00 01 00 83 10 80 ff 01 00 00 11 22 33 44 55 66 77 88 00

<0222>15:27:09:035 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 04 00 01 00 83 10 80 ff 01 00 00 11 22 33 44 55 66 77 88 00

<0223>15:27:09:219 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0002, op 0f83 (FW\_UPDATE\_APPLY) NULL

<0224>15:27:09:530 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 02 00 01 00 83 10 c0 ff 01 00 00 11 22 33 44 55 66 77 88 00



<0225>15:27:09:530 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 02 00 01 00 83 10 c0 ff 01 00 00 11 22 33 44 55 66 77 88 00

<0226>15:27:09:717 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0004, op 0f83 (FW\_UPDATE\_APPLY) NULL

<0227>15:27:09:999 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 04 00 01 00 83 10 c0 ff 01 00 00 11 22 33 44 55 66 77 88 00

<0228>15:27:09:999 [INFO]:(GATEWAY)HCI\_GATEWAY\_RSP\_OP\_CODE: 91 81 04 00 01 00 83 10 c0 ff 01 00 00 11 22 33 44 55 66 77 88 00

<0229>15:27:10:186 [INFO]:(GATEWAY)mesh OTA success

<0230>15:27:10:201 [INFO]: (GATEWAY) cmd sendback src: 0x0001 dst: 0x0001, op 1b83 (FW\_DISTRIBUT\_CANCEL) NULL

<0231>15:27:10:217 [INFO]: (GATEWAY) cmd sendback src:0x0001 dst:0x0001,op 1d83(FW\_DISTRIBUT\_STATUS): 00 00

<0232>15:27:10:232 [INFO]:(cmd\_rsp)Status Rsp\_\_\_\_\_: 01 00 01 00 83 1d 00 00

<0233>15:27:10:232 [INFO]:(cmd\_name)mesh OTA completed or get info ok!

<0234>15:27:10:232 [INFO]: (GATEWAY) HCI\_GATEWAY\_RSP\_OP\_CODE : 91 81 01 00 01 00 83 1d 00 00

<0235>15:27:10:279 [INFO]:(gw\_vc\_log)OTA, block sum: 0,cur: 0, chunk sum: 0,cur: 0, Progress:100% NULL

# 25 Subnet Bridge

# **25.1 Function Introduction**

A Bluetooth network contains one or more subnets, and the subnets are usually isolated from each other using different network key encryption. In mesh 1.0, only devices that are on the same subnet and use the same network key can communicate with each other; devices between different subnets cannot communicate with each other. Secure isolation using subnets is a powerful feature: as shown in the figure below, in a hotel where each room is isolated from each other by subnets, devices in one room will not interfere with devices in another room.



Figure 25.1: Hotel subnet map

In some specific cases, there is a need for messages to be able to be transmitted between subnets. For example, the need for a guest in a hotel room to press the room cleaning service button to request service from the housekeeping team. Since the mesh key of each room cannot be the same (based on permission control considerations, the guest's mobile phone console can only have the key of the current room, not the key of other rooms, including the housekeeping team's key), in mesh 1.0, it is necessary to add a non-mesh gateway (e.g., TCP/IP) for each room to report the message to achieve this, and it is not possible to report the message to the housekeeping team through the current mesh network.

The Subnet bridge feature now enables communication between devices on different subnets, even if they do not share a common subnet and network key. By selecting a mesh node that supports two keys as a bridge point, and by configuring the bridge table of the bridge node, it is possible to achieve that after a guest presses the room clean button, the message can reach the housekeeping team through subnet

bridging, but other devices in the room that do not have a bridge table configured cannot communicate with devices outside of this room.

Field	Size (bits)	Description
Directions	8	Allowed directions for the bridged traffic
NetKeyIndex1	12	NetKey index of the first subnet
NetKeyIndex2	12	NetKey index of the second subnet
Address1	16	Address of the node in the first subnet
Address2	16	Address of the node in the second subnet

#### Figure 25.2: Bridging tables

In the bridging table shown above,

"Directions" specifies whether bi-directionality is supported.

"NetKeyIndex1, NetKeyIndex2" specifies between which two netkeys the bridge is established.

"Address1, Address2" specifies the two addresses between which the bridge is to be established.

When the network layer receives the message, it will check the bridge table to decide whether to bridge or not. If the bridge conditions are met, the message is re-encrypted and forwarded using another mesh network's NetKey specified in the bridge table.

An overview of the functions can also be found in https://www.bluetooth.com/mesh-feature-enhancementssummary/, this SIG official website description.

## 25.2 Subnet Bridging Principles

The subnet bridging feature forwards messages to specific subnets at the network layer by configuring a bridging table for nodes with multiple subnets.

When a node receives a message, if the source and destination addresses of the message are in the bridge table and the message is encrypted using the NetKey of the source subnet, it will decrypt the message using the NetKey of the source subnet and then re-encrypt the message with the NetKey of the target subnet before forwarding it to the specified subnet; if the source and destination addresses of the message are not in the bridge table, the message will not be forwarded to any other subnet.

## 25.3 Configuration

(1) Set the macro MD\_SBR\_CFG\_SERVER\_EN to 1 in the mesh\_config.h file.

911:	#define MD_SCHEDULE_EN	MD_TIME_EN_// because both of them save in same flash s
912:	#define MD PROPERTY EN	
913:	#define MD LOCATION EN	<pre>0 // location.sensor.batterv use same flash addr. but </pre>
914:	#define MD BATTERY EN	0
915:	#if ( PROJECT MESH LPN    PR	OJECT MESH SWITCH    PROJECT SPIRIT LPN )
916:	<pre>// not support directed forwarding</pre>	model and subnet bridge model.
917:	_ #else	u u u u u u u u u u u u u u u u u u u
918:	#define MD_DF_CFG_SERVER_EN	0 // directed forwarding server model.
919:	#define MD_DF_CFG_CLIENT_EN	0
920:	#define MD SBR CFG SERVER EN	<pre>0 // subnet bridge server model.</pre>
921:	#define MD_SBR_CFG_CLIENT_EN	0
922:	#endif	
923:	#define MD_SAR_EN	0
924:	<pre>#define MD_ON_DEMAND_PROXY_EN</pre>	0
925:	#define MD_OP_AGG_EN	0
926:	#define MD_LARGE_CPS_EN	0
927:	#define MD_SOLI_PDU_RPL_EN	MD_ON_DEMAND_PROXY_EN
928:	<pre>_ #ifPROJECT_MESH_SWITCH</pre>	
929:	#define MD_SERVER_EN	<pre>0 // SIG and vendor models</pre>

Figure 25.3: Open MD\_SBR\_CFG\_SERVER\_EN

- (2) Compile the 8258\_mesh project.
- (3) Burn at least 3 dongle nodes: light-room 1, light-room 2, cleaning unit.

## 25.4 Function Display

Scene display:

Console Center: Netkey Room 1 and 2, Public Service

ROOM 1 Light 1

Netkey Room 1

Netkey Public Service

Phone in Room 1: only Netkey Room1 ROOM 2 Light 2 Netkey Room 2

Netkey Public Service

Phone in Room 2: only Netkey Room2

Public Service Room: Cleaning Device only Netkey Public Service

Figure 25.4: Scene display

The Subnet Bridge feature allows a node with multiple subnets to be configured with a bridge table to forward messages to specific subnets.



Please refer to the "Subnet Bridge Setting" section of the chapter Android and iOS APP User Guide for detailed operation instructions.

Note: The sig\_mesh\_tool on PC does not have a UI to configure the node's subnet bridge parameters at this time.

reint semiconductor

# 26 Direct Forwarding

Direct Forwarding reduces the number of packets forwarded over the air by participating in the forwarding of commands at specified path nodes (routing tables). Direct Forwarding focuses on improving network utilisation, not on increasing transmission rates.

**Managed flood**: It is the propagation of mesh messages from the source outwards, similar to a stone thrown into water producing ripples that spread in all directions. The range of transmission is controlled through ttl. Whether the relay feature of a node is enabled determines whether the node will relay the message. This transmission mode is called managed flooding.

Managed flooding does not control the direction of message delivery and wastes bandwidth on parts of the network that are not related to the message. For example, if there are 2 switches in the middle of a large conference room that control the podium and the lights in the back row, when controlling the lights at the podium, messages are also retransmitted between the back row light nodes.



Figure 26.1: Directed Forwarding & Managed Flooding schema

**Routing table**: A routing refers to a path identifier that specifies all nodes that a message transmission path passes through from the starting point to the endpoint, and only nodes on the path can forward the message. The end point can be unicast, multicast and virtual address. Each routing node will save all path through it. These all paths are called routing table. Select several nodes in the network to form a path, and a routing may have one or more paths.

# 26.1 Routing Principles

When a message is sent in Direct Forwarding, it will check if the path exists first, and if it does, it will be sent as routed. Otherwise it will be sent as flooding and routing establishment will be triggered automatically. Messages sent by flooding are encrypted with a network key, and messages sent by routing are encrypted with a directed key (derived from the network key).

When a routing node receives a mesh message encrypted by a directed key, it will look up whether there is a corresponding path in the routing table according to the source and destination addresses. If the corresponding path is found, the message is forwarded by routing, otherwise the message will not be relayed. This achieves the purpose of forwarding messages along the specified route.

An overview of the functions can also be found in https://www.bluetooth.com/mesh-feature-enhancementssummary/, this SIG description from the official website.



In DF\_TEST\_MODE\_EN mode, nodes flash lights when forwarding messages encrypted with a DIRECTED key. The "path establishment" message is encrypted with the DIRECTED key, so all nodes flash their lights, indicating that all nodes forwarded the "path establishment" message. After the path is established, only the nodes on the path flash lights, indicating that only the nodes on the path forwarded this message.

# 26.2 Routing Table Types

Routing is divided into two ways: fixed routing and non-fixed routing:

- (1) Fixed routings are configured and managed by provisioner. after the network has been built, routing nodes are selected based on their location in the network. Usually, this requires professional personnel to install and configure.
- (2) Non fixed routings are automatically created and maintained by the sender which is the starting point of the path.

#### 26.2.1 Test Firmware Configuration

Open MD\_DF\_CFG\_SERVER\_EN and DF\_TEST\_MODE\_EN and compile the 8258\_mesh project.

#### 26.2.2 Fixed Routing

The fixed routing is configured and managed by the provisioner to forward with nodes on a specified path. So it is necessary to configure the following in advance on the app:

Go to page of Direct Forwarding–Direct Toggles, then open Direct Forwarding(Main), Direct Relay, Direct Proxy, Direct Friend to the nodes on the path. Note: If you don't enable Direct forwarding(main), the bottom will prompt "(relay) check direct forwarding first".

< Direct Toggle List	
adr-0x0002 cid-1102 pid-0100	2
Direct Forwarding(main)	$\checkmark$
Direct Relay	$\checkmark$
Direct Proxy	$\checkmark$
Direct Friend	$\checkmark$
<pre>adr-0x0004 cid-1102 pid-0100</pre>	
Direct Forwarding(main)	
Direct Relay	
Direct Proxy	
Direct Friend	
<pre>adr-0x0006 cid-1102 pid-0100</pre>	
Direct Forwarding(main)	
Di. (relay)check direct forwarding first	
Direct Proxy	

Figure 26.2: Fixed routing directangle toggle list interface

(DF means Direct Forwarding below)

- (a) Direct Forwarding(main) This is the main switch for DF. if disable, all DF features will be disable, include Direct Relay/Proxy/Friend.
- (b) Direct Relay if disable the DF messages will not be relayed to other nodes by current node which received the DF message. DF message is encryption by DF key.
- (c) Direct Proxy if disable, the message send from App can not be sent to other node by rounting, and will be sent by flooding.
- (d) Direct Friend due to LPN(low power node) do not support DF function, because it is a low power node which is not listenning the ADV all the time. and LPN receive message only from the Friend node which has establish friendship with current LPN. so if other node want to send message to a LPN by rounting, we need to enable "Direct Friend" function. then the message can be send to the Friend node by rounting, the Friend node will cache the message and then sent to the LPN when LPN wakeup.


The Direct forwarding interface allows you to add a fixed-routing path to the mesh network by clicking the Add Table button at the bottom. A path contains a start point and an end point, as well as the nodes through which the path passes. When a message is routed from the start point, all nodes on the path participate in forwarding the message; nodes not on the path ignore the message.

Note that commands sent from the phone are generally not used in fixed routing mode, but in Non fixed routing mode, because the mobile app location is not fixed, and the node with which it makes a GATT connection is also not fixed.

So the starting address of a path configuration for a fixed routing is usually a certain lamp node, for example as follows: the start of the path is 0x0006, the node on the path is 0x000e, and the node at the destination address is 0x0016.

Add Forwarding	Table	Forwarding Table Add	ed	A	pp Hor	nepage	
< Add Forwarding	Table	< Direct Forwarding		с	De Defaul	<b>/ice</b> t Mesh	+
Select origin device:	select >	Direct Toggles	>	ALL ON	ALL OFF	CMD	LOG
💡 Node-0006		Table List		•	•	•	•
Select target device:	select >	O. Origin		02(Pid-01)	04(Pid-01)	06(Pid-01)	08(Pid-01)
💡 Node-0016		0x0006					
Select nodes on the route: Node-000E	select >	Codoto Backward Waldsted III Nodes on route ♥ Node-000E		9 12(Pid-01)	9 14(Pid-01)	9 16(Pid-01)	
SAVE		ADD TABLE		Q Device	Ē	**	\$
	Figure	<b>26.3</b> : Adding a fi	xed r	outin	g		

Pressing the SW2 button on node 0x0006 will issue the Generic Onoff command, and since test mode is turned on, i.e., DF\_TEST\_MODE\_EN is turned on, the source address of the command is the node itself (0x0006), and the destination address is the destination address inside the path list of the first fixed routing,

i.e., 0x0016.

After the command is issued, we can see that the red LEDs of nodes 0x06, 0x0e and 0x16 on the path are blinking, and the LEDs of other nodes that are not on the path are not blinking, and the nodes that are not blinking indicate that they will not forward the message, which means that the routing function has been realized.

## 26.2.3 Non-fixed Routing

The Non-fixed routing do not require Direct Forwarding in the APP.

When the command initiator is the mobile app, the routing table will be created automatically and maintained by the node which is GATT connected with the mobile app. the GATT connected node proxy messages in a controlled flooding manner first, and then triggers routing establishment.

When the command initiator is not a mobile app, such as a time gateway, etc., the command sender (path origin) creates and maintains the routing information.



The path establishment message is network key encrypted, so all nodes flash their lights to indicate that the node forwarded the path establishment message. After the path is established, Directed Key was used for encryption, so only the nodes on the path will flash their lights, indicating that only the nodes on the path forwarded the command.

**Non-fixed routing establishment rules:** The figure below shows the schematic diagram of the two paths from PO (Path Origon) to PT (Path Target), where the shortest path which RSSI is greater than RSSI thread-hold are selected.



Figure 26.4: Non-fixed routing establishment rules

The test example uses the App to send commands as follows:

- (1) Add the node to the network using the app.
- (2) The mobile app is in the home page and automatically connects to a node in an initial state where no dynamic paths have been created yet.
- (3) The mobile app sends the Generic Onoff command to any non-directly connected node, for example, node 0x0016, which is in flood mode, so all nodes forward the command, that is, all nodes blink the red LED. At this time, it is detected that there is no path for the source and destination addresses, and the path establishment is triggered automatically.
- (4) After 5 seconds, the establishment of the path is completed. The established paths can be set to more than one, and in the test mode, only the optimal one path is selected.
- (5) The mobile APP sends a light on/off command again to the same destination address 0x0016, at which time only the nodes that are on the same path (including the start and end of the path) will blink at a frequency of 2Hz for 2 seconds.
- (6) The mobile app sends the Generic Onoff command to the node again, and only the node for the path from the previous step is blinking. If no messages are sent for a period of time (in test mode it is 12 minutes by default, in non-test mode it is 24 hours by default), the path is deleted. Sending the onoff light again will go to step 3 to re-establish the path.

# 27 Private-beacon

An overview of the functions can be found in https://www.bluetooth.com/mesh-feature-enhancementssummary/, This SIG official website description.

Please refer to "4.2.44 Mesh Private Beacon", "4.4.11 Mesh Private Beacon Server model" and "4.4.12 Mesh Private Beacon Client model" in "MshPRT\_v1.1.pdf" for the corresponding chapters of the spec. " can also be retrieved by typing private in the bookmark bar of the spec.

# 27.1 Application Background

In some scenarios, such as wearable and other devices that need to be mobile, if the mesh beacons sent out by such devices have plaintext static data, then these messages could be tracked and the location of that device could be tracked. So private-beacon is defined to solve such problems because in private-beacon mode, these beacon data will always change and be encrypted so that they cannot be tracked.

# **27.2 Function Introductions**

The private function ensures that static information in beacon messages is not visible to devices outside the network because it has been encrypted with a network key, increasing security and privacy.

## 27.2.1 Mesh Private Beacon

As shown in the following figure, Mesh Private beacon is added to the beacon, the decrypted content of this beacon is the same as that of the secure network beacon, and the function is also the same.

So the mesh private beacon is also sent after successful networking, unlike the secure network beacon, the private beacon's ivi index and flag appear encrypted, and the address of the adv is non-reslovable.

The unprovisioned device beacon does not have a corresponding private mode because the node is not yet networked and is not involved in being tracked.

Value	Definition
0x00	Unprovisioned Device beacon
0x01	Secure Network beacon
0x02	Mesh Private beacon
0x03-0xFF	Reserved for Future Use

#### Figure 27.1: Value of different beacon packages

The following figure describes each field of the private beacon, as well as the calculation process. The SDK corresponding function is mesh\_tx\_sec\_privacy\_beacon():



Figure 27.2: Segment of private\_ivi

## 27.2.2 Private Network Identity and Private Node Identity

As shown in the following figure, two new types of connectable broadcast packets have been added, which are Private network identity and Private Node Identity.

After decrypted, Private network identity is equivalent to Network ID and Private Node Identity is equivalent to Node Identity.

Type Value	Description
0x00	Network ID type
0x01	Node Identity type
0x02	Private Network Identity type
0x03	Private Node Identity type
0x04-0xFF	Reserved for Future Use

Figure 27.3: Identification type values

The following figure describes whether the device should send Node Identity or Private Node Identity when the node sends Node Identity state in the following combinations, where the first column "Node Identity state" and the second column "Private Node Identity state" are the conditions, and the third column "Advertising" is the packets that need to be sent.

Node Identity state	Private Node Identity state	Advertising
0x00	Does Not Exist	No Identity Advertising
0x00	Disable (0x00)	No Identity Advertising
0x00	Enable (0x01)	Private Node Identity
0x01	Does Not Exist or Disable (0x00)	Node Identity
0x02	Does Not Exist or Not Supported (0x02)	No Identity Advertising

#### Figure 27.4: Type value of private

The following figure describes whether the device should send Network ID or Private network identity when the node is in the transitive state of sending Network ID in the following combinations, where the first column of "Node Identity state" and the second column of "Private Node Identity state" are the conditions, and the third column of "Advertising" is the packets that need to be sent.

GATT Proxy state	Private GATT Proxy state	Advertising
0x00	Does Not Exist	No Proxy Advertising
0x00	Disable (0x00)	No Proxy Advertising
0x00	Enable (0x01)	Private Network Identity
0x01	Does Not Exist or Disable (0x00)	Network ID
0x02	Does Not Exist or Not Supported (0x02)	No Proxy Advertising

#### Figure 27.5: Type value of private

Private-beacon maintains state through two models: private-beacon server model, private-beacon client model.

In the SDK, the corresponding judgement function is mesh\_get\_identity\_type().

## 27.2.3 Introduction to Opcode

Please refer to "4.3.12 Mesh Private Beacon Messages" in "MshPRT\_v1.1.pdf" for the description of the corresponding section of the spec.

**PRIVATE\_BEACON\_SET:** Enable or disable the sending of private-beacon.

**PRIVATE\_GATT\_PROXY\_SET**: Enable or disable the sending of private gatt proxy, i.e., control the private node identity and the private network identity.

**PRIVATE\_NODE\_IDENTITY**: Enable or disable the sending of private node identity.

Element	Model ID	State	Message	Rx	Тх
Mesh Private	0xTBD	Mesh Private Beacon	PRIVATE_BEACON_GET	М	-
Beacon Main			PRIVATE_BEACON_SET	М	-
(Primary)			PRIVATE_BEACON_STATUS	-	М
		Private GATT Proxy	PRIVATE_GATT_PROXY_GET	М	-
			PRIVATE_GATT_PROXY_SET	М	-
			PRIVATE_GATT_PROXY_STATUS	-	М
		Private Node Identity	PRIVATE_NODE_IDENTITY_GET	М	-
			PRIVATE_NODE_IDENTITY_SET	М	-
			PRIVATE_NODE_IDENTITY_STATUS	-	М

Figure 27.6: Private beacon opcode

# 27.3 Test Steps

Telink

T

The firmware SDK turns on MD\_PRIVACY\_BEA and PRIVATE\_PROXY\_FUN\_EN.

For the test procedure using the App, please refer to section "33.4.5 Private beacon" of the the chapter Android and iOS APP User Guide.

Telink Semicondi

# **28 Minor Mesh Enhancements**

An overview of the functions can also be found in https://www.bluetooth.com/mesh-feature-enhancementssummary/, this SIG description from the official website<sub>o</sub>

# 28.1 Opcodes Aggregator Server Model

#### 28.1.1 Application Background

For example, when networking, many key bind commands can be aggregated and packaged into one command to save networking time.

#### 28.1.2 Function

The opcode aggregator allows different opcode messages under the same server model to be packaged into a single OPCODES\_AGGREGATOR\_SEQUENCE message type to be sent via the LTV structure (length,Opcode,Parameters).

The receiver node get all the opcode and parameters of the AGGREGATOR message by parsing the LTV structure, and packs all the response status into an OPCODES\_AGGREGATOR\_STATUS through the LTV structure to responde.

An opcode aggregator reduces interaction, processing and response time by compressing a series of messages into one.

Please refer to "4.4.19 Opcodes Aggregator Server model" and "4.4.20 Opcodes Aggregator Client model" in "MshPRT\_v1.1.pdf" for the corresponding chapters of the spec.

#### 28.1.3 Test Steps

- Firmware SDK Turn on MD\_OP\_AGG\_EN.
- Use the sig mesh app to network nodes.

When in the app bind process, the app compresses all the bind messages into a single message, reducing the interaction, processing and response time. This is shown in the following figure:

Item	$\pm \pm \circ$	Sequence Nu $\vee$	π. ~	Source Address V	B (	Mesh Control Information		
🗷 🚔 🎧 Mesh Provisioning Data					6	A Network		
🙀 🥋 Mesh Provisioning Complete						IV Index	0x0000001	
🛪 🎧 Mesh Config Composition Data Get (Page=255)		0x000E32	10	0x0001 (Unicast)	1 6	A Device Key		
🕫 😡 Mesh Config Composition Data Status (Page=0, SIG Models=Configuration Server, SIG Models=Health Server, SIG Models=	iels=Health			0x002B (Unicast)		Kev	4C75A831:54357064:423	
R Mesh Unknown (0x8072)				0x0001 (Unicast)	6	Adresses		
Haw Mesh Access Message (Seq=0x000E34, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=0	0, SegN=13)	0x000E34	10	0x0001 (Unicast)		Source Address	0x0001 (Unicast)	
🗛 🎧 Mesh Access Message (Seq=0x000E35, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=	1, SegN=13)	0x000E35	10	0x0001 (Unicast)		Destination Address	0x002B (Unicast)	
🖶 🙀 🎧 Mesh Access Message (Seq=0x000E36, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=2	2, SegN=13)	0x000E36	10	0x0001 (Unicast)	1	B 4 Devices		
🗛 🎧 Mesh Access Message (Seq=0x000E37, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=3	3, SegN=13)	0x000E37	10	0x0001 (Unicast)		Source	4C:FE:2B:64:96:F9 (Resol	
🙀 🎧 Mesh Access Message (Seq=0x000E38, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=4	4, SegN=13)	0x000E38	10	0x0001 (Unicast)		Destination	A4:C1:38:46:D6:BD	
Resh Access Message (Seq=0x000E39, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=	5, SegN=13)	0x000E39	10	0x0001 (Unicast)		Mech Arress Message		
🗛 🎧 Mesh Access Message (Seq=0x000E3A, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=0	6, SegN=13)	0x000E3A	10	0x0001 (Unicast)		y Mesh Access Message		
🗛 🎧 Mesh Access Message (Seq=0x000E3B, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=2	7, SegN=13)	0x000E3B	10	0x0001 (Unicast)	6	🗄 👫 Access Payload		
🙀 🙀 Mesh Access Message (Seq=0x000E3C, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=8	8, SegN=13)	0x000E3C	10	0x0001 (Unicast)		a OnCode	Unknown (0v8072)	
🕫 🎧 Mesh Access Message (Seq=0x000E3D, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=	9, SegN=13	0x000E3D	10	0x0001 (Unicast)	Data	1		4 <b>)</b>
🕫 🎧 Mesh Access Message (Seq=0x000E3E, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegD=	10, SegN=1.	0x000E3E	10	0x0001 (Unicast)	Dat	a type: Mesh Access Message 🔹	Search	
🙀 🙀 Mesh Access Message (Seq=0x000E3F, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegO=1	11, SegN=1.	0x000E3F	10	0x0001 (Unicast)		0 1 2 3 4 5 6	7 8 9 0123456789	
Mesh Access Message (Seq=0x000E40, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegD=	12, SegN=1.	0x000E40	10	0x0001 (Unicast)	0x0	000: 80 72 2B 00 28 00 00 0	0 00 30	
🙀 🙀 Mesh Access Message (Seq=0x000E41, NID=0x1D, SRC=0x0001.u, DST=0x002B.u, SeqZero=0x0E34, SegD=	13, SegN=1.	0x000E41	10	0x0001 (Unicast)		00A: 34 II B8 2B 63 DB E0 C	B 24 39 4. +C59	
Resh Segment Acknowledgement (OBO=No, SeqZero=0x0E34, BlockAck=0x00003FFF)		0x000008	10	0x002B (Unicast)	0.0	01E: 00 00 02 00 10 80 3D 2	B 00 00=+	
Be Mesh Encrypted Access Traffic (x 1, 0 s)				0x002B (Unicast)	0x0	Q28: 00 03 00 10 80 3D 2B 0	0 00 00=+	
Hesh Encrypted Access Traffic (x 1, 0 s)		0xA9AF0D	80	0xAF17 (Virtual)	0x0	032: 00 10 10 80 3D 2B 00 0	0 00 02=+	



# 28.2 Large Composition Data Models

## 28.2.1 Application Background

Some devices require a large amount of variable data to describe their composition, configuration data and other attributes. When the node has a lot of elements, for example, 100, then the length of the composition data will be very long, more than 380byte (a mesh message can only send a maximum of 380byte), then it is not possible to send the composition data status by a single message, then you need to segment to get the composition data status.

## 28.2.2 Function

Composition data consists of a series of pages, each of which is a composition state, where page 0 defines the elements of the node composition as well as the supported models. LARGE\_COMPOSITION\_DATA\_GET can be read starting from a byte of the specified page, as well as specifying the read length for segmented reads.

Please refer to "4.4.21 Large Composition Data Server model" and "4.4.22 Large Composition Data Client model" in "MshPRT\_v1.1.pdf" for the corresponding chapters of the spec.

## 28.2.3 Test Steps

- Firmware SDK Open MD\_LARGE\_CPS\_EN.
- Send LARGE\_COMPOSITION\_DATA\_GET for testing via the INI command.

# **28.3 SAR Configuration Models**

## 28.3.1 Application Background

When devices from different vendors use different default values for packet grouping behaviour for segment packets, such as retry interval, retry count, timeout time, etc., this may lead to inefficient mesh



message transmission within the mesh network. Unified configuration of SAR behaviours through the SAR Configuration Server model improves performance.

#### 28.3.2 Function Description

SAR means: Segmentation And Reassembly.

Packetisation and reorganisation parameters can be configured through SAR Configuration Server model related commands and can help to improve the efficiency of segment sending and receiving, especially if there are devices from multiple manufacturers in the network.

The SAR Transmitter contains parameters for sending sub-packets: the time interval between sub-packets, and parameters related to the retransmission interval and number of times.

The SAR Receiver contains parameters for receiving the sub-packet: parameters of responding seg ack and timeout.

Please refer to "4.4.15 SAR Configuration Server model" and "4.4.16 SAR Configuration Client model" in "MshPRT\_v1.1.pdf" for the description of the corresponding sections of the spec.

#### 28.3.3 Test Steps

- Firmware SDK Open MD\_SAR\_EN.
- Send SAR-related commands for testing via the INI command.

# 28.4 EPA(Enhanced Provisioning Authentication)

#### 28.4.1 Application Background

Adding another algorithm further enhances the security of data authentication when networking, especially when networking in static OOB mode.

- Each time a network is configured, the provisioner and provisionee will ask to regenerate the public key and private key for the network.
- The length of oob (including static oob / output oob / input oob) is changed from 16byte to 32byte.
- Use sha256 algorithm.

## 28.4.2 Function Description

The old algorithm name is BTM\_ECDH\_P256\_CMAC\_AES128\_AES\_CCM, and the new algorithm name is BTM\_ECDH\_P256\_HMAC\_SHA256\_AES\_CCM.

The main difference with spec V1.0 is:

• Each time a network is formed, the provioner and provisionee ask to regenerate the public key and the private key for the network to ensure that retransmitted messages is invalid.

- New Algorithm:EPA(BTM\_ECDH\_P256\_HMAC\_SHA256\_AES\_CCM,) uses the longer sha256 algorithm (the original algorithm was AES128), making it impossible to use a mainframe computer to perform traversal operations, etc.
- The length of oob (including static oob / output oob / input oob) in the new algorithm has been changed from 16byte to 32byte.
- The length of random and confirm in the new algorithm have been changed from 16 bytes to 32 bytes, making it impossible to use a mainframe computer for traversal operations.

#### Data format:

EPA: Provisioner determines whether a node supports EPA based on the parameter of capability reported by the node, as shown in the Algorithm field in the figure below. In order to be compatible with all providers, the node can choose to set bit0 and bit1 simultaneously to indicate that it supports both the old algorithm and the new algorithm, and then the provider will choose which way to enter the network.

#### 5.4.1.2 Provisioning Capabilities

The device sends a Provisioning Capabilities PDU to indicate its supported provisioning capabilities to a Provisioner. The format of the parameters for the Provisioning Capabilities PDU is defined in Table 5.20.

Field	Size	Description	
	(000013)		
Number of Elements	1	Number of elements supported by the device	
Algorithms	2	Supported algorithms and other capabilities (see Table 5.22)	
Public Key Type	1	Supported public key types (see Table 5.23)	
OOB Type	1	Supported, OOB Types (see Table 5.24)	
Output OOB Size	1	Maximum size of Output OOB supported (see Table 5.25)	1
Output OOB Action	2	Supported Output OOB Actions (see Table 5.26)	
Input OOB Size	1	Maximum size in octets of Input OOB supported (see Table 5.27)	
Input OOB Action	2	Supported Input OOB Actions (see Table 5.28)	

Table 5.20: Provisioning Capabilities PDU parameters format

#### The Number of Elements values are defined in Table 5.21.

Value	Description
0x00	Prohibited
0x01-0xFF	Number of elements supported by the device
Table E 04: Num	abox of Elements field values

Table 5.21: Number of Elements field values

#### The Algorithms values are defined in Table 5.22.

	Bit	Name	
B - (	0	BTM ECDH P256 CMAC AES128 AES CCM	
(	1	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	
	2-15	Reserved for Future Use	
	Table 5.22: Alg	orithms field values	

Figure 28.2: epa description

Please refer to "5.4.1.2 Provisioning Capabilities" in "MshPRT\_v1.1.pdf" for the description of the corresponding sections of the spec.

#### 28.4.3 Test Steps

• The firmware SDK turns on PROV\_EPA\_EN by default.

## 28.5 On-Demand Proxy Model

#### 28.5.1 Application Background

From a user experience point of view, enabling proxy function on all nodes is the preferred mode, i.e., all nodes are sending connectable broadcast packets, and the mobile APP can connect to the device at any time to control the network. However, when there are a lot of nodes, there will be a lot of connectable broadcast packets in the air, which will affect the available bandwidth, make it harder for the APP and the nodes to make a GATT connection, as well as increase the number of collisions between the subsequent mesh ADV messages and the connectable broadcast packets, which will have a negative impact on the performance of the mesh network. This is where the On-Demand Proxy Model can be used for optimisation.

#### 28.5.2 Function Description

When the On-Demand Proxy Model feature is enabled, the node is in the state of not sending connectable broadcast packets, and then the proxy client (mobile app) sends Solicitation PDUs to the node requesting the node to start broadcasting private beacons for a total of how long it will send private beacons, which is determined by g\_ mesh\_model\_misc\_save.on\_demand\_proxy = ON\_DEMAND\_PRIVATE\_GATT\_PROXY\_S (default is 30 seconds) to determine. This value can also be set by the APP or gateway via the On-Demand Proxy Model related commands.

After the device side receives the Solicitation PDU, it starts to send the private beacon of duration g\_mesh\_model\_misc\_save.on\_demand\_proxy (it is 30 seconds by default) if all the following 3 conditions are met as follows:

- (1) The node's current proxy and private proxy features are both set to support, but are in the disable state.
- (2) On-Demand Private GATT Proxy value in 0x01~0xff (set by ON\_DEMAND\_PRIVATE\_PROXY\_SET and eventually stored in g\_mesh\_model\_misc\_save.on\_demand\_proxy)
- (3) The Solicitation PDU was successfully decrypted using the network key.

Field	Size (octets)	Description
Opcode	2	The message opcode
On-Demand_Private_GATT_Proxy	1	New On-Demand Private GATT Proxy state

Table 4.249: ON\_DEMAND\_PRIVATE\_PROXY\_SET message structure

#### Figure 28.3: On-Demand Proxy descriptions



The corresponding chapters are described in "4.4.13 On-Demand Private Proxy Server Model" and "4.4.14 On-Demand Private Proxy Client Model" of "MshPRT\_v1.1.pdf".

Solicitation PDUs have a new sequence number mechanism to prevent replay.

#### 28.5.3 Test Steps

#### 28.5.3.1 Testing with APP

The firmware SDK requires MD\_ON\_DEMAND\_PROXY\_EN, PRIVATE\_PROXY\_FUN\_EN, and MD\_PRIVACY\_BEA to be turned on. for the sake of demonstration, the test was conducted with only one device on the current network.

- After the above functions are configured, the GATT proxy of the node is turned on by the SDK by default, so it does not satisfy the three conditions mentioned in the "Introduction to Functions" subsection of this chapter. Therefore, the node is always in the state of sending connectable broadcast packets. So the mesh app can connect to the device and configure it.
- Go to the app homepage, long press on the node that needs to be configured, then go to settings->private beacon and follow the settings as below to turn off config GATT Proxy and private GATT Proxy. As shown in the following figure:

13:27     Image: season       Private Beacon       Config GATT Proxy       Private GATT Proxy       Config Node Identity       Private Node Identity       Private Beacon       Private Beacon		$\sim$
Private Beacon         Config GATT Proxy         Private GATT Proxy         Config Node Identity         Private Node Identity         Private Beacon         Private Beacon	27	<b></b> 5G 🗩
Config GATT Proxy	Private E	Beacon
Private GATT Proxy	GATT Proxy	
Config Node Identity	GATT Proxy	
Private Node Identity	Node Identity	
Config Beacon	Node Identity	Ŏ
Private Beacon	Beacon	
	Beacon	
		27 Private E GATT Proxy GATT Proxy Node Identity Beacon Beacon

Figure 28.4: set private beacon

Note: If the customer wants to modify the SDK to have GATT proxy turned off for the default node, then change the mesh\_global\_var\_init() inside the



model\_sig\_cfg\_s.gatt\_proxy = FEATURE\_PROXY\_EN ? GATT\_PROXY\_SUPPORT\_ENABLE : GATT\_PROXY\_NOT\_SUPPORT

Change to

model\_sig\_cfg\_s.gatt\_proxy = FEATURE\_PROXY\_EN ? GATT\_PROXY\_SUPPORT\_DISABLE : GATT\_PROXY\_NOT\_SUPPORT

 After going through the above settings, exit the current mesh App and disconnect the app from the node. Scan with a common Bluetooth device scanning App (e.g. Light blue), you will see that after 30 seconds (ON\_DEMAND\_PRIVATE\_GATT\_PROXY\_S), the node no longer sends broadcast packets, at this time, open the mesh App to reconnect to the node, you can see that the app can't be connected to the node again, as shown in the figure below:

	18:51			ul 🗢 🗩	
	S	Dev		+	
	ALL ON	ALL OFF	CMD	LOG	
	X				
	0A(Pid-01)				
					-
$\mathcal{D}_{\mathcal{D}}$					
$\sum$	Q	Ē	융	\$	
	Single	Group	Network	Setting	

Figure 28.5: Disconnection effects

The main purpose of keeping the connectable broadcast packets sent for a set period of time after disconnection is to consider, for example, the convenience of reconnecting the App after an accidental disconnection.

• If you need to get the node to resend broadcast packets, you can click the Home -> Newtork -> solicitation PDU button as shown in the following figure to get the node to start sending broadcast packets. the destination address of the solicitation PDU is ADR\_ALL\_PROXY.

17:38			all 🗢 💽
	Net	work	
용 Mesh Default	<b>Info</b> Mesh		>
🥍 Scene	s		>
Direct	Forwarding		>
Mesh	ΟΤΑ		>
Proxy	Filter		>
((•)) Solicita	ation PDU		
Q Single	Group	Network	Setting
			/
JULE 28.	. <b>6</b> : Sen	d solicit	ation PI

After the app triggers to send solicitation, the app will continuously send solicitation pdu for 10s, which is customised by the app. The solicitation pdu is sent by sending broadcast packet from mobile phone, no need to perform GATT connection, the interval of sending packet is about 100ms. For details of solicitation pdu, please refer to this section solicitation-pdu-rpl-cfg-models

When the node receives the solicitation pdu, it will start sending connectable broadcast packets for the value set to g\_mesh\_model\_misc\_save.on\_demand\_proxy earlier.

During this time you can see that the node is already sending ADV by using a packet grabber or Bluetooth scanner. Then, our APP can automatically initiate a connection to the node and take control of the mesh network.

• The result after reconnecting with the node is shown below:

13:27			11 5G 🔲
Ç	Dev Default	rice Mesh	+
ALL ON	ALL OFF	CMD	LOG
1			
02(Pid-01)			
Q	Group	Retwork	<b>\$</b>
SILICIE	Oroup		occurs

If you need to modify the duration that a node sends connectable broadcasts via a command, you can follow the steps below:

 Long-press the node to be configured to enter the interface under settings->device config, click on set under on demand private proxy, and then enter a time value, which is the aforementioned g\_mesh\_model\_misc\_save.on\_demand\_proxy, and this value determines how long it takes for a node to stop sending connectable broadcast packets. The following figure shows this.

18:30	al 🗟 (							
1、Default TTL				~				
2、Relay & Rela	2、Relay & RelayRetransmit							
3、Secure Net	vork Beac	on		~				
4、GATT Proxy				$\sim$				
5, N Set on D								
6, F value(1 byte)	input new value value(1 byte):0x a							
	CEL	CONFIR	м	~				
8、Network Tra	Insmit			~				
9、on demand	private pr	оху		^				
The On-Demand Pr advertising with Pri 7.2) can be enabled reception of a Solic value:0x0	ivate GATT F vate Networ on demand itation PDU.	Proxy state indic k Identity type ( and can be trig	cates whet see Sectio gered upo	her in in				
		GET	SET					

Figure 28.8: set on demand private

• Afterwards, you can click "get" button to get the value you set before, as follows:

	18:30	al 🗟 I	•
<	Device Con	fig	
1,	Default TTL		~
2、	Relay & RelayRetransmit		~
3、	Secure Network Beacon		~
4,	GATT Proxy		~
5、	Node Identity		~
6,	Friend		~
7、	Key Refresh Phase		~
8,	Network Transmit		~
9 The adv 7.2) rece valu	on demand private proxy On-Demand Private GATT Proxy eritising with Private Network Iden can be enabled on demand and o eption of a Solicitation PDU. ee:0xA	state indicates whet tity type (see Section an be triggered upo GET SET	her on
e 28	<b>B.9</b> : get on den	nand priv	/ate

# 28.6 Solicitation PDU RPL CFG Models

The mobile phone sending a Solicitation command to the node can request the node to start sending private beacons. This Solicitation command is sent in ADV format, and it is not the same format as ordinary network message such as ONOFF. The main difference is that when encrypting and decrypting the solicitation PDU, the iv index is fixed to be 0. The reason is: after sharing the network, APP doesn't know the iv index, and the node doesn't send the private beacon, so there is no way to know the real iv index of the current network, and there is no way to ask the node to start sending connectable ADV through mesh network message, so there is no way to connect and control the nodes through proxy function.

In addition, the solicitation command has an independent sequence number, which is different from the sequence number of ordinary network PDUs, and there is a need to clear the solicitation sequence number cache. Therefore, spec defines Solicitation PDU RPL cfg models to clear the RPL (Relay Protect List) of solicitation PDUs cached on the device.

Field	Size (octets)	Description
Opcode	2	The message opcode
Address_Range	2 or 3	Unicast address range

Table 4.251: SOLICITATION\_PDU\_RPL\_ITEMS\_CLEAR message structure

Figure 28.10: PDU\_RPL configuration mods

Solicitation PDU RPL cfg models are for On-Demand Proxy Models.

# 29 Networked Lighting Control(NLC)

A functional overview can be found in the description of the official SIG website https://www.bluetooth.com/ mesh-feature-enhancements-summary/ in the section "3. Bluetooth® Mesh Device Profiles".

# 29.1 Application Background

Bluetooth® Mesh technology provides a rich set of features and options to implement many lighting and sensing applications. This has helped Bluetooth Mesh establish itself as the preferred technology for scalable commercial and industrial applications. However, the optional nature of Bluetooth Mesh features can cause challenges for implementers when they must decide which options to choose for their chosen product segments. If vendors operating in the same product segments choose a different set of options that do not work well with other peer products (e.g., mesh features chosen for light bulbs are not compatible with features selected for light switches), a situation can arise where product ecosystems do not interoperate, which degrades the user experience.

To address this issue, the Bluetooth SIG has come with the concept of Bluetooth Mesh Device Profiles. These profiles are new class of mesh specifications. Device Profiles define which options and features of the mesh specifications are mandatory for a certain kind of end product. The first suite of mesh device profiles is based on the lighting system architecture described in Building a Sensor-Driven Lighting Control System Based on Bluetooth Mesh whitepaper published in 2020. Collectively these profiles are called as Bluetooth Networked Lighting Control (NLC) Profiles, and they are defined as follows:

# 29.2 All NLC Profiles

## 29.2.1 NLC Profiles list

- (ALSNLCP) Ambient Light Sensor NLC Profile
- (BLCNLCP) Basic Lightness Controller NLC Profile
- (BSSNLCP) Basic Scene Selector NLC Profile
- (DICNLCP) Dimming Control NLC Profile
- (OCSNLCP) Occupancy Sensor NLC Profile
- (ENMNLCP) Energy Monitor NLC Profile

## 29.2.2 User Experience when Lights and Sensors work together

The NLC includes the processing linkage of sensor and light, and the main user experience of this linkage is summarized as follows:

• The user experience with ambient light sensor: when there is no ambient light sensor, the light is in the ON state, and the LED will output a maximum brightness of 65535. When there is a ambient light sensor and the light is in the ON state, the LED will output a relatively low brightness based on the current ambient light brightness through the PID algorithm, instead of the maximum brightness of 65536. The purpose is to save energy consumption. (This document will not demonstrate the

effectiveness of the PID algorithm for now, but will first demonstrate the different effects between day and night time.)

- The user experience with occupancy sensor: The light identifies whether there is someone moving by reading the status message sent by the occupancy sensor, if yes it will automatically triggers a change in the light's status, without the need for the app or speaker to receive sensor data first and then send control ONOFF commands to the light.
- The user experience with energy monitor sensor: Detect devices with high energy consumption, and if abnormal energy consumption is found, prompt whether to replace or improve the equipment to achieve the goal of energy conservation.

#### Note:

The sensor node sends a sensor status message instead of directly sending control commands such as onoff set and lightness set message, because it is required that many nodes need to listen to the status of this sensor, but the actions they take after receiving it may be different, such as performing light on/off operations or brightness adjustment operations. If the sensor sends a fixed control commands, then this function cannot be implemented.

# 29.3 Publish\_adress Configuration Methods

NLC often uses the publish set command to configure the destination address for messages sent by a button or sensor.

If you want to change the publish adress, you can set it through the ini command of the host computer, or you can configure it through the App. Please refer to "33.2.7 Device Setting (Switch Device)" of the chapter Android and iOS APP User Guide for the configuration method of the App. Device Setting (Switch device)".

The following is an example of the gateway sending the publish set command:

CMD-cfg pub set sig	=e8 ff	00 0	0 00	00	00 0	0 02	00	03	03	00	bb	00	00	00	ff	00	15	05	12
---------------------	--------	------	------	----	------	------	----	----	----	----	----	----	----	----	----	----	----	----	----

The data "O3 OO bb OO OO OO OO ff OO 15 O5 12" corresponds to the following structure. spec V1.1 corresponds to section <4.3.2.16 Config Model Publication Set>:

typedef struct{
 u16 ele\_adr;
 u16 pub\_adr;
 mesh\_model\_pub\_par\_t pub\_par;
 u16 model\_id; // u32 for vendor model
}mesh\_cfg\_model\_pub\_set\_t;

The publish address of the device can be changed by changing the following three variables.

ele\_adr: element adress, there is one or more elements on each device, so you need select the elements.

pub\_adr: publish adress, the destination address of the publish message.

model\_id: Each element has one or more models on it, so you have to select a model by its model id.



The following is the publish adress set by the ini command, the element adress of this setting is 0x0003, the publish ardess is 0x00bb, and the model id is 0x1205 (SIG\_MD\_SCENE\_C).

Low Energy Overview Message Log 📃 Spectrum					4 ▶ 🗙	Details	<b>Ф Х</b>
Protocol: Single 👻 All layers 🗧 🛹 📾 🏶 🖗 🖗 🐑 🦸 🖂 🧊 👄 🛛 2 items displayed			A Q	🕻 🂁 🔹 Seard	:h •   🔮	➤ All fields  All fields  Show in overview	Display 👻 🔝
Item	↓ ∨ Source Address ∨	Destination A $\vee$	Payload Length $\lor$	Time $\sim$	PDU $\vee$	Name	Value ^
B 🎲 Mesh Scene Recal Unacknowledged (Scene Number=1)	0x0002 (Unicast)	0x00AA (Unicast)	31 bytes	6.935 694 7	ADV_NONC	Destination Address	0x00BB (Unicast)
Some Recal Unacknowledged (Scene humber=2)	Dadd03 (Unicast)	0x008B (Unicast)	31 bytes	9.011 384 2	ADV_NONG	Stover Transport POU     Sogenetid Missipe     Application Key Flag     Application Key Identifier     POU     Mesh Application Information     Source Address     Source Address     Source Address     Source Address     Source Address     Source Address	No Yes OxOD SF 83 2C CC 03 29 55 63 ! Ox987665FF Ox0003 (Unicast) Ox0088 (Unicast)
						Data	9 <b>x</b>
						Data type:         Raw Data         0         1         2         3         4         5           0x00000:         02         1F         29         2F         15         38           0x00001:         02         AA         B7         D7         99         58           0x0012:         7A         A2         B7         D7         99         58           0x0012:         7A         D2         EE         B8         5C         FB           0x00118:         21         7D         98         76         65         FF	Search         •           6         7         8         012345678           C1         A4         18         .)/.8           BF         76         *[.vd           99         73         B0         vs.           7E         98         8         !}.ve.~

Figure 29.1: pub result

# **29.4 DICNLCP**

#### 29.4.1 Function

The Dimming Control NLC Profile (DIC) 1.0 – represents a wall slider, a dial, or a long-press switch function to dim lights up/down.

The corresponding spec is "DICNLCP\_v1.0.pdf", for detailed function description, please refer to this document.

The main purpose of DICNLCP is to define a device that can send Generic Onoff, Delta Level, and Move Level messages and can configure the destination address of the messages.

The demo SDK DICNLCP test uses 825x\_switch project of firmware demo SDK, the function is set to support DICNLCP and BSSNLCP at the same time. NLCP\_DIC\_EN and NLCP\_BSS\_EN are both equal to 1, customer can switch off NLCP\_BSS\_EN according to the need. NLCP\_DIC\_EN and NLCP\_BSS\_EN are both equal to 1. NLCP\_DIC\_EN and NLCP\_BSS\_EN can be switched off according to customer's need. it is at wake up mode as default beacause relay function is needed. If you want to disable relay and enter low power mode, just set SWITCH\_ALWAYS\_MODE\_GATT\_EN to 0 is enough. After disabling it, it will enter the low power mode.

## 29.4.2 nlc\_switch Button

For hardware, use Remote control board, see "Switch operation" section of this document for details.





Figure 29.2: Switch button introduction

1\_ON corresponds to RC\_KEY\_1\_ON; 1\_OFF corresponds to RC\_KEY\_1\_OFF ..... .4\_ON corresponds to RC\_KEY\_4\_ON; 4\_OFF corresponds to RC\_KEY\_4\_OFF.

A\_ON corresponds to RC\_KEY\_A\_ON; A\_OFF corresponds to RC\_KEY\_A\_OFF.

#### **29.4.3 Element Address**

The 825x\_switch project occupies 4 element addresses by default, the first element address is called ele\_adr\_primary, which is assigned during networking.

#### 29.4.4 nlc\_switch Button Functions

The demo SDK tests that DICNLCP only uses RC\_KEY\_1\_ON, RC\_KEY\_1\_OFF, RC\_KEY\_2\_ON, RC\_KEY\_2\_OFF, RC\_KEY\_3\_ON, RC\_KEY\_3\_OFF and RC\_KEY\_A\_ON, RC\_KEY\_A\_OFF to send commands.

When sending commands, RC\_KEY\_1\_ON, RC\_KEY\_1\_OFF use ele\_adr\_primary as the source address, and OxCOOO(NLC\_DICMP\_GROUP\_ADDR\_START) as the destination address by default; send Generic Onoff command.

RC\_KEY\_2\_ON, RC\_KEY\_2\_OFF use ele\_adr\_primary + 1 as source address, default use 0xC001 as destination address; send Generic Onoff command.

RC\_KEY\_3\_ON, RC\_KEY\_3\_OFF use ele\_adr\_primary + 2 as source address, default use 0xC002 as destination address; send Generic Onoff command.

The up/down/right/left/right buttons (RC\_KEY\_UP / RC\_KEY\_DN / RC\_KEY\_L / RC\_KEY\_R) are used to do mode selection without sending commands, and the RC\_KEY\_A\_ON and RC\_KEY\_A\_OFF buttons also are used to do mode selection. See the following nlc\_switch button onoff Command Mode, nlc\_switch button delta\_level Command Mode and nlc\_switch key move\_level command mode for how to do mode selection.



In addition for demo, RC\_KEY\_4\_ON and RC\_KEY\_4\_OFF are the keys to send scene store/recall command as BSSNLCP mode.

The above key function definition, customers can define and modify the key function according to the actual PCBA.

#### 29.4.4.1 nlc\_switch button onoff Command Mode

After the remote control board is powered on, the default value of select\_pub\_model\_key in SDK is equal to ONOFF mode. RC\_KEY\_1\_ON, RC\_KEY\_1\_OFF, RC\_KEY\_2\_ON, RC\_KEY\_2\_OFF, RC\_KEY\_3\_ON, RC\_KEY\_3\_OFF send the Generic Onoff command, see "else branch" of dicmp\_switch\_send\_publish\_command() below for details:

```
void dicmp_switch_send_publish_command(u32 ele_offset, bool4 onoff, u32 select_pub_model_key)
{
. . . . . .
   {
#if NLCP_DIC_EN
        if((RC_KEY_UP == select_pub_model_key)||(RC_KEY_DN == select_pub_model_key)){
            s32 delta = DICMP_LEVEL_DELTA_VALUE;
            if(!onoff){
                delta *= -1;
            }
            u16 pub_addr = dicmp_get_publish_addr(ele_offset, SIG_MD_G_LEVEL_C, 1);
            access_cmd_set_delta(pub_addr, 0, delta, CMD_NO_ACK, 0);
        }else if((RC_KEY_L == select_pub_model_key)||(RC_KEY_R == select_pub_model_key)){
            s16 move = DICMP_LEVEL_DELTA_VALUE;
            if(!onoff){
                move *= -1;
            }
            u16 pub_addr = dicmp_get_publish_addr(ele_offset, SIG_MD_G_LEVEL_C, 1);
            access_cmd_set_level_move(pub_addr, 0, move, CMD_NO_ACK, 0);
        }else{ // SIG_MD_G_ONOFF_C
            u16 pub_addr = dicmp_get_publish_addr(ele_offset, SIG_MD_G_ONOFF_C, 0);
            access cmd onoff(pub addr, 0, onoff ? G ON : G OFF, CMD NO ACK, 0);
        }
#endif
   }
}
```

## 29.4.4.2 nlc\_switch button delta\_level Command Mode

Press RC\_KEY\_UP or RC\_KEY\_DN to set the value of select\_pub\_model\_key to Delta Level mode. The RC\_KEY\_1\_ON, RC\_KEY\_1\_OFF, RC\_KEY\_2\_ON, RC\_KEY\_2\_OFF, RC\_KEY\_3\_ON, RC\_KEY\_3\_OFF will send the Delta Level command, which controls the level of the target node to increase or decrease by the specified value. For details, see spec "MshMDL\_v1.1.pdf", chapter "3.2.2.4 Generic Delta Set".



RC\_KEY\_1\_ON, RC\_KEY\_1\_OFF - The default destination address of the message is: 0xD000 (NLC\_DICMP\_GROUP\_ADDR\_START\_LEVEL\_MODEL)

RC\_KEY\_2\_ON, RC\_KEY\_2\_OFF - The default destination address for messages is: 0xD001

RC\_KEY\_3\_ON, RC\_KEY\_3\_OFF - The default destination address of the message is: 0xD002

#### 29.4.4.3 nlc\_switch button move\_level Command Mode

Press RC\_KEY\_L or RC\_KEY\_R to set the value of select\_pub\_model\_key to Move Level mode. The RC\_KEY\_1\_ON, RC\_KEY\_1\_OFF, RC\_KEY\_2\_ON, RC\_KEY\_2\_OFF, RC\_KEY\_3\_ON, RC\_KEY\_3\_OFF will send Move Level command.

Move level message simply means that the level value of the control node changes from the current value to the maximum value (when the move level parameter of the move command is positive) or to the minimum value (when it is negative) at the specified rate. See spec "MshMDL\_v1.1.pdf", chapter "3.2.2.6 Generic Move Set", etc. for details.

The default value for the destination address of a message is the same as the delta level mode.

#### 29.4.4.4 nlc\_switch button to Switch to on/off Command Mode

Pressing RC\_KEY\_A\_ON or RC\_KEY\_A\_OFF restores the value of select\_pub\_model\_key to ONOFF mode, which corresponds to sending the Generic Onoff command.

#### 29.4.5 Test Steps

#### 29.4.5.1 SDK Settings

• The firmware SDK uses the 825x\_switch project, sets LIGHT\_TYPE\_SEL to LIGHT\_TYPE\_NLC\_CTRL\_CLIENT, and compiles the firmware.

Note that at this point, the configuration related to model enablement uses the configuration information in nlc\_ctrl\_client\_model\_config.h instead of mesh\_config.h.

#### 29.4.5.2 Add to Network

To add the switch nodes to network, you can refer to section 12.5 Switch Engineering Long Press Handling Logic.

#### 29.4.5.3 Supplement of Group Add Command in App

Take using app to add colour temperature light to the living room as an example. For ease of use, it actually sends 4 group add commands to these 4 models by default: SIG\_MD\_G\_ONOFF\_S, SIG\_MD\_LIGHTNESS\_S, SIG\_MD\_LIGHT\_CTL\_S, SIG\_MD\_LIGHT\_CTL\_TEMP\_S.

Since lightness and temperature model are two independent state, and now belong to the same room. if we need to use level control commands such as level delta to control the room, we cannot use 0xC000 as



the destination address to control lightness, because the temperature model will also be controlled, which does not meet expectation.

Therefore when it is needed to use the level command, open the App's "home page" – "Setting" – "Enable Subscription Level Service model Id" function switch, and let the App add the group number 0xD000 to element 0 (element containing the lightness model); add the group number 0xD001 to element 1 (element containing the temperature model).

In this way, level control commands such as level delta can be used to control lightness via group number 0xD000 and colour temperature via group number 0xD001.

- If it is added to "Kitchen room", it corresponds to "OxDO10, OxDO11";
- If it is added to "Master bedroom", it corresponds to "0xD020, 0xD021".

#### 29.4.5.4 Key Default Function Test

Step-by-step details can be found in "nlc\_switch Button Functions", the 3 modes mentioned were tested.

- (1) App mode settings and grouping
  - Open the App's "home page" "Setting" "Enable Subscription Level Service model Id" function switch.
  - Use App to add the first group of the light node to "Living room" (involves group number 0xC000,0xD000,0xD001, see Supplement of Group Add Command in App for details).
  - Use the App to add the group part of the light node to the "Kitchen room" (involving group numbers 0xC001,0xD010,0xD011).
  - Use the App to add the third group of the light node to the "Master bedroom" (involving group numbers 0xC002,0xD020,0xD021).
- (2) Generic on/off test
  - Press RC\_KEY\_1\_ON or RC\_KEY\_1\_OFF to switch to onoff mode. (The remote control belongs to onoff mode by default when it is just powered on).
  - Pressing the keys RC\_KEY\_1\_ON and RC\_KEY\_1\_OFF controls the onoff state of the nodes belonging to the Living room (0xC000) group.
  - Pressing the keys RC\_KEY\_2\_ON and RC\_KEY\_2\_OFF controls the onoff state of the nodes belonging to the Kitchen room (0xC001) group.
  - Pressing the keys RC\_KEY\_3\_ON and RC\_KEY\_3\_OFF controls the onoff state of the nodes belonging to the Master bedroom (0xC002) group.
- (3) Generic Level Delta test
  - Press key RC\_KEY\_UP or RC\_KEY\_DN to switch to Delta Level mode.
  - The keys RC\_KEY\_1\_ON and RC\_KEY\_1\_OFF controls the increase or decrease of the lightness of the nodes belonging to the Living room (0xC000) group.
  - The keys RC\_KEY\_2\_ON and RC\_KEY\_2\_OFF control the increase and decrease of lightness of the nodes belonging to the Kitchen room (0xC001) group.
  - The keys RC\_KEY\_3\_ON and RC\_KEY\_3\_OFF control the increase and decrease of lightness of the nodes belonging to the Master bedroom (0xC002) group.
- (4) Generic Level Move test
  - Press key RC\_KEY\_L or RC\_KEY\_R to switch to Move Level mode.

- The keys RC\_KEY\_1\_ON and RC\_KEY\_1\_OFF control the nodes belonging to the Living room (0xC000) group to increase the lightness to the maximum or decrease it to the minimum value.
- The keys RC\_KEY\_2\_ON and RC\_KEY\_2\_OFF control the increase of the lightness to the maximum or the decrease to the minimum value for the nodes belonging to the Kitchen room (0xC001) group.
- The keys RC\_KEY\_3\_ON and RC\_KEY\_3\_OFF control the nodes belonging to the Master bedroom (0xC002) group to increase the lightness to the maximum or decrease it to the minimum value.

## 29.4.5.5 Configure the Publish Address Test for the Key

Configure the destination address of the command of a key by sending the publish set command. see Publish\_adress Configuration Methods for the description of publish.

The following is an example of the publish address for the level delta command that configures RC\_KEY\_1\_ON and RC\_KEY\_1\_OFF:

Method 1. The INI commands actually tested by the host computer are referenced below:

CMD-cfg\_pub\_set\_sig = e8 ff 00 00 00 00 00 00 02 00 03 02 00 01 D0 00 00 ff 00 15 03 10

This command sets the publish address of the level client model (0x1003) on element address (0x0002) in relink Semiconi the remote control to 0xD001.

Method 2. The App is tested as follows

18:3	5		al 🗢 🗊
<		Device Setting	
C	ONTROL		SETTINGS
eleAdr: 0x	0002		
model: 0x	1003		
pubAdr: 0x	D001		AddressList
			SEND
eleAdr: 0x	0003		
model: 0x	1001		
pubAdr: 0x	C001		AddressList
			SEND
eleAdr: 0x	0004		
model: 0x	1001		
pubAdr: 0x	C002		AddressList
			SEND
eleAdr: 0x	0005		
model: 0x	1001		
pubAdr: 0x	C003		AddressList
int.	Figure	e 29.3: app p	ub set

model: 0x1003 (Generic Level Client), because you want to configure the publish address of the level delta or level move command.

pubAdr: This is set to the destination address of the command sent from this button. Here, for example, it is set to 0xD001 to control the colour temperature of the lamp belonging to "Living room".

# 29.5 BSSNLCP

#### 29.5.1 Function Description

Basic Scene Selector NLC Profile (BSS) 1.0 – represents a wall switch or a wall station to select lighting scenes or turn the lights on/off.

Please refer to spec "BSSNLCP\_v1.0.pdf" for the corresponding chapter introduction and this document for detailed function introduction.

BSSNLCP Defines a device that can send the SCENE STORE/RECALL command and can configure the destination address of the messages.

#### 29.5.2 Hardware Introduction

The firmware SDK uses 825x\_switch project, for hardware we use remote control board, see "nlc\_switch Button"

#### **29.5.3 Button Functions**

The demo SDK DICNLCP test uses 825x\_switch project of firmware demo SDK, the function is set to support DICNLCP and BSSNLCP at the same time. NLCP\_DIC\_EN and NLCP\_BSS\_EN are both equal to 1, customer can switch off NLCP\_BSS\_EN according to the need. NLCP\_DIC\_EN and NLCP\_BSS\_EN are both equal to 1. NLCP\_DIC\_EN and NLCP\_BSS\_EN can be switched off according to customer's need. it is at wake up mode as default beacause relay function is needed. If you want to disable relay and enter low power mode, just set SWITCH\_ALWAYS\_MODE\_GATT\_EN to 0 is enough. After disabling it, it will enter the low power mode.

Test BSSNLCP sending command, currently the demo only uses RC\_KEY\_4\_ON, RC\_KEY\_4\_OFF key, other key definition please refer to "nlc\_switch-button-functions", customer can redefine according to the actual need.

RC\_KEY\_4\_ON sends SCENE\_STORE command and RC\_KEY\_4\_OFF sends SCENE RECALL command.

- RC\_KEY\_4\_ON uses ele\_adr\_primary as source address and OxCOOO(NLC\_DICMP\_GROUP\_ADDR\_START) as destination address by default; and it can be modified by sending publish set command.
- RC\_KEY\_4\_OFF uses ele\_adr\_primary as source address and OxCOO1(NLC\_DICMP\_GROUP\_ADDR\_START) as destination address by default; and it can be modified by sending publish set command.

See dicmp\_switch\_send\_publish\_command()'s "if branch" below for details:

```
void dicmp_switch_send_publish_command(u32 ele_offset, bool4 onoff, u32 select_pub_model_key)
{
#if NLCP_BSS_EN
    if(3 == ele_offset){
        u32 ele_offset_scene = 0;
        if(onoff){ // RC_KEY_4_ON
            ele_offset_scene = 0;
            u16 pub_addr = dicmp_get_publish_addr(ele_offset_scene, SIG_MD_SCENE_C, 0);
            sw_tx_src_addr_offset = ele_offset_scene;
            access_cmd_scene_recall(pub_addr, 0, 1, 0, 0);
                    // RC_KEY_4_OFF
        }else{
            ele_offset_scene = 1;
            u16 pub_addr = dicmp_get_publish_addr(ele_offset_scene, SIG_MD_SCENE_C, 0);
            sw_tx_src_addr_offset = ele_offset_scene;
            access_cmd_scene_recall(pub_addr, 0, 2, 0, 0);
        }
    }
    . . . . . . . . . . .
}
```

#### 29.5.4 Test Steps

#### 29.5.4.1 SDK Settings

• Firmware SDK using 825x\_switch project, set LIGHT\_TYPE\_SEL to LIGHT\_TYPE\_NLC\_CTRL\_CLIENT.

#### Note:

In this case, the configuration related to model enablement uses the configuration information in nlc\_ctrl\_client\_model\_config.h instead of the configuration information in mesh\_config.h.

#### 29.5.4.2 Add to Network

• To add the switch nodes to network, you can refer to section 12.5 Switch Engineering Long Press Handling Logic.

#### 29.5.4.3 Button Test

• Pressing RC\_KEY\_4\_ON will send SCENE\_STORE\_NOACK, and pressing RC\_KEY\_4\_OFF will send SCENE\_RECALL\_NOACK.

For both RC\_KEY\_4\_ON and RC\_KEY\_4\_OFF, the source address is primary address, the default destination address is Living room(0xC000), and the scene ID is 0x0001.

• You can check whether the command was sent successfully by capturing the packet, as shown below:

Low Energy Overview Message Log					4 ⊫ <b>×</b>	Details	ė 🗙
rotocol: Single 👻 🗚 layers 🗧 🛹 🖙 🧆 🖗 🖗 👚 📎 🦻 🖂 🧊 🖛 🛛 3 items displayed			A	' 🔍 🌺 🔹 Sear	ch •   <sup>®</sup> ⊉	🛛 All fields 📑 Show in overview	Display 👻 📑
Item	 Payload Length $\sim$	Time v	PDU Type 🛛 🗸	RF Channel I $ \sim $	Comp ~	Name	Value
* 🐉 Connectable + Scannable Undirected ("DICMP" A4:C1:38:F1:2B:6D, 10 Scanners, 13.1 s)	26 bytes	0.083 662 6	ADV_IND	37 (adv)	Telink Semi	Application Key Identifier	0x0D
B S Mesh Scene Recal Unacknowledged (Scene Number=1)	31 bytes	7.971 243 7	ADV_NONCONN	37 (adv)		PDU	91 4D B3 48 69 B6 78 0F F
Hesh Scene Recal Unacknowledged (Scene Number=2)	31 bytes	9.570 341 3	ADV_NONCONN	37 (adv)		NetMIC	0x6977939F
						🗉 👔 Mesh Application Information	
							Scene Roal Unacknowled 2 2 0x66780FFA V
						<	>
						Data	9 <b>x</b>
					>	Data type: Mesh Access Message 0 1 2 3 4 5 0x0000: 82 43 02 00 02 B6 0x0009:	- Search - 6 7 8 012345678 78 0F FA .Cx
ming							9 <b>×</b>



Configure the destination address of the key by sending the publish set command. see Publish\_adress Configuration Methods for the description of publish.

The following is an example of the publish address of the scene recall command with RC\_KEY\_4\_ON and RC\_KEY\_4\_OFF configured:

Method 1. The INI commands actually tested by the host computer are referenced below:



CMD-cfg\_pub\_set\_sig = e8 ff 00 00 00 00 00 00 02 00 03 02 00 02 c0 00 00 ff 00 15 05 12

This command sets the publish address of scene client model (0x1205) to Master bedroom (0xC002) on element address (0x0002) in the remote control.

Method 2. The App was tested as follows:

10:2	7		al 🗢 🖿
<		Device Setti	ng
C	ONTROL		SETTINGS
eleAdr: 0x	0002		
model: 0x	1205		
pubAdr: 0x	C002		AddressList
			SEND
eleAdr: 0x	0003		
model: 0x	1001		
pubAdr: 0x	C001		AddressList
			SEND
eleAdr: 0x	0004		
model: 0x	1001		
pubAdr: 0x	C002		AddressList
			SEND
eleAdr: 0x	0005		
model: 0x	1001		
pubAdr: 0x	C003	device connecte	AddressList

Figure 29.5: app pub set

model: Because you want to configure the publish address of the scene recall command, fill in 0x1205 (Scene Recall Client Model) here.

pubAdr: Set it as the destination address of the command sent by this button, here, for example, scene 1 of the lamp belonging to the Master bedroom is modified to Recall, so it is set to 0xC002.

## 29.6 BLCNLCP

#### 29.6.1 Function Description

Basic Lightness Controller NLC Profile (BLC) 1.0 - represents a luminaire with an integrated controller.

The corresponding spec is "BLCNLCP\_v1.0.pdf", for detailed function introduction, please refer to this document.

BSSNLCP: Define a light type device, the device not only supports common switch, brightness, colour temperature and other controls, but also supports light control model. when light control mode is off, it is an ordinary light, the state of the light is controlled by commands such as generic onoff, lightness set, etc. When light control mode is enabled, the light control server model corresponding to the light node can directly receive the status from the sensor, such as OCCUPANCY sensor(PRESENCE\_DETECTED), Ambient light sensor, etc., and then do the corresponding control on the status of the light node itself. Commonly used application scenarios, such as the light in the stairway, when detecting someone moving, it will automatically light up for a period of time, and then automatically fade to the standby brightness state, the standby brightness value is customisable, it can be 0, or it can be a relatively low brightness value. , conducto

#### 29.6.2 Hardware Introduction

Hardware uses 825x dongle.

#### 29.6.3 Test Steps

Note that the light control server model also supports the generic onoff model, so when a generic onoff message is received, the light control model is required to occupy an element independently in order to distinguish whether it is controlling the brightness on/off state or light control onoff state. see "Table 6.186: Light LC Setup Server elements and states" in MshMBT\_v1.0.pdf for details.

After the light control model is independent as an element, you need to pay attention to the configuration of the element address in the parameter area for the group add command for the light control model. the light control model is on the last element. For color temperature lamp, it is equal to primary address + 2, and for HSL lamp, it is primary address + 3. The same rule for setting LC mode onoff.

#### 29.6.3.1 SDK Settings

Firmware SDK use 825x\_mesh project, set NLCP\_BLC\_EN to 1.

- (1) When the node is powered on, the light control mode is off by default, and the method of controlling the brightness and other states of the light is the same as the operation without the light control function.
- (2) App Sends the Light LC mode set enable command to the node or through an INI command:

light\_lc\_mode\_set =e8 ff 00 00 00 00 00 00 02 00 82 92 01

Note:

The destination address is not primary address (node address), but primary address +(ELE\_CNT\_EVERY\_LIGHT - 1).

(3) App sends LIGHT\_LC\_ONOFF\_SET on command to the node, or via an INI command:

light\_lc\_onoff\_set =e8 ff 00 00 00 00 00 00 02 00 82 9a 01 00

#### Note:

The destination address is not primary address (node address), but primary address +(ELE\_CNT\_EVERY\_LIGHT - 1).

The nodes change brightness according to the following curve.

LuxLevel/Lightness			Light LC Time Fade On	Light LC Time Run	Light LC Time Fade	Light LC Time Prolong	Light LC Time Fade Standby		
LuxLevel/Lightness On LuxLevel/Lightness Prolong							Auto		
LuxLevel/Lightness Standby_			ř						Time
State	Off	Standby	Fade On	Run	Fade	Prolong	Fade Standby Auto	Standby	
Timer									_

Figure 6.4: Operation of a Light Lightness Controller

#### Figure 29.6: Operation\_of\_a\_Light\_Lightness\_Controller

The time parameters and luminance values of each step of the curve are defined by the following macros:

#define	LC_PROP_VAL_LightnessOn	(LIGHTNESS_	MAX)
#define	LC_PROP_VAL_LightnessProlong	((LIGHTNESS	_MAX + 1) / 4)
#define	LC_PROP_VAL_LightnessStandby	((LIGHTNESS	_MAX + 1) / 20
••••			
#define	LC_PROP_VAL_TimeFade	(2*1000)	// unit: ms
#define	LC_PROP_VAL_TimeFadeOn	(2*1000)	// unit: ms
#define	LC_PROP_VAL_TimeFadeStandbyAuto	(3*1000)	// unit: ms
#define	LC_PROP_VAL_TimeProlong	(4*1000)	// unit: ms
#define	LC_PROP_VAL_TimeRun	(5*1000)	// unit: ms

(4) Wait for step 3 to enter the standby state, and then send the INI command of OCCUPANCY sensor(PRESENCE\_DETECTED) status to the node, and the node will also change its brightness according to the curve in step 3.



sensor\_occupancy\_64 = e8 ff 00 00 00 00 00 00 02 00 52 01 42 00 64

(5) Wait for step 4 to enter standby state, then send the sensor status command to the lamp node via the sensor node introduced by "ocssnlcp", and the lamp node will also change its brightness according to the curve in step 3.

When the destination address of the sensor status message is a group address, you also need to subscribe to the group address for the generic onoff model or light control model on LC model element, the INI command example is as follows:

light\_lc\_model\_sub\_set = e8 ff 00 00 00 00 02 01 02 00 80 1b 04 00 01 c0 0f 13
or
light\_g\_onoff\_sub\_set = e8 ff 00 00 00 00 02 01 02 00 80 1b 04 00 01 c0 00 10

## 29.7 ocssnlcp

#### **29.7.1 Function Description**

Occupancy Sensor NLC Profile (OCS) 1.0 - represents an occupancy sensor

The corresponding spec is "OCSNLCP\_v1.0.pdf", please refer to this document for detailed function description.

When a person is detected approaching, the state of the sensor changes, which triggers the sensor to publish sensor status, which contains the current status of whether there is a person or not.

#### 29.7.2 Test Steps

#### 29.7.2.1 SDK Settings

Firmware SDK using 825x\_mesh\_project

- Set LIGHT\_TYPE\_SEL to LIGHT\_TYPE\_NLC\_SENSOR.
- NLC\_SENSOR\_TYPE\_SEL Select NLCP\_TYPE\_OCS
- SENSOR\_PROP\_ID Select PROP\_ID\_PRESENCE\_DETECTED

#elif (NLC\_SENSOR\_TYPE\_SEL == NLCP\_TYPE\_OCS)
#define SENSOR\_PROP\_ID PROP\_ID\_PRESENCE\_DETECTED

Then compile it to get the firmware.

#### 29.7.2.2 Function

• Publish set using the ini command



CMD-cfg\_pub\_set\_sig = e8 ff 00 00 00 00 02 00 02 00 03 02 00 ff ff 00 00 FF 00 00 00 11

For the description of the above commands, you can refer to Publish\_adress Configuration Methods, and note that you have to modify the destination address, element address and publish address inside the cfg\_pub\_set\_sig command according to the unicast address of the current test node.

The sensor\_measure\_proc() function polls the status value detected by the sensor to see if the trigger condition is reached, if it is reached, the value of pub\_flag will be 1, and then it will see if there is a publish address configured, if there is one, then it will send a sensor status message, see the processing of sensor\_measure\_proc() for details.

```
u32 sensor_measure_proc()
{
    . . .
    u32 measure_val = 0;
    memcpy(&measure_val, p_sensor_data->p_raw, min2(sizeof(measure_val),
→ p_sensor_data->len_raw));
    if(sensor_measure_quantity < measure_val){</pre>
        if((measure_val - sensor_measure_quantity) > p_cadence->cadence_unit.delta_down){
            pub_flag = 1;
        }
    }
    else{
        if((sensor_measure_quantity - measure_val) > p_cadence->cadence_unit.delta_up){
            pub_flag = 1;
        }
    }
    if(pub_flag){
        model_pub_check_set(ST_G_LEVEL_SET_PUB_NOW, (u8 *)&model_sig_sensor.sensor_srv[0].com,
   0);
    }
}
```

measure\_val: Previous state value (0/1)

sensor\_measure\_quantity: Current status value (0/1)

pub\_flag: Publish flag. it will be set to 1 when delta between the current state value and the previous one reaches the threshold.

- The change amount threshold can be configured by sending the Sensor Cadence Set, when not configured, p\_cadence->cadence\_unit.delta\_down/p\_cadence->cadence\_unit.delta\_up is 0.
- Since the current dongle board does not have a sensor peripheral, there are currently two ways for the demo sdk to change the sensor value:



Method 1: Go through the BDT to set the value of sensor\_measure\_quantity. For example, if you set 1 for the first time and 0 for the second time, the node will publish sensor status once. Or the node will publish sensor status once if it set 0 the first time and 1 the second time. In any case, the node publishes sensor status if the value is reversed from the previous time.

Method 2: Change the value of sensor\_measure\_quantity by pressing the key. In addition to the above "SDK Settings" in this section, open the UI\_KEYBOARD\_ENABLE, then compile SDK, then burn the compiled firmware to 8258\_dongle, then every time you press SW2, the value of sensor\_measure\_quantity will change once, and the node will publish status once, and the implementation code is as follows:

voic	d mesh_proc_keyboard (void)
{	
	<pre>if(KEY_SW2 == kb_event.keycode[0]){</pre>
	<pre>sensor_measure_quantity = !sensor_measure_quantity;</pre>
	}
}	

The following image shows the result of a packet capture of the publish sensor status:

Low Energy Overview Message Log H Spectrum 4 🖡					Details	9
ocol: Single 👻 All layers 🔸 🛹 🚥 🎰 💡 🖹 🕑 🔊 🖂 🎝 💽 🐡 🛛 3 items displayed			7 Q 🖄 -	Search 🔹 🔮	🛛 All fields 📑 Show in overview 🛛 Display 🕶 🚉	
tem # ↓ ↓	Sequence Nu ∨ 0x000287 0x000288	Source Address ∨ 0x0002 (Unicast) 0x0002 (Unicast)	Payload Length V 31 bytes 30 bytes 31 bytes		Name NetMC NetMC NetMC Network Network Network Network Network Network Network Network Network Detmaton Adress Detmaton Adress	Value 0x5853A261 0x0002 (Unicast) 0xFFFF (Al)
					Wesh Access Message      Yess Payload      OpCode      Gentrated Sensor Data      Format      Length      Property ID      W Ran Value      TransMic      C	Sensor Status B 1 Presence Detected 0x00 0xF11A6D70
					Data         Ota           Data type:         Mesh Access Message +           0         1         2         3         4         5         6           0x00000:         52         01         4D         00         00         F1         1A           0x00009:             0         0         F1         1A	q Search 7 8 012345678 6D 70 R.Mmp

Figure 29.7: pub ocs

Property ID: PROP\_ID\_PRESENCE\_DETECTED(0x004D), it means that the sensor at this moment is the occupancy sensor.

# 29.8 ALSNLCP

## 29.8.1 Function Description

Ambient Light Sensor NLC Profile (ALS) 1.0 - represents an ambient light level sensor. (Ambient light sensor)



The corresponding spec is "ALSNLCP\_v1.0.pdf", for detailed function description, please refer to this document.

When the detected change in ambient illumination is greater than the set threshold, the sensor will be triggered to publish sensor status, and the sensor status contains the current ambient light level.

#### 29.8.2 Test Steps

#### 29.8.2.1 SDK Settings

The firmware SDK uses 825x\_mesh\_project, sets LIGHT\_TYPE\_SEL to LIGHT\_TYPE\_NLC\_SENSOR, enables it, and compiles the firmware.

#### 29.8.2.2 Function Test ALSNLCP

• Publish set using the ini command

CMD-cfg\_pub\_set\_sig = e8 ff 00 00 00 00 02 00 02 00 03 02 00 ff ff 00 00 FF 00 00 00 11

For a description of the above commands you can refer to Publish\_adress Configuration Methods.

• Use the ini command to send Sensor Cadence Set to set the threshold for the amount of change in ambient light level.

e8 ff 00 00 00 00 02 00 02 00 55 4E 00 02 01 00 00 01 00 00 0C F0 FF 00 10 00 00

The structure corresponding to the command is as follows:

```
typedef struct{
    u16 prop_id;
    sensor_cadence_t cadence;
    sensor_setting_par_t setting;
}sensor_states_t;
```

prop\_id: sensor property id, the setting here is PROP\_ID\_PRESENT\_AMBIENT\_LIGHT\_LEVEL(0x004E)

trig\_type: State trigger type. 0: Change value. 1: Percentage.

delta\_down: The minimum change value that triggers a node to publish sensor status when the ambient light level decrease. The publish is triggered when the illumination reduction reaches the set threshold, which is 0x000001, but can be set to any other value.

delta\_up: The minimum change value that triggers a node to publish sensor status when the ambient light level increase. The publish is triggered when the amount of illumination reaches a set threshold, which is 0x000001, but can be set to any other value.
- 🗉 Telink
  - Set the value of sensor\_measure\_quantity, i.e. the current illuminance, through BDT. For example, if the current sensor\_measure\_quantity value is 0, and it is set to 10, then the change in illumination is greater than the set threshold, then the node will trigger publish status once.

The following is the result of the publish sensor status packet capture.



#### Figure 29.8: pub als

Property ID: PROP\_ID\_PRESENT\_AMBIENT\_LIGHT\_LEVEL(0x004E), indicates that the sensor status contains the light level.

## **29.9 ENMNLCP**

#### 29.9.1 Function Description

Energy Monitor NLC Profile (ENM) 1.0 – represents a sensor reporting energy consumption

The corresponding spec is "ENMNLCP\_v1.0.pdf", for detailed function description, please refer to the energy consumption monitor part in this document.

#### 29.9.2 Test Steps

#### 29.9.2.1 SDK Settings

Firmware SDK using 825x\_mesh\_project

- Set LIGHT\_TYPE\_SEL to LIGHT\_TYPE\_NLC\_SENSOR.
- For NLC\_SENSOR\_TYPE\_SEL, select NLCP\_TYPE\_ENM.

Then compile it to get the firmware.

#### 29.9.2.2 Function Test

• Publish set using the ini command.

CMD-cfg\_pub\_set\_sig = e8 ff 00 00 00 00 02 00 02 00 03 02 00 ff ff 00 00 FF 00 00 00 11

For a description of the above commands you can refer to Publish\_adress Configuration Methods.

- The amount of change threshold can be configured by sending the Sensor Cadence Set, which defaults to 0 when not configured, same goes for setting delta\_down. delta\_up. How to do, please refer to Function Test ALSNLCP.
- Go through the BDT to set the value of sensor\_measure\_quantity, which is the current energy value. For example, the value of sensor\_measure\_quantity is 0, set it to 10, then the node will publish status.

The following is the result of the publish sensor status packet capture.



Figure 29.9: pub enm

Property ID: PROP\_ID\_PRECISE\_TOTAL\_DEVICE\_ENERGY\_USE(0x0072) indicates that the sensor status contains energy detection.

## **30 Ellisys Decrypts Mesh Packets**

## **30.1 Click Record to Grab the Packet**

## 30.2 Fill in Mesh Information for Decryption

 Click on the mesh security option in the menu bar view drop down box and fill in the mesh key information in big endian format (network key, appkey key, and IV index. If you want to see messages using device keys such as the configuration model, such as the publish set command, you also need to fill in the device key of the device).

otocol Hide:	'elcome	Message Log	instant Spectrum 🛛 🖓 🛞 35 items k	ept, 322 filtered	7 Q 🂁 +	⊈ کے Search	Mesh Security	Device Keys Application Keys	]		5
Item	T	Status V OK OK OK OK OK OK OK	Payload ~	Time         I         I           0.384 262 625         3.735 103 750         3.740 676 125         5.000 447 250           6.793 368 875         6.800 713 125         9.884 462 625         9.884 042 625	Sequence Nu 0x0000AA 0x00011F 0x0000AB 0x000120 0x000120 0x0000AC 0x0000421	Source > 0x0001 (Unica 0x0002 (Unica 0x0001 (Unica 0x0002 (Unica 0x0001 (Unica 0x0001 (Unica	NID 0x04	Key D16F6FAE:SD49E959:96ADA303:	IV Index 0DE07871 0x0000000	Emption Key CD81836C-409408FC:3FE08818:8185EF79	Privacy Key CE3A3E47:FC3F3
	Mesh Time Status ("AL Seconds=2 Mesh Encrypted Network Traffic Mesh Time Status (TAI Seconds=2 Non-Connectable Undirected Adv Mesh Time Status (TAI Seconds=2	ок ок ок ок	23 bytes (55 6F AD 38 C1 A4 3E 2	10, 107 410 125         27, 461 426 375         39, 595 747 500         55,086 995 250         71,363 909 000	0x000122 0x000123 0x000124	0x0002 (Unica 0x0002 (Unica	٢				_
	Mesh Unprovisioned Device Beaco Mesh Encrypted Network Traffic Mesh Time Status (TAI Seconds=2 Mesh Time Status (TAI Seconds=2 Mesh Time Status (TAI Seconds=2 Mesh Time Status (TAI Seconds=2	OK Warning OK OK OK		80.439 865 375 92.117 962 500 103.735 304 625 134.706 413 875 164.425 848 625 179.587 345 500	0x000125 0x000126 0x000127	0x0002 (Unica 0x0002 (Unica 0x0002 (Unica	Time 27.461 426 3 92.117 962 5 179.587 345 274.604 418 352.389 751	Missing Key           175         Network Key (NID 0x7D)           100         Network Key (NID 0x10)           500         Network Key (NID 0x2D)           525         Network Key (NID 0x2E)			
stant Tin	Mesh Time Status (TAI Seconds=2	ок		195.865 694 000	0x000128	0x0002 (Unica >	332.309 731	023 Network ney (ND 0x22)			

Figure 30.1: Mesh security

 If you don't know the key, you can use BDT tool to get the key from the node, mesh\_key global variables include: device key, network key and app key; iv index is in iv\_idx\_st. The structure of different sdk versions may change, The location of each member field of global variable "mesh\_key" and "iv\_idx\_st" can be found in the code. Take V3.3.3 as an example.

BDT connect to 1:us	b#vid_248a&	pid_8266#68	x341d0d86&0&4#{2	28d78fad-5a12-11d1-ae5b-0000f803a8c2} - 🗆 🗙
Device File View Tool	Help			
I <u>8</u> 258 ▼ <sup>1</sup> √ EVK ▼	Setting (	🖲 Erase 👤	Download + Activat	te 🕩 Run II Bause 🍽 Step 🔍 PC 🥵 Single step 🔹 🥂 Reset 😨 manual mode 🔹 🚽 Olear
b0 10	b0	10	2 SWS	602 06 ■ Stall 602 88 > Start
Ŧ	Download			값입 Tdebug 표 Log windows
Variable Name	Addr	Len	Value ^	^
mesh_key	44308	917		[11:46:08]:
mesh_key_addr	430e4	4	00030384	1032 EVK: SWITE OK!
mesh_md_cfg_s_addr	430a8	4	000310ec	[11:46:26]: device key
mesh_md_g_onoff_lev	43100	4	000338b0	917 bytes have finished!
mesh_md_g_power_o	43080	4	00071858	04430 ba 44 00 f6 ea 1c 7f 6c a6 4d e5 5d b5 7c 09 e5
mesh_md_health_add	430b4	4	000324c8	044318: 32 80 93 aa b4 c7 d4 de t2 tc 09 lc 26 30 3d 47
mesh_md_light_ctl_ad	430bc	4	000368b0	044328: 18 a9 e2 e7 0e 57 da 62 65 6d d3 8a 54 c9 2a 4f
mesh_md_lightness_a	4309c	4	0003545c	044338: ba 41 93 19 59 bd au 16 04 9d d8 3b 19 ac 6b ce 044348: 75 53 9f d5 25 50 ee 85 1d 8f 7b 09 bd 8b 32 f0
mesh_md_mesh_ota_	43140	4	00074738	044358: 3a d7 4c eb 39 6b 6b 9c 2c 1a d4 3b 87 c9 e9 ab
mesh_md_vd_light_ad	430a0	4	00070178	044368: 10 52 1a a1 d7 ba 41 4a 68 df 1c 85 55 f6 66 38
mesh_misc_addr	430b0	4	0003d520	044378: 3d 9e 25 13 58 27 29 14 a4 02 d5 da 9e 17 24 4b 044388: b9 c1 89 89 63 85 24 2e c2 a1 af 1b 15 76 5a a0
mesh_need_random_c	44060	1	0000000	044398: 8a 16 7a ff 2f 3f 5f fb b8 57 73 1c e0 90 61 97
mesh_node_out_oob_	43fa0	4	0000000	0443a8: 75 01 db f3 11 27 dc 99 c9 ec 93 b8 b5 59 fd 8a
mesh_provision_end_t	43fb4	4	00000000	0443b8: 62 0c 00 00 01 00 00 00 60 96 47 71 73 4f bd 76
mesh_provision_mag_	430ac	4	0003a073	0443d8: 00 00 00 00 00 00 00 00 00 00 00 00 00
mesh_rsp_random_de	4407c	2	0000000	0443e8: 00 00 00 00 00 00 00 00 00 00 00 00 00
mesh_rsp_random_de	4406c	4	0000000	0443f8: 00 00 00 00 00 00 00 00 00 00 00 00 00
mesh_rx_seg_par	46c14	34		
			~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	

#### Figure 30.2: Get mesh key

BDT connect to 1:us	b#vid_248	a&pid_826a#	6&81fb60d&0&4	#{2	28d78fad-5a12-11d1-ae5b-0000f803a8c2}	
evice File View Tool I	Help					
🗒 885 🔹 🍾 EVK 🔹 👘	Setting	🖉 Erase 👤	Download + Act	ivate	ite I⊫ Ryn II Pause I⊯ Step Q, PC 📌 Single step 🔹 🤻 Reset 😨 manual mode 🕶	🚽 Clear
🖞 Unlock 🛛 🗂 Cac <u>h</u> e						
0 10	b0	10	2 SW	s	602 06 Stall 602 88	Start
Ŧ	Download				했 Tdebug 표 Log windows	
Variable Name	Addr	Len	Value	^		
gatt_adv_send_flag	42d8b	1	0000001		[13:45:38]:	
gatt_adv_tick.17921	43d60	4	0337f3d0		042dd8: 01 00 00 00 01 00 00 00 00 00 00 00 a7 00	00 00
nb_pub_100ms	43c78	4	00000000		042de8: 01 00 00 00	00 00
hb_sts_change	43c80	1	00000000		Total Time: 20 ms	
nb_sub_100ms	43c7c	4	0000000			
nci_eventMask	42f0c	4	00000010		iv index	
nci_le_eventMask	4502c	4	0000007			
nci_tlk_module_event	44e44	4	00000000			
neartbeat_en	42d5c	1	0000001			
dx_num.14483	43c8c	1	0000000			
rq_stk	42f5c	384				
v_idx_st	42dd8	20				
v_update_by_sno.131	44d8f	1	00000000			
key_bind_all_ele_en	44356	1	0000000	~	, <	
k device: ok	File Path:	C:\Users\ZB	\Desktop\SDK\teli	ink	k sig mesh src\ble lt mesh\8258 mesh\8258 mesh.bin Version :	5.7.2

#### Figure 30.3: Get mesh iv



#### Note:

The iv index in BDT is the little endianness of the display, in ellisys need to fill in the big endianness, such as the above figure is to fill in "00000001" or "01". if SIG Mesh SDK version is later than V4.1.0.0(include), the iv index in BDT is the big endianness of the display.

• Perform mesh decryption and view it.

Energy Overview HCI Injection Overview HCI Injection Overview (Secondary) HCI Injection Overview (Tertiary)	Message Log			4 Þ 🗙	Detais	
ol: Single 🗸 All layers 🔸 🏕 🕯 🥧 🖗 👚 😒 🦻 🦂 🖉 🧑 🔶 76 items kept, 23 filtered		Y	Q 👌 - 1	Search 🛛 - 🛙 🔮	🛛 All fields 📑 Show in overview	Display - 😭 Search
r: Item × != "Mesh Unprovisioned Device Beacon" ×, "Mesh Secure Network Beacon" ×		3			Name	Value
I	₹ 4 ∨	Source Address ∨	Time 🗸	Applicati ~ ^	Application Key Flag	Yes
A mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1002)		0x0006 (Unicast)		Mesh	<ul> <li>Application Key Identifier</li> </ul>	0x0D
Mesh Config Model App Bind (Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1004)		0x0001 (Unicast)	0.049 31	Mesh	PDU ANNUC	10 94 4F 5D 9B 3D 08 00
Mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1004)		0x0006 (Unicast)	0.161 43	Mesh	<ul> <li>Netruc</li> </ul>	0X0AD022D7
Mesh Config Model App Bind (Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1006)		0x0001 (Unicast)	0.016 33	Mesh	🗏 🌍 Mesh Network Information	
Mesh Config Model App Bind (Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1007)		0x0001 (Unicast)	0.234 42	Mesh	Network	
Mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1006)		0x0006 (Unicast)	0.015 75	Mesh	🗉 🔧 Subnet	
Mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1007)		0x0006 (Unicast)	0.148 01	Mesh	🗄 😚 Devices	
Mesh Config Model App Bind (Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1300)		0x0001 (Unicast)	0.059 01	Mesh	A March Natwork DDU	
Mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1300)		0x0006 (Unicast)	0.185 52	Mesh	S Mesh Network PD0	
Mesh Config Model App Bind (Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1301)		0x0001 (Unicast)	0.016 80	Mesh	🔷 IVI	0x1
Mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1301)		0x0006 (Unicast)	0.145 76	Mesh	NID	0x41
Mesh Config Model App Bind (Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1303)		0x0001 (Unicast)	0.027 90	Mesh	I CTL	Access Message
Mesh Config Model App Bind (Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1304)		0x0001 (Unicast)	0.276 20	Mesh		10
Mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1303)		0x0006 (Unicast)	0.005 77	Mesh	Sequence Number	0x0003B7
Mesh Config Model App Bind (Element Address=0x0006.u, App Key Index=0x000, Vendor Model ID=0x0000)		0x0001 (Unicast)	0.186 22	Mesh	Source Address	0x0001 (Unicast)
Mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, SIG Model ID=0x1304)		0x0006 (Unicast)	0.059 09	Mesh	Destination Address	0x0006 (Unicast)
Mesh Config Model App Status (Status=Success, Element Address=0x0006.u, App Key Index=0x000, Vendor Model ID=0x0000)		0x0006 (Unicast)	0.094 00	Mesh	Lower Transport PDU	
Mesh Config Model App Bind (Element Address=0x0007.u, App Key Index=0x000, SIG Model ID=0x1002)		0x0001 (Unicast)	0.020 56	Mesh	NetMIC	0x8AB822B7
Mesh Config Model App Bind (Element Address=0x0007.u, App Key Index=0x000, SIG Model ID=0x1306)		0x0001 (Unicast)	0.249 88	Mesh	🗏 🗉 🚺 Mesh Application Information	
Mesh Config Model App Status (Status=Success, Element Address=0x0007.u, App Key Index=0x000, SIG Model ID=0x1002)		0x0006 (Unicast)	0.096 44	Mesh	E de Nature	
Mesh Config Model App Status (Status=Success, Element Address=0x0007.u, App Key Index=0x000, SIG Model ID=0x1306)		0x0006 (Unicast)	0.070 39	Mesh	Application Key	
Mesh Light Lightness Get		0x0001 (Unicast)	3.318 25	Mesh	Adresses	
Mesh Light Lightness Status (Present Lightness=65'535)		0x0006 (Unicast)	0.220 05	Mesh	Source Address	0x0001 (Unicast)
Mesh Generic OnOff Set (On Off=Off)		0x0001 (Unicast)	2.082 17	Mesh	<ul> <li>Destination Address</li> </ul>	0x0006 (Unicast)
Mesh Generic OnOff Status (Present On Off=On, Number of Steps=10, Step Resolution=100 miliseconds)		0x0006 (Unicast)	0.007 71	Mesh	🗟 🔧 Devices	
Mesh Generic OnOff Set (On Off=On)		0x0001 (Unicast)	0.982 02	Mesh	Source	A4:C1:38:30:85:82
Mesh Generic OnOff Status (Present On Off=On, Number of Steps=10, Step Resolution=100 millseconds)		0x0006 (Unicast)	0.078 96	Mesh	Destination	A4:C1:38:30:85:82
Mesh Generic OnOff Set (On Off=Off)		0x0001 (Unicast)	0.954 38	Mesh	R March Accors Massage	
Mesh Generic OnOff Status (Present On Off=On, Number of Steps=10, Step Resolution=100 miliseconds)		0x0006 (Unicast)	0.041 20	Mesh	w mesh Access Message	
Mesh Encrypted Network Traffic NID=0x52 (x 1, 0 s)			0.198 41	Mesh	Access Payload	
Mesh Generic OnOff Set (On Off=On)		0x0001 (Unicast)	0.846 62	Mesh	OpCode	Generic OnOff Set
Mesh Generic OnOff Status (Present On Off=Off, Number of Steps=10, Step Resolution=100 miliseconds)		0x0006 (Unicast)	0.109 21	Mesh	On Off	Off
				~	Transaction Identifier	1



An explanation of steps 1, 2, and 3 in the figure:

- 1) Click the "mesh" button for mesh decryption;
- You can use the mouse to drag a certain item from "All Fields" on the right to the main interface for display;
- 3) The above figure is an example of step 2.



## **30.3 Other Methods to Get the Key**

#### 30.3.1 Provision UART Log of provision flow Via Firmware

10:16:23.500]	LIB]:(sdk)mesh tx NoAck,op:0x4e82(LIGHINESS_STATUS),src:0x53b7,dst:0xffff,sno:0x000081 par_len:2 par:ff ff	
10:16:29.551]	LIB]:(sdk)mesh ble connect cb	
10:16:30.900]	LIB]:(sdk)rcv provision invite 00 00	
10:16:30.902]	LIB]:(sdk)send capa cmd 01 02 00 01 00 00 00 00 00 00 00	
10:16:31.749]	LIB];(sdk)rcv start cmd 02 00 00 00 00	
10:16:31.754]	LIB]: (sdk)rcv pubkev cmd	
10:16:31.754	3 44 93 cl 19 55 eb fc 6c c3 77 fa a9 e2 25 bf 4e 36 8f f4 50 8a d6 9f 2f 9f f8 d9 33 43 76 fb	
10:16:31.756	c de bd 61 1e 6d a5 50  0d 86 fe b8 00 f4 be da  3d 9d 71 c8 ae 34 78 17  35 f9 66 c9 fb e6 30 2a	
10:16:31.765]		
10:16:31.771]	LTB]:(sdk)send pubkey cmd	
10:16:31.771]	3 9d df 25 57 20 75 81 80 85 28 ce 81 da be 03 f8 98 48 9d 70 fd 69 f6 a2 9b 4e 55 68 bc 96 01	
10:16:31.771]	1 43 ff aa 03 c5 99 b5 a5 4f 4e 4b 77 94 6b 2b 75 09 11 f8 a6 eb 44 97 a1 1c 0e 2a b7 ff 7a 04	
10:16:31.771		
10:16:31.921	- LTB:(sdk)rev comfirm end 05 09 d9 bc 6c 77 38 9b 3f cc 29 2d 76 4d eb ca 3a	
10:16:34.019]	LTB]: (sdk)send comfirm cmd 05 4a c5 2e 52 ec 99 f4 ce bf 3e 69 06 5e 44 68 30	
10:16:34.082]	LTB1: (sdk)rcv random cmd 06 69 02 40 92 99 db d7 c7 9f d1 b8 86 1e a7 95 af	
10:16:34.0921	ITB]: (sdk)send random cmd 06 97 e2 ac 2b ef f9 6a e5 05 2f e6 a5 35 68 fc 9c	
10:16:34, 157]	LIB]: (sdk)rcy provision data cmd	
10:16:34.159]	7 da 91 0e 48 cd 2e 66 dl e0 5c 40 98 43 fe d5 a2 0d 57 55 30 77 fe ae 3a f4 59 db 0d 0c 0f 09	
10:16:34, 164]		
10:16:34.171	LIB]: (provision) device key k7 2b 2b 2e fd 05 d7 c6 77 cc 8b bf fb 4e 33 b0)	
10:16:34, 180]	LTB]: (sdk)provision net infn is f5 43 f2 2c 83 a3 14 al ce aa a2 e0 fc 9f ff 10 00 00 00 00 00 00 00 93 00	
10:16:34.229	LIB]: (sdk)provision suc!	
10:16:38.247]	LIB]; (sdk)mesh ble disconnect cb	
10.16.20 0101		
10.10.30.0101	LID::/sdk/mesh_bie_connect_cp	
10:16:40.107	LLD]: (SOK)/MEST_DIE_CONTRECT_CD LLD]: (SOK)/MEST_DIE_CONTRECT_CD LLD]: (SU MANEtabor to to Beacom with GATT, IV index step0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
10:16:40.107] 10:16:40.112]	LIB (1500, man, 542, competence) with GAT, IV index step; 00 00 00 00 00 00 00 00 00 00 00 00 00	
10:16:40.107] 10:16:40.112] 10:16:40.118]	LD): dax.mmesc.ple_commet_cc LIB): ('u_update) app tx beacon: with GATT, IV index step0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
10:16:40.107] 10:16:40.112] 10:16:40.118] 10:16:40.130]	LD: 16 (av., man, 5.e_commet.com LD: (15, (14, matrix) pp, tabeacom; thit 6.477, 17 (index strato): 0 00 00 00 00 00 00 00 00 00 00 00 00	
10:16:40.107] 10:16:40.112] 10:16:40.112] 10:16:40.118] 10:16:40.252]	LD: (sak/mash.ple_commet_co LD: (u_update) app tx beacon: with GATT, IV index step0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
10:16:40.107] 10:16:40.112] 10:16:40.118] 10:16:40.130] 10:16:40.252] 10:16:40.252]	Llb: (sku/mash.psc_com/set_co Llb: (su_pdats) pp tz beacom wih GATT, IV index step0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
10:16:40.107] 10:16:40.112] 10:16:40.113] 10:16:40.130] 10:16:40.252] 10:16:40.252] 10:16:40.267] 10:16:40.324]	LB): Gak/memory.org LB): (upudate) par tx beacom with GATT, IV index step0: 00 00 00 00 00 00 00 00 00 00 00 00 0	
10:16:40.107] 10:16:40.112] 10:16:40.113] 10:16:40.130] 10:16:40.252] 10:16:40.267] 10:16:40.323] 10:16:40.333]	LIB: (sk_make_heat_ole_context_conduct	: 00
10:16:40.107] 10:16:40.112] 10:16:40.113] 10:16:40.130] 10:16:40.252] 10:16:40.324] 10:16:40.333] 10:16:41.679]	<pre>LIB: [scm/medit_low_control=1_contto=1_control=1_control=1_control=1_control=1_control=1_co</pre>	: 00
10:16:40.107] 10:16:40.112] 10:16:40.130] 10:16:40.252] 10:16:40.252] 10:16:40.324] 10:16:41.679] 10:16:41.690]	LID: (skx/mach.ple_commet_co LID: (skx/mach.ple_commet_commet_commet_co LID: (skx/mach.ple_commet_co	: 00
10:16:40.107] 10:16:40.112] 10:16:40.130] 10:16:40.252] 10:16:40.252] 10:16:40.333] 10:16:41.679] 10:16:41.690] 10:16:41.750]	<pre>Ling: (sak/mean, bie_context_co- Ling: (sak/mean, bie_context_co- Ling); (sak/mean, bie_context_context_context_context_context_context_context_context_context_context_context_context_context_context_context_contex</pre>	: 00
10:16:40.107] 10:16:40.112] 10:16:40.113] 10:16:40.252] 10:16:40.252] 10:16:40.267] 10:16:40.333] 10:16:41.679] 10:16:41.690] 10:16:41.750] 10:16:41.766]	LlB: (sk_make.be_context.com LlB: (sk_make.be_context.com) (sk_make.be	: 00
10:16:40.107] 10:16:40.112] 10:16:40.113] 10:16:40.252] 10:16:40.267] 10:16:40.324] 10:16:41.679] 10:16:41.750] 10:16:41.865]	LIB: (sub_main_prot_be_compet_	: 00
10:16:40.107 10:16:40.112 10:16:40.113 10:16:40.130 10:16:40.257 10:16:40.267 10:16:40.333 10:16:41.679 10:16:41.750 10:16:41.750 10:16:41.853	<pre>Ling 1:get_ment to Mack, op:0x3e00(MDEXT, IV index respondence). do 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</pre>	: 00
$\begin{array}{c} 10.16.36, 101\\ 10.16:40, 102\\ 10.16:40, 112\\ 10.16:40, 130\\ 10.16:40, 130\\ 10.16:40, 252\\ 10.16:40, 257\\ 10.16:40, 254\\ 10.16:40, 333\\ 10.16:41, 679\\ 10.16:41, 766\\ 10.16:41, 766\\ 10.16:41, 865\\ 10.16:41, 938\\ 10.16:41, 938\\ \end{array}$	<pre>Link : Galvament Die_contret_cont</pre>	: 00
$\begin{array}{c} 10:16:40,107\\ 10:16:40,102\\ 10:16:40,102\\ 10:16:40,138\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:40,233\\ 10:16:40,333\\ 10:16:41,679\\ 10:16:41,750\\ 10:16:41,750\\ 10:16:41,885\\ 10:16:41,885\\ 10:16:41,983\\ 10:16:41,951\\ 10:16:41,951\\ \end{array}$	<pre>Link 1:sex/mean_bie_context.com</pre> UIII 1:sex/mean_bie_context.com app Link 1:sex/mean_bie_context.c	: 00
$\begin{array}{c} 10:16:40,107\\ 10:16:40,102\\ 10:16:40,118\\ 10:16:40,252\\ 10:16:40,267\\ 10:16:40,267\\ 10:16:40,267\\ 10:16:40,333\\ 10:16:41,670\\ 10:16:41,670\\ 10:16:41,760\\ 10:16:41,760\\ 10:16:41,883\\ 10:16:41,938\\ 10:16:41,938\\ 10:16:41,938\\ 10:16:41,938\\ 10:16:42,013\\ \end{array}$	<pre>Link : Galward has not a construct on the GATT, IV index step0: 00 00 00 00 00 00 00 00 00 00 00 00 0</pre>	: 00
$\begin{array}{c} 10:16:40, 107\\ 10:16:40, 112\\ 10:16:40, 113\\ 10:16:40, 113\\ 10:16:40, 252\\ 10:16:40, 252\\ 10:16:40, 252\\ 10:16:40, 252\\ 10:16:40, 324\\ 10:16:41, 679\\ 10:16:41, 679\\ 10:16:41, 676\\ 10:16:41, 760\\ 10:16:41, 760\\ 10:16:41, 983\\ 10:16:41, 983\\ 10:16:41, 981\\ 10:16:41, 981\\ 10:16:42, 013\\ 10:16:42, 013\\ 10:16:42, 024\\$	<pre>Link 1:sex/mean_big_context_co- link 1:sex/mean_big_context_co- big_context_co- link 1:sex/mean_big_context_co- big_context_co- link 1:sex/mean_big_context_co- big_context_co- link 1:sex/mean_big_context_co- big_conte</pre>	2 00
$\begin{array}{c} 10:16:40,107\\ 10:16:40,102\\ 10:16:40,113\\ 10:16:40,103\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:41,252\\ 10:16:41,670\\ 10:16:41,760\\ 10:16:41,760\\ 10:16:41,883\\ 10:16:41,961\\ 10:16:42,024\\ 10:16:42,013\\ 10:16:42,013\\ 10:16:42,2125\\ \end{array}$	<pre>Lik: (sk2, make, be_conset).com with GAT, IV index step0: 00 00 00 00 00 00 00 00 00 00 00 00 0</pre>	: 00
$\begin{array}{c} 10:16:40, 107\\ 10:16:40, 112\\ 10:16:40, 113\\ 10:16:40, 113\\ 10:16:40, 252\\ 10:16:40, 252\\ 10:16:40, 252\\ 10:16:40, 252\\ 10:16:40, 324\\ 10:16:41, 303\\ 10:16:41, 679\\ 10:16:41, 750\\ 10:16:41, 750\\ 10:16:41, 883\\ 10:16:41, 981\\ 10:16:41, 981\\ 10:16:41, 981\\ 10:16:41, 981\\ 10:16:42, 212\\ 10:16:42, 212\\ 10:16:42, 2137\\ 10:16:42, 2$	<pre>Link 1:sex/mean_big_cormetted_cormetted_cormetted_corm the darT, IV index step: 00 00 00 00 00 00 00 00 00 00 00 00 00</pre>	: 00
$\begin{array}{c} 0 & 11 & 11 & 10 & 110 \\ 10 & 116 & 40 & 1107 \\ 10 & 116 & 40 & 1102 \\ 10 & 116 & 40 & 1180 \\ 10 & 116 & 40 & 1260 \\ 10 & 116 & 40 & 2527 \\ 10 & 116 & 40 & 2527 \\ 10 & 116 & 40 & 2527 \\ 10 & 116 & 40 & 2527 \\ 10 & 116 & 40 & 2527 \\ 10 & 116 & 41 & 603 \\ 10 & 116 & 41 & 6781 \\ 10 & 116 & 41 & 6781 \\ 10 & 116 & 41 & 8651 \\ 10 & 116 & 41 & 8651 \\ 10 & 116 & 41 & 8651 \\ 10 & 116 & 41 & 9381 \\ 10 & 116 & 42 & 0131 \\ 10 & 116 & 42 & 0131 \\ 10 & 116 & 42 & 0131 \\ 10 & 116 & 42 & 20031 \end{array}$	<pre>Link 1 (stw.medn.ple_context.com Link 1 (stw.medn.tx 1 (stw.sci.stw.s</pre>	: 00
$\begin{array}{c} 10:16:40,102\\ 10:16:40,112\\ 10:16:40,113\\ 10:16:40,113\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:40,252\\ 10:16:40,324\\ 10:16:41,679\\ 10:16:41,750\\ 10:16:41,750\\ 10:16:41,865\\ 10:16:41,865\\ 10:16:41,865\\ 10:16:41,933\\ 10:16:41,933\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,223\\ 10:16:42,23\\ 10:16,23\\ 10:16;42,23\\ 10:16,23\\ 10:16;$	<pre>Link 1: Sum Amage Die_contrest.com</pre>	: 00
$\begin{array}{c} 10&11.4&1.40&10.7\\ 10&16&40&112\\ 110&16&40&112\\ 10&16&40&138\\ 10&16&40&138\\ 10&16&40&252\\ 10&16&40&252\\ 10&16&40&252\\ 10&16&40&324\\ 10&16&40&324\\ 10&16&41&679\\ 10&16&41&679\\ 10&16&41&679\\ 10&16&41&865\\ 10&16&41&865\\ 10&16&41&865\\ 10&16&41&865\\ 10&16&41&862\\ 10&16&41&862\\ 10&16&41&82\\ 10&16&42&82\\ 10&16&42&125\\ 10&16&42&23\\ 10&16&4&2&23\\ 10&16&4&2&2&23\\ 10&16&4&2&2&2&2\\ 10&16&4&2&2&2&2\\ 10&16&2&2&2&2&2\\ 10&16&2&$	<pre>Link 1:sex/mean_big_context_conte</pre>	: 00
$\begin{array}{c} 10:16:40,107\\ 10:16:40,112\\ 10:16:40,112\\ 10:16:40,130\\ 10:16:40,257\\ 10:16:40,257\\ 10:16:40,257\\ 10:16:40,257\\ 10:16:40,333\\ 10:16:41,690\\ 10:16:41,760\\ 10:16:41,760\\ 10:16:41,760\\ 10:16:41,951\\ 10:16:41,951\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,213\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,223\\ 10:16:42,23\\ 10:16,23\\ 10:16,23\\ 10:16,23\\ 10:16,23\\ 10:16,23\\ 10:16,23\\ 1$	<pre>Link 1: SumAr Mean Dife_content to Let Diff Links return to 00 00 00 00 00 00 00 00 00 00 00 00 00</pre>	: 00
$\begin{array}{c} 10:16:40.107\\ 10:16:40.112\\ 10:16:40.182\\ 10:16:40.182\\ 10:16:40.252\\ 10:16:40.252\\ 10:16:40.252\\ 10:16:40.252\\ 10:16:40.333\\ 10:16:41.690\\ 10:16:41.690\\ 10:16:41.760\\ 10:16:41.760\\ 10:16:41.833\\ 10:16:41.833\\ 10:16:41.232\\ 10:16:42.232\\ 10:16:42.232\\ 10:16:42.232\\ 10:16:42.2352\\ 10:16:42.352\\ 10:16:42$	<pre>Link 1:sex/mean_big_context_comest_come</pre>	: 00

#### Figure 30.5: Provision UART log

# Home page->setting->Mesh Info



Figure 30.6: Android key

#### 30.3.3 Via iOS App

Home page->setting->Mesh Info

无SIM卡 🗢	上午11:10	100% 🗲
<	Mesh Info	
IV Index: 0x000000	00	
Sequence Number:	0x01664C	
Local Address: 0x00	01	
NetKey List		>
AppKey List		>
<u>zeji</u>	Figure 30.7: iOS key	

#### 30.3.4 Via JSON File

Open "mesh\_database.json" inside the folder which contains "sig\_mesh\_tool.exe". Search "netKeys" and "appKeys" for net key and app key.



Figure 30.9: json iv

# 31 Operating Instructions for Telink-developed Bluetooth Mesh Decryption and Analysis Tool

## 31.1 Application Background

When debugging and developing Bluetooth Mesh products, in addition to opening the Log on the device side, sometimes it is also necessary to use a packet capture tool to analyze whether the format of the mesh messages in the air and the interaction flow is correct. Currently, the price of packet capturing instruments in the market is relatively expensive. In the absence of professional packet capturing tools, Telink-developed Bluetooth Mesh packet decryption analysis tool can be used for preliminary analysis.

This tool requires only one TLSR8258 Dongle (hereinafter referred to as Monitor) and one serial module, and then compile the 8258\_mesh\_monitor project(Download link of the SDK: http://wiki.telink-semi.cn/tools\_and\_sdk/BLE\_Mesh/SIG\_Mesh/sig\_mesh\_sdk.zip). Burn the compiled 8258\_mesh\_monitor.bin, and then monitor and decrypt the advertised mesh messages in the current mesh network, GATT proxy pdu and decrypted mesh messages encrypted with Device Key are not supported.

## 31.2 Operation Procedure

#### 31.2.1 Configure Monitor serial port

The packet capture tool uses the 8258\_mesh\_monitor compilation option, and MESH\_MONITOR\_EN has been enabled in the project settings. It shares the application layer code with 8258\_mesh project, and configures the IO of the serial port in the header file app\_config\_8258.h through the macro UART\_TX\_PIN.



Figure 31.1: mesh\_monitor\_uart\_io\_setting

Clean the compilation to get the 8258\_mesh\_monitor.bin file, which is burned into the Monitor.

## 31.2.2 Connect the serial hardware

Connect Monitor's uart tx and uart rx to the rx and tx of the serial module respectively, open the serial port debugging assistant (general-purpose serial port tool can be used), select the corresponding COM port, set the baud rate to 115200, and hex display.



Figure 31.2: mesh\_monitor\_hardware\_connection

1		Û	8	-	⊟	
Port Settings						
Port COM18(USB-SERIAL CH: V						
BaudRate 115200 V						
Data Bits 8 🗸 🗸						
Parity Bits None 🗸						
itop Bits 1 V						
Flow Ctrl None ~						
Com						
Close						
Receive Settings						
🔾 ASCII 💿 Hex						
Auto Wrap						
Auto Response						
Show Send						
Show Timestamp						
Show Log						
Output Log To File						
Telink Log Parsing						
Save Log Clear Log						
end Settings						
🔾 ASCII 💿 Hex						
Mutli Line Send						
Send New Line					Send	
Repeat Send 1000 🗘 ms						
0M18. 115200. 8. None. 1	Ru: O Bytes Tx: O Bytes			v	ersion	

Figure 31.3: mesh\_monitor\_baudrate\_setting

## 31.2.3 Add Monitor to a Mesh Network

Before joining the mesh network, Monitor is equivalent to an unprovisioned node and does not enable the monitoring function. After it is added to the mesh network with the app/gateway, it will automatically turn on the monitoring function.



#### 31.2.4 Log Parsing

After the Monitor joins the mesh network, it decrypts and outputs the advertise type mesh messages within the mesh network that it monitors through the serial port, in the format:

OxF5 + rf\_head + rf\_length + advA + mesh\_payload\_length(type+PDU) + Ox2A(Mesh type) + Network PDU + RSSI + Frequency Offset + Channel

- rf\_len: length of all data after it (excluding rssi, Frequency Offset, channel).
- advA: mac of the sending node.
- mesh\_payload\_length: data length of the mesh type and PDUs.
- RSSI: energy value in dBm. Note: data type is s8.
- FREQUENCY: Frequency deviation value. Unit is kHz. Note: data type is s16.
- CHANNEL: Indicates on which advertise channel the packet was sent.

The Network PDU corresponds to the member variables in the red box in the structure mesh\_cmd\_bear\_t in the figure below, which is consistent with the PDU format defined in the Mesh specification, and the specific mesh message is known by parsing the reported raw data according to the format.



Figure 31.4: mesh\_monitor\_report\_format

As shown in the following figure, after adding a lighting node and Monitor, the gateway sends onoff set and customized vendor set messages to the mesh network, respectively, and the lighting node receives them and replies with onoff status and vendor status.

Т			③ ① 函 - 日 >	<
Port Setti	ings		[14:13:18.008 (Rx)] E5 02 16 04 58 E4 38 C1 04 0F 24 C7 04 34 00 00 01 00 FF FF 4D 82 02 01 04 FA 04 00 25 opoff set	
Port	COM3(USB-SER	RIAL CH3 <sup>,</sup> V	[14:13:18.499 (Rx)]	
BaudRate	115200	$\sim$	F5 02 17 0C 5E E7 38 C1 A4 10 2A C7 0A 2B 00 00 02 00 01 00 4D 82 04 00 01 0A DF 0B 00 25 onoff status [14:16:43.899 (Rx)]	
Data Bits	8	$\sim$	F5 02 17 AA 5B E4 38 C1 A4 10 2A C7 0A 3B 00 00 01 00 FF FF 4D C2 11 02 01 0B EA F9 FF 25 [14:16:44.312 (Rx)] vendor set	
Parity Bits	None	$\sim$	F5 02 16 0C 5E E7 38 C1 A4 0F 2A C7 0A 2C 00 00 02 00 01 00 4D C4 11 02 01 DE 05 00 27	
Stop Bits	1	$\sim$	venuor status	
Flow Ctrl	None	$\sim$		
	Close			
Receive S	Settings			
O ASCII	🔘 Hex	x		
🔽 Auto W	/rap			
Auto R	esponse			
Show S	Send			
🔽 Show T	ïmestamp			
🔽 Show L	.og			
Output	t Log To File			
Telink I	og Parsing			
Save Lo	g	Clear Log		
Send Set	tings			
	He	x		
Mutli L	ine Send			
Send N	lew Line		Send	
Repeat	Send 1000 🗘	ms		~
COM3, 1152	:00, 8, None, 1		Rx: 118 Bytes Tx: 0 Bytes Version: v3.	0.4



#### **31.2.5 Extended Functions**

(1) If monitoring the mesh packets of the default network (the network before performing the mesh operation), you can send a command to the Monitor through the serial port to enable the monitoring function.

a8 ff + 00 + monitor\_en $_{\circ}$ 

Telink

- monitor\_en is 1 for on, 0 for off.
- Setting success returns a8 ff + 00 + 00.
- Setting failure returns a8 ff + 00 + 01 (01 means error code is 1).

(2) Monitor enables sno filter (i.e. relay protect list) by default, if you need to disable it, you can set it by sending command to Monitor through serial port.

a8 ff + 01 + sno\_filter.

- A sno\_filter of 1 turns on sno filtering and 0 turns off sno filtering.
- Setting success returns a8 ff + 01 + 00.
- Setting failure returns a8 ff + 01 + 01 (01 means error code is 1).
- (3) Monitor can specify the packet capture channel, which can be set by the following commands

a8 ff + 02 + chn1 + chn2 + chn3



- chn of 25, 26, 27 are advertise channels. Here chn as parameters can be 1, 2, 3 respectively. For example: (a) a8 ff 02 25 (b) a8 ff 02 25 26 (c) a8 ff 02 25 26 27. Note: 25, 26, and 27 are in hexadecimal, which corresponds to 37, 38, and 39 in decimal.
- Setting success returns a8 ff + 02 +00.
- Setting failure returns a8 ff + 02 + 01 (01 means error code is 1)

(4) Monitor provides whether to capture unprovision beacon advertise packets that are not in the network, which can be commanded through the serial port.

a8 ff + 03 + unprovision\_beacon\_enable

- unprovision\_beacon\_enable is 1 to turn on packet capture, 0 to turn it off.
- Setting success returns a8 ff + 03 + 00.
- Setting failure returns a8 ff + 03 + 01 (01 means error code is 1).

These are the steps of Bluetooth Mesh Packet Decryption and Analysis Tool, which can quickly locate and analyze the problem by capturing the over-the-air packets.

# 32 Spirit LPN

## 32.1 Function Description

The Spirit LPN is a privately defined low-power node that does not need to establish a friendship, but is implemented by the node periodically waking up for a period of time to receive packets. The default of the demo SDK is to wake up every 360ms, and each wake up lasts for 20ms to perform a broadcast scan and receive commands. Since the node is not receiving packets all the time, the sender (gateway) needs to send commands continuously to ensure the success rate. The duration of continuously sending commands should be greater than the wake-up cycle and scanning cycle of low-power nodes. For example, the demo SDK should have a minimum of 360+20=380ms. and it is recommended to exceed 1000ms.

## 32.2 Configuration

#### 32.2.1 Set Gateway to Continuous Packet Sending Mode

#### 32.2.1.1 Enable Key Detection

For testing purposes, the spirit LPN is switched on and off by pressing the SW2 button of the gateway, and UI\_KEYBOARD\_ENABLE is turned on in the 8258\_mesh\_provision project to enable key detection.

app_config_8258.h (vendor\mesh_pro	vision) <b>%</b> app_config_8278.h (vendor\mesh_provision) app_mesh.h (	(proj_lib\sig_mesh) ble_ll_
app_config_8258.h Symbol Name (Alt+L)	146: <b>#endif</b> 147: 148: // keyboard 149:- <b>#if SMART PROVISION ENABLE</b>	
# endif       ^         # BLE_REMOTE_PM_ENABLE       _         # PM_DEEPSLEEP_RETENTION       _         # BLE_REMOTE_SECURITY_EN/       _         # BLE_IR_ENABLE       _	150: #define UI_KEYBOARD_ENABLE       1         151: #endif       1         152:       153:_#ifndef UI_KEYBOARD_ENABLE         154: #define UI_KEYBOARD_ENABLE       1         155: #endif       1         156:       4	
□	157:-     #if UI_KEYBOARD_ENABLE       158:     #define MATRIX_ROW_PULL       159:     #define MATRIX_COL_PULL       160:     PM_PIN_PULLUP_10K	00K // drive pin pull // scan pin pull



Change "#if O" to "#if 1" in the mesh\_proc\_keyboard() function.

app.c (vendor\mesh_provision) * ×	app_mesh.h (proj	_lib\sig_mesh)	mesh_common.c (vendor\common)	mesh_common.h (vendor\a
app.c *	359: 360:	else if(KE mesh_s	<pre>EY_SW2 == kb_event.keycode[0]){ smart_provision_start();</pre>	
Symbol Name (Alt+L)	361: 362:	} #endif		
# if MD_SENSOR_EN     # endif     # if (BLT_SOFTWARE_TIMER_E	- 363: 364: 365: 366:	= <b>#if 1</b> if(KEY_SW2 static	<pre>mesh_proc_  2 == kb_event.keycode[0]){     u8 onoff=1;</pre>	keyboard
—₃ soft_timer_test0 →♣ endif -₃ app event handler	367: 368: 369: 370:	onoff access } #andif	= !onoff; s_cmd_onoff(0xffff, 0, onoff, CMD_NO_	_ACK, 0);
<b>3 app_host_event_callback</b> ■ ﷺ if (UIL KEVROARD_ENARLE)	371: 372:	##Hull	EST MODE EN)	





## 32.2.1.2 Configure the Numbers of Gateway Sending Packets Continuously



Set the number of extra packets in set\_material\_tx\_cmd() function, the interval of extra packets is 10-12ms.Among 10-12ms, 2ms is the running time deviation of the mainloop cycle. Change "#if 0" to "#if 1", the packets will be sent continuously for 100 times with 10-12ms interval in demo. Of course, you can also set different values according to the demand, i.e. change the value of p\_tx\_head->val[0].

#### 32.2.2 Setting the Wake-up Period and Scan Window for LPN

By default, the wake-up period of spirit LPN node is 360ms, and the scanning window is 20ms, if you want to change the wake-up period and scanning window time, you can follow the below modification.

Telink

т

## 32.2.2.1 Setting the Wake-up Period

	· · · · · · · · · · · ·	
	408: #define ADV_INTERVAL_UNIT	(ADV INTERVAL 10MS)
app_mesn.n	409:	
Complete Names (Alterna)	410: #define ADV_INTERVAL_1_2_S	<b>1920</b> //625*1920 = 1.2s
Symbol Name (Alt+L)	411:	
	412: #if MI_SWITCH_LPN_EN	
	413: #define ADV_INTERVAL_MIN	(ADV_INTERVAL_1_2_S)
# ADV_INTERVAL_MIN	414: #define ADV_INTERVAL_MAX	(ADV_INTERVAL_1_2_S)
# ADV_INTERVAL_MAX	415: #elif DU_LPN_EN	
🖻 🏶 elif DU LPN EN	416: #define ADV_INTERVAL_MIN	(DU_ADV_INTER_VAL-(DU_ADV_INTER_VAL/10))
ADV INTERVAL MIN	417: #define ADV_INTERVAL_MAX	(DU_ADV_INTER_VAL+(DU_ADV_INTER_VAL/10))
	418: #elif SPIRIT PRIVATE LPN_EN	
	419 #define ADV_INTERVAL_MIN	(ADV_INTERVAL_360MS)
elif SPIRIT_PRIVATE_LPN_	420 #define ADV_INTERVAL_MAX	(ADV_INTERVAL_360MS)
ADV_INTERVAL_MIN	421: #elit (GATT_LPN_EN  PROJEC	
ADV INTERVAL MAX	<pre>422: #define ADV_INTERVAL_MIN</pre>	(ADV_INTERVAL_160MS)
	<pre>423: #define ADV_INTERVAL_MAX</pre>	(ADV_INTERVAL_200MS)
	424: <b>□</b> #else	
# ADV_INTERVAL_WIN	<pre>425: #define ADV_INTERVAL_MIN</pre>	(ADV_INTERVAL_UNIT)
ADV_INTERVAL_MAX	<pre>426: #define ADV_INTERVAL_MAX</pre>	(ADV_INTERVAL_UNIT)
🗆 🏶 else	427: <b>#endif</b>	
ADV INTERVAL MIN	428:	
	429:-#if MI API ENABLE	

Figure 32.4: Setting the wake-up period

That is, change the value of ADV\_INTERVAL\_MIN, ADV\_INTERVAL\_MAX.

### 32.2.2.2 Setting the Scanning Window after Wake-up

vendor_model.c	283:[] 284: <b>#endif</b>
Symbol Name (Alt+L)	<pre>285: #if SPIRIT_PRIVATE_LPN_EN 286: #if SPIRIT_PRIVATE_LPN_EN 287: //note:there is 1.2s response delay after receive reliable command, refer to MESH_RSP_BASE_DELAY_STEP 288: //user should use bls_ll_setAdvParam() to set lseep time(adv interval) in soft timer mode. 299: f/vser can set gatt_adv_send_fing ==0 to stop gatt adv and save current 209: #define INDICATE_RETRY_CNT 6 201: #define INDICATE_RETRY_CNT 6 202: 203: mesh_sleep_pre_t mesh_sleep_time={0, RUN_TIME_US}; 204: 205: #if SPIRIT_VENDOR_EN 205: #if SPIRIT_VENDOR_EN 205: #df SPIRIT_VENDOR_EN 205: {ATTR_TARGET_TEMP}, 209: }; 300:</pre>

Figure 32.5: Setting the scan time

That is, change the value of RUN\_TIME\_US.

## 32.3 Function Demonstration

- Enable UI\_KEYBOARD\_ENABLE for 8258\_mesh\_provision project, then Compile 8258\_mesh\_spirit\_lpn and 8258\_mesh\_gw project, burn the compiled fw into 8258 dongle via BDT respectively. The gateway is plugged into the usb port and the spirit lpn is networked via sig\_mesh\_tool.exe.
- After the networking is completed, pressing gateway SW2 can control the onoff state of spirit\_lpn node. Every time it is pressed, the onoff status of spirit\_lpn node will be toggled.



## 32.4 Platform Access Setting

The spirit LPN is in generic mode by default, and can be used by the demo SDK's Gateway and App for networking.

If you are accessing Tmall Genie mode, set MESH\_USER\_DEFINE\_MODE to MESH\_SPIRIT\_ENABLE, please refer to Connect with a Platform section.

relint Semiconductor

# 33 Android and iOS APP User Guide

## 33.1 App download

The Android App can be obtained from the firmware SDK package, for example:

 $\label{eq:link_sig_mesh_sdk_app\android\TelinkBleMesh\TelinkBleMeshDemo-V4.1.0.0-20231113.apk.$ 

The iOS app can be obtained by searching for telinksigmesh in the App Store.

Developers can also recompile the App, and the corresponding code can be obtained from the firmware SDK package: \telink\_sig\_mesh\_sdk\app

## 33.2 Device Network

Networking is divided into manual networking mode and automatic networking mode.

### 33.2.1 Manual Provision Networking

#### 33.2.1.1 Add Device in Manual Mode

The Android/iOS version of the APP enables manual provision mode by default. After launching the app, click the upper right corner of the "+" button of the main interface to enter the add interface, the APP will automatically search for peripheral devices as shown below. You can click the button on the right side of the corresponding device to network, or you can click the button on the top right corner of the corresponding device in the devices in the list.

APP Homepage	Android Device List	iOS Device List
C Device Default Mesh	+ <b>Device Scan</b> C Selectable	🖌 Device Scan 🖒
ALL ON ALL OFF CMD LO CO(Pid-01) OE(Pid-01) 12(Pid OE(Pid-01) 12(Pid OE(Pid OE(Pid-01) 12(Pid OE(P	<ul> <li>address: [Unallocated] uuid:88744661080C3A3A940A0255556666666 me:: 6636665555502 Prepared</li> <li>scan device found</li> <li>ADD</li> <li>address: [Unallocated] uuid:1AC091DBF4DADF379126065555666666 Prepared</li> <li>scan device found</li> <li>ADD</li> <li>scan device found</li> <li>ADD</li> <li>address: [Unallocated] uuid:256703FAD6490634A1A40455556666666 me:: 66:66:66:55:50:04 Prepared</li> <li>scan device found</li> <li>ADD</li> </ul>	Image: Duration of the state of the sta
Q	ADD ALL	ADD ALL

#### Figure 33.1: Manual mode adding devices

## 33.2.1.2 Status During Manually Adding Devices

#### Android APP

The Scan-device found is the status that the device is scanned and found, and the left arrow expands the status of each state during the networking process (as shown below).

## Android Device Status

Android Device Status Lis
---------------------------

<	Device Scan Selectable	С
Prepared add	Iress: [Unallocated] d:88744661069C3A3A940A0255556 c: 66:66:66:55:55:02	×
> scan	- device found	ADD
add	Iress: [Unallocated]	X
a uuic mac	a: 1ACU91DBF4DADF3791260655556 a: 66:66:66:55:55:06	00000
Prepared		
> scan	- device found	ADD
add	Iress: [Unallocated]	X
uuic 🕎	d:256703FAD6690634A1A40455556	66666
Prepared		
	device found	

Figure 33.2: Status display for android manual mode add device process

#### iOS APP

The iOS version of the APP only displays the current status, more detailed information can be viewed in the APP's log record.





#### 33.2.2 Auto Provision Networking

From APP home page - setting - settings in the Provision Mode, select Normal (Auto) to switch to Auto provision mode. At home page click the upper right corner of the + sign to enter the Device Scan interface, at this time the title will show Device Scan (Auto provision) and it automatically adds all the surrounding unnetworked devices as the figure below. If scanning timeout, the Android app will pop-up the return button in the upper-left corner, the iOS version of the APP will pop-up Go Back button at the bottom of the app.

Networking Process		Scan Tir	neout
Device Scan		Aut	Scan C
address: 0x0006 uuid:11A0AB29593 81D3687652A09A3 9416E1 - mac: 99:99:99:66:66:02 Bind Success address: 0x0008 uuid:625742FB890 48930AA6BCE67C 6D5A159 - mac: A4:C1:38:59:FD:6F Provisioning		address: 0x0006 uuid:11A0AB29593 81D3687652A09A3 9416E1 - mac: 99:99:99:66:66:02 Bind Success	address: 0x0008 uuid:625742FB890 48930AA6BCE67C 6D5A159 - mac: A4:C1:38:59:FD:6F Bind Success
Figure	<b>33.4:</b> Android auto prov	ision	
Networking Process		Scan Timeout	_
Device Scan (Auto)	C AS DD: AS	Device Scan(Auto)	004 88:44:70:
GO BACK		GO BACK	

Figure 33.5: iOS auto provision

#### 33.2.3 Rescan Peripheral Devices

The icon at the upper right corner of Device Scan interface is the reload button. The manual provision networking mode can reload the device list as the figure below. The auto provision mode can re-scan the peripheral devices and automatically network as the figure below.

## Android Reload Device List

< Device Scan Selectable	G
address: [Unallocated] uuid:8B744661069C3A3A940A025555666 mac: 66:66:66:55:55:02 Prepared	×
> scan device found	ADD
address: [Unallocated] uuid:1AC091DBF4DADF37912606555566 mac: 66:66:66:55:55:06 Prepared	66666 ×
> scan device found	ADD
address: [Unallocated] uuid:256703FAD6690634A1A4045555666 mac: 66:66:66:55:55:04 Prepared	×
> scan device found	ADD

#### Rescan And Automatically Network



Figure 33.6: Android device reloads device list & rescan and auto-networking

#### 

iOS Reload Device List

## Rescan And Automatically Network



Figure 33.7: iOS device reloads device list & rescan & auto-networking

## 33.3 Device Interface

Device interface (APP home page) lists 4 states, directly connected node name is shown in blue, off state is shown in dark grey, offline state is shown in light grey with slash, and different Pid devices show different icons.

#### Telink SIG Mesh SDK Developer Handbook

C	<b>De</b> Defaul	<b>/ice</b> t Mesh	+
ALL ON	ALL OFF	CMD	LOG
-	•	×.	
0C(Pid-01)	0E(Pid-01)	10(Pid-01)	12(Pid-301)
Q	Ē	윦	\$
Device			

Figure 33.8: Android & iOS app device interface

#### 33.3.1 Refresh Device

The icon C on the top left corner of the Android/iOS APP homepage can refresh the current networked device status.

#### 33.3.2 All on/off

The Android/iOS APPs control all networked devices to turn on/off the lights by sending "all on/off" commands. Blue is on, dark grey is off.

#### 33.3.3 Single Device on/off

Click on the corresponding device icon to turn on/off the light. Blue is on, dark grey is off.

#### 33.3.4 CMD Command

The CMD command has built-in Vendor on/off, Vendor on/off no-ACK, Vendor on/off get, Generic on/off (For iOS, it is not built-in yet), Opcode Aggregator (Lightness Default/Range Get, TTL/Friend/Relay Get), Device/App Key Get command for iOS. Users can also customize the commands through APP Custom for Android (APP Vendor Data for iOS), which can customize the following commands.

• Access Type: (decide what key to use to send)



- dst adr: (destination address)
- opcode: operation code
- params: the parameters that follow the opcode
- rsp opcode: (response code for opcode)
- rsp max: (if rsp opcode is not 0, the value is the quantity of nodes are expected to receive replies, and if not enough replies are received, a retry will be performed), retry count (the maximum number of times a retry will be performed if the number of replies specified by rsp max is not received)
- ttl: time to life.
- tid position: this value must be 0 for non-vendor commands and indicates the position of the tid for vendor commands (a value of 0 indicates that there is no tid field, 1 indicates that the tid is at the para[0] position, and 2 indicates that it is at the para[1] position).

Android CMD - 1	Android CMD - 2	iOS CMD Interface
< CMD >	< CMD >	<b>Հ</b> СМД
Action: Vendor On V	Action: Vendor On	vendor_on vendor_off vendor_onoff_get
Access Type: Application(App Key)  *dst adr: 0x FFFF	Actions       *d.       Vendor On	devKey_aggregator vendor data:
*opcode: 0x 0211C2 params: 0x 0100 01 00	<sup>10</sup> Vendor Off pa Vendor On/Off Get	A3 FF 00 00 00 00 02 00 02 00 80 72 02 00 04 80 0C 04 80 0F 04 80 26
rsp opcode: 0x 0211C4 rsp max: 0	<ul> <li>Vendor On NO-ACK</li> <li>Vendor Off NO-ACK</li> </ul>	
retry count: 2ttl: 10	ret     Generic On       ttl     Generic Off	
tid position: 1	Copcode Aggregator(Lightness Default Get, + Lightness Range Get) Opcode Aggregator(TTL Get + Friend Get + Relay Get) [Custom]	

Figure 33.9: Android & iOS CMD interface

#### 33.3.5 Log

The log function is enabled by default to record the log information of the current operation. For Android version, you can turn on/off Enable LOG in APP homepage – setting – settings. Click Save in File button to save the logs, save path: default storage/TelinkBleMesh. (Note: iOS automatically save, if you need to export the log to PC, you can use iTunes – File Sharing – TelinkBleMesh to export the log to PC.) Click the Refresh button to refresh the log (for iOS version, exit and re-enter the log interface), click Clear button to clear the log information.

< Lo	g
2-24 15:47:54.958/SIG-Mesh : Mesh etworkKey=721E9A249C2B1BAC69- etKeyIndex=0x0, appKey=1D6E0295 ppKeyIndex=0x0, ivIndex=0, sequenc rovisionIndex=16, sccenes=0, groups-	Info{nodes=3, 42C6A76082CB92, 41836FD03A6277007F561F2C, 2eNumber=1792, localAddress=1, =8}
2-24 15:47:54.990/SIG-Mesh : Splas	hActivity onCreate
2-24 15:47:55.023/SIG-Mesh : permi	ssion check pass
2-24 15:47:55.024/SIG-Mesh : Splas	hActivity onResume
2-24 15:47:55.532/SIG-Mesh : Splas	hActivity finish
2-24 15:47:55.544/SIG-Mesh : MainA	Activity onCreate
2-24 15:47:55.613/SIG-Mesh : Mesh	Service#init
2-24 15:47:55.625/MeshController :	handleNetworkInfoUpdate : 2048 -
2-24 15:47:55.650/SIG-Mesh : post e	event : com.telink.ble.mesh.EVENT
2-24.15:47:55.650/Networking : init :	etworkinoopuateEvent
2-24 15:47:55 652 (MechController)	bluetooth event: 12 - bluetooth
nabled	processer event. 12 - processer
2-24 15:47:55.652/SIG-Mesh : post e TYPE_BLUETOOTH_STATE_CHANGE	event : com.telink.ble.mesh.EVENT BluetoothEvent
2-24 15:47:55.652/Networking : ena	bleDLE: false value : 11
2-24 15:47:55.746/SIG-Mesh : Main/	Activity onResume
2-24 15:47:55.746/SIG-Mesh : main	auto connect
2-24 15:47:55.747/MeshController :	auto connect
2-24 15:47:55.747/MeshController : MODE_AUTO_CONNECT	start scan:
2-24 15:47:55.824/MeshController :	scan:nullmac:
5:55:55:55:39:00 -record: 02:01:06:0 6:28:18:00:48:E6:D6:C0:92:5E:84:FE: :50:00:00:00:00:00:00:00:00:00:00:00:01:0 0:00:00:00:00:00:00:00:00:00:00:00	)3:03:28:18:00: :1E:FF:11:02:00:39:55:55:55:5 2:03:04:05:06:07:08:09:0A:0B:
2-24 15:47:55.825/MeshController :	check network id pass? false
2-24 15:47:55.846/MeshController : 5:55:55:55:64:00 -record: 02:01:06:0 6:28:18:00:48:E6:D6:C0:92:5E:84:FE: A0:00:00:00:00:00:00:00:00:00:00:01:0 0:00:00:00:00:00:00:00:00:00:00:00:00	scan:nullmac: )3:03:28:18:00: :1E:FF:11:02:00:64:55:55:55:5 )2:03:04:05:06:07:08:09:0A:0B:
SAVE IN	I FILE

Figure 33.10: Android & iOS log interface

## 33.3.6 Device Setting (Light device)

Long press the icon of the networked Light device on the Android/iOS APP homepage to enter Device Setting interface.

< De	vice Setting	)
CONTROL	GROUP	SETTINGS
ele adr: 16		
Lum Level(at ele adr: 0x10):		+
Lum(100) (at ele adr: 0x1	10):	
Temp Level(at ele adr: 0x11):	_	+
Temp(100) (at ele adr: 0>	(11):	

Figure 33.11: The light device setting interface

#### 33.3.6.1 Light Device Control

The "Ele Adr X" is used to switch the device on and off; "Lum Level" is used to adjust the brightness of the device; "Temp Level" is used to adjust the colour temperature of the device.

#### Light Device Network Lighting Control

When the Light device supports Light LC Server, the Network Lighting Control sub-page entry will appear under the Control page of the device. The Network Lighting Control page is shown in the following figure.

- (1) The Light device needs to turn on the Enable LC mode and Enable LC Occupancy mode switches before it will process the sensor data reported by the sensor device and determine whether to execute the Light Control action.
- (2) The Set LC light on/off button is used to send the LightLCLightOnOffSet command to the Light device.
- (3) The following Properties list contains 3 Lightness parameters and 7 Time parameters, all of which are already configured by default on the device side. Users can get and set these parameters, for detailed description of each parameter, please refer to the Handbook document on the firmware side or the sig mesh protocol document.

IM卡 <b>令</b>	10:38 Device Setting	82% 4	无SIM卡 <b>令</b> 1 く Lightin	6:21 g Control
ONTROL	GROUP	SETTINGS	Enable LC mode	
∱ <sup>™</sup> Netwo	ork Lighting Control	>	Enable LC Occupancy	y mode
			Set LC light on/off	
adr:0x423	ele adr:0x425		Properties:	
n I evel (at ele ad	(r:0x423):	+	Lightness On	value
um(5)(at ele adr:0	x423):	I	Lightness Prolong	value
-0-			Lightness Standby	value
emp Level(at ele a	dr:0x424):	+	Time Eade On	value
emp(100)(at ele ad	dr:0x424):		Time Fade On	Value
			Time Run On	value
			Time Fade	value
			Time Prolong	value

#### Figure 33.13: Light Device Lighting Control interface

#### Sensor Device Sensor Control

When the Sensor device supports Sensor Server, the Sensor Control sub-page entry will appear under the Control page of the device. The Sensor Control page is shown in the following figure.

- (1) The publish adr is the address where sensor status data can be received. The publish adr of 0 means that the sensor does not report status data. The publish adr of 0xFFFF means that all devices can receive sensor status data. If a single device is required to receive status data from this sensor, it is needed to set publish adr to the element address where the Light LC Server for that device is located.
- (2) The period is set to 0 by default, which means that Sensor Data will not be reported periodically, but data reporting will be done when there is a change in the value of Sensor Data.
- (3) Sensor Data is the sensor data reported by the sensor via the SensorStatus command, Sensor Descriptor is the configuration parameter cured by the sensor reported by the sensor via the SensorDescriptorStatus command, and SensorCadence is the sensor's modifiable configuration parameters reported by the sensor via the SensorCadenceStatus command.



#### Figure 33.14: Sensor Device Sensor Control interface

## 33.3.6.2 Single Device Group

Telink

т

The "Group" is used to group the device (a single device supports a maximum of 8 groups).

<	Device Setting	
CONTROL	GROUP	SETTINGS
Living room		
J.		
Kitchen		$\checkmark$
Master bedroon	n	$\checkmark$
Secondary bedr	oom	
Balcony		
Bathroom		
Hallway		
others		

#### Figure 33.15: Android & iOS add group interface

## 33.3.6.3 Light Device Settings

The Settings menu enables user to view the UUID, and execute operations including Device Config, Composition Data, Network Keys, Subnet Bridge Setting, Schedule Setting, Subscription Models, Device Ota, Publication, and Kick Out.

< De	evice Setting	g
CONTROL	GROUP	SETTINGS
UUID: 650BB419	– D91EDF3586980C5	555666666
H Device C	Config	>
🕅 Composi	ition Data	>
🛱 Network	Keys	>
🛏 Subnet B	Bridge Setting	>
C Schedule	er Setting	>
Gochedule	a osting	
Subscrip	tion Models	>
↓ Device C	DTA	>
	( ) 0015	
((•)) Publicati CTL)	on (ele : 0018 m	odel:
	KICK OUT	

Figure 33.16: Android & iOS settings interface

#### **Device Config**

Telink

т

The Device Configuration is to configure the device's TTL, Relay and Relay Retransmission, Secure Network Beacon, Gatt Agent, Nedeidentity, Friend, Key Refresh Phase, Network Transmission.

< Device Config	
Default TTL The Defoult TTL state determines the TTL value used wher sending messages. value: 0A	~
Relay & RelayRetransmit The Relay state indicates support for the Relay feature; The Relay Retransmit state is a composite state that controls parameters of retransmission of the Network PDU relayed the node. Value: enabled retransmit count: 05 retransmit interval steps: 02	e by
Secure Network Beacon	>
GATT Proxy	>
Node Identity	>
Friend	>
Key Refresh Phase	>
Network Transmit	>

Figure 33.17: Android & iOS Device Config

#### **Composition Data**

Telink

T

The "Composition Data" is used to view the data of the device (including: cid/pid/vid/crpl/features/relay support/proxy support/freind support/low power support/position type of each sig model and vendor model). Clicking the icon C on the top right corner can refresh the data.

<	Composition Data	G
Composition-Da cid: 0x0211 pid: 0x0001 vid: 0x3233 crpl: 0x0069 features: 0x000 relay support friend support friend support low power sup elements: (2) element adr. 0 SIG model – SIG model – SIG model – SIG model –	ata: ata: true : true t: true poport: false	
ents: (2) ment adr: 0 G model – G model –	0x0010 0x0000 - config server 0x0002 - health server 0x0003 - health client 0xFE00 - firmware update server 0xFF00 - object transfer server 0x7F00 - object transfer server 0x1200 - Time Server 0x1200 - Time Server 0x1200 - Generic OnOff Server 0x1000 - Generic Level Server 0x1004 - Generic Default Transition	1 Time
SIG model – SIG model –	0x1006 – Generic Power OnOff Serv 0x1007 – Generic Power OnOff Setu 0x1203 – Scene Server 0x1204 – Scene Setup Server 0x1206 – MeshScheduler Server 0x1207 – MeshScheduler Setup Serv 0x1300 – Light Lightness Server 0x1303 – Light Lightness Setup Server 0x1304 – Light CTL Server 0x1304 – Light CTL Setup Server el – 0x000211 0x1001	rer Ip Server ver ver

#### Figure 33.18: Android & iOS Composition Data

#### Networ Keys (iOS: NetKey List / AppKey List)

In a Network users can create different Network Keys (iOS: NetKey List / AppKey List). In Network Keys (iOS: NetKey List / AppKey List) you can view the key bound to the current device, you can also configure the specified node with a new Network Key in order to connect different keys to different devices, and also share the specified node out to become a shared device by means of the key. The detailed operation is as follows:

**Preset conditions**: prepare two mobile phones A and B; add more than 2 devices to mobile phone A for default network key.

#### Steps for Android: (Android phone as phone A)

(1) Mobile phone A creates a new Net Key for the specified device. The detailed operation:

Long press a device that needs to creat a new Net Key on the APP homepage – Settings – Network Keys – Click "+" on the upper right corner to select a Net key (Currently, there are two built-in Net / APP key, which can be viewed in APP home page – click the Network in the lower right – Mesh info).

(2) Mobile phone A shares device to mobile phone B by sharing Net Key:



Mobile phone A APP home page – Setting – Share – Export – select the newly created Net Key – export by file/QR code;

Mobile phone B APP home page – setting – Manage Network – Import button at the bottom right corner – Import by file/QR code.

At this time, the device with the newly configured Net key just now becomes a shared device, and mobile phone B can only get the status of the device corresponding to the Net key, and the other device shows offline status due to different Net key.

#### Steps for iOS: (iOS phone as phone A)

(1) Mobile phone A creates a new Net Key for the specified device. The detailed operation:

Click Network in the lower right corner of APP homepage – Mesh info – Netkey List – create a new Net Key;

Return to Mesh info interface – App Key List – create a new App Key (Note: the key is required to be the same as the currently existing App Key; index, BoundNetkey bind to the newly created Net Key);

Long press a device that needs to creat a new Net Key/App Key on the APP homepage – Settings – NetKeys List – Click "+" on the upper right corner to select a Net key – Done – return to Device Setting – select AppKey List – Click "+" on the upper right corner to select a App key – Done.

(2) Mobile phone A shares device to mobile phone B by sharing Net Key:

Mobile phone A APP home page – Setting – Manage Network – Click the Network just configured Network Key – Share Export – Select the new Network Key just created – Export by file/QR code-Export;

Mobile phone B APP home page – setting – Manage Network – Import button at the bottom right corner – Import by file/QR code.

At this time, the device with the newly configured Net key just now becomes a shared device, and mobile phone B can obtain and control the status of the device corresponding to this Net key, and the other device displays offline status due to the difference in Net key.

#### Subnet Bridge Setting

The Subnet Bridge feature allows bridging tables to be configured to multiple subnet (Network Keys) nodes within a Network, allowing messages to be forwarded to specific subnets. For example, if node 1 is configured with shared devices for Netkey1 and Netkey2, and node 2 is configured with private devices for Netkey1 only, and Netkey2 wants to control the private devices of Netkey1, then it needs to configure a bridge table from network Netkey2 to Netkey1 by configuring shared node 1.

Mobile phone A operation (initially with network sharing devices): APP home page long press a shared device (configured Netkey1 and Netkey2, Netkey2 has been shared to mobile phone B) – setting – Subnet Bridge Setting – Turn on Enable Subnet Bridge switch – Click ADD Subnet Bridge button. – Add Bridging Table interface, for Net key 1 fill in the shared Net key; for Net Key 2 fill in the Net key that needs to be converted (Note: that is, the Net key of the original network sharing device) – For Address 1 Fill in the Local Address of the Net Key that has been shared (Viewing steps: mobile phone B that imports Net Key through sharing: APP home page – Network – Mesh info). – For Address 2 fill in the short address of the device to be controlled – Add Bridge Table to save.

Mobile phone B operation (get netkey through sharing): long press the node specified by mobile phone A to enter Device Setting interface to switch the control node.



#### Scheduler

Click "+" on the top right corner of the Scheduler list interface to add scheduler. After setting the conditions in the scheduler setting interface, click 🕙 to get and set the time (click setTime for iOS). Then click to save scheduler (Note: Scheduler is turned off by default, the device needs to enable MD\_TIME\_EN macro).

	Scheduler List	+ < Scheduler Setting 🛇 🗸	Scheduler List +	< 0 setTime 🗸
¢	Scheduler List	<ul> <li>Custom</li> <li>Year</li> <li>Any</li> <li>Custom</li> <li>year(2000-2099)</li> <li>Month</li> <li>SelectAll</li> <li>Jan</li> <li>Feb</li> <li>Mar</li> <li>Apr</li> <li>May</li> <li>June</li> <li>July</li> <li>Aug</li> <li>Sep</li> <li>Oct</li> <li>Nov</li> <li>Dec</li> <li>Day</li> <li>Any day</li> </ul>	Scheduler List +	Vear: Any year          Custom(2000 to 2009)         2000-2099         Month:       All         Jan       Feb         All         Jan       Feb         Jul       Aug         Oct       Nov         Dec       Day:         Any day         Custom(1 to 31)
	ADD	Custom day(1-31) Hour Any hour Once a day (at a random hour) Custom hour(0-23) Minute Any minute		1-31         Hour:       ✓         Any hour of the day         Once a day         Custom(00 to 23 hours)         0-23         Minute:       ✓         Any minute of the hour         Every 15 minutes(0, 15, 30, 45)         Every 20 minutes(0, 20, 40)

Figure 33.19: Android & iOS Scheduler

The Schedulers added in the Scheduler list interface can also be edited by clicking . After setting the conditions in the scheduler setting interface, click to get and set the time (click setTime for iOS). Then click to save scheduler.

< Scheduler List +	< s	Scheduler Se	tting 🕲 🗸
scheduler index: 0x0 Year: Any Month', Jan/Feb/Mar/Apr/May/June/July/Aug/ Sen/Oct/Nwork Day: Any Minute: Any Second: Any Week: Monday/Tuesday/Wednesday/Thursday/ Friday/Saturday/Sunday	Index: 0x0 Year Any Custom year(2000 Month SelectAll	0-2099)	Mar
Proday/saturoay/sunday	<ul> <li>Jan</li> <li>Apr</li> <li>July</li> <li>Oct</li> <li>Day</li> <li>Any day</li> </ul>	<ul> <li>Feb</li> <li>May</li> <li>Aug</li> <li>Nov</li> </ul>	<ul> <li>✓ Mar</li> <li>✓ June</li> <li>✓ Sep</li> <li>✓ Dec</li> </ul>
	day(1-31) Hour Any hour Once a day	) r (at a random hou	r)
	hour(0-23 Minute Any minute	3)	

Figure 33.20: Android & iOS edit Scheduler

#### Subscription Models



Subscription models can be viewed for the currently supported Models for the device:

- ID: 0x1000 (model name: Generic onoff server)
- ID: 0x1300 (model name: Light Lightness server)
- ID: 0x1303 (model name: Light CTL server)
- ID: 0x1306 (model name: Light CTL Temperature server)
- ID: 0x1307 (model name: Light HSL server)

#### **Device OTA**

#### Android APP:

Device OTA can perform GATT OTA upgrade on the device. The OTA interface can display the current device information, the different pid upgrade options of devices (unticked by default, users can tick the item as needed), click select file to select firmware, it shows target firmware version after selecting the firmware. Click "START" to start the upgrade, it will prompt start OTA and display the upgrade progress. When the upgrade is completed, it displays OTA\_SUCCESS and the progress is 100%, and the device flashes slowly. To check whether the device is upgraded to the target version, users can refresh and view the vid data by long pressing the device on the APP homepage – settings – Composition Data (refer to section 2.6.3.2 Composition Data ).



Figure 33.21: Android device OTA interface

#### iOS APP:
The OTA interface can display the current device information and the target version Pid and Vid. Tick the corresponding version and click "START" to start the upgrade. When the upgrade is completed, the device flashes slowly. To check whether the device is upgraded to the target version, users can refresh and view the vid data by long pressing the device on the APP homepage – settings – Composition Data (refer to section 2.6.3.2 Composition Data ).

<	OTA Pid:0x1 Vid:0x4100	()	<	OTA Pid:0x1 Vid:0x4100	()	<	OTA Pid:0x1 Vid:0x4100	(!)
8258_n	mesh_V4.1.0.0_20230926 PID:0x1VID:0x4100		82	58_mesh_V4.1.0.0_20230926 PID:0x1 VID:0x4100		8258_me	sh_V4.1.0.0_20230926 PID:0x1 VID:0x4100	
8258_n	mesh_V4.2.0.0 PID:0x1 VID:0x4200	$\checkmark$	82	58_mesh_V4.2.0.0 PID:0x1 VID:0x4200	$\checkmark$	8258_me	sh_V4.2.0.0 PID:0x1 VID:0x4200	~
ſarge ve	Current device info and version ersion information	ormation						
							Hits OTA success	
							Done	
				During the upgrade proces	s			
	OTA tips			OTA:20.7%			OTA success	
	Start OTA						Start OTA	
			Figure	<b>33.22</b> : iOS device OTA i	nterf	эсе		

## Publication (ele: xxxx model: CTL)

The Android/iOS APP will send status every 20 seconds after opening the corresponding device publication (it can be viewed in the log interface, CT light reports Ctlstatusmessage, HSL light reports Hslstatusmessage).

#### KICK OUT

The "KICK OUT" is used to kick out the current device. After kicking out, the device flashes slowly, and the device will be in the state of pending network.

# 33.3.7 Device Setting (Switch Device)

Long press the SW10 + SW13 keys of the unnetworked Switch remote control to trigger the broadcast (Note: the flicker frequency of the unnetworked state light is 200ms/s, and the networking state is 500ms/s) for networking. After the networking is successful, the broadcast also needs to be triggered. Long press the Switch icon on the APP homepage to enter the Device Setting interface to connect the remote control. At this time, the bottom of the interface will show "Device Connected" which represents the connected. If the connection fails, the position will pop up a button and user can click to reconnect.

Remote Control
TROL SETTINGS
tr: 0x 000C
× 1001
X C000
0 to reset default publication SEND
× 000D
2001
Adr 0 to reset default publication SEND
0× 000E
0× 1001
0× C002
dr 0 to reset default publication SEND device connected

Figure 33.23: Android & iOS switch device setting interface

# 33.3.7.1 Switch Device Control

The 0x0008 in Eleadr corresponds to Switch remote control SW7/SW10 keys, 0x0009 corresponds to SW8/ SW11, 0x000A corresponds to SW9/SW12, 0x000B corresponds to SW3/SW6, and Model can execute Switch-supported models (for details, please refer to Switch Device Composition Data). 0xC000 in Pubadr corresponds to Living room in Group, 0xC001 corresponds to Kichen, 0xC002 corresponds to Masterbedroom, and 0xC003 corresponds to Secondary bedroom. The Group to be controlled can be set at the specified key.

# 33.3.7.2 Switch Device Setting

Please refer to 33.2.6.3 Light Device Settings.

# 33.4 Group Interface

There are 8 groups in the Group interface (refer to 2.6.2 Single Device Group, Device interface – long press a device – click group for grouping), and assign devices to groups before operation.

#### Telink SIG Mesh SDK Developer Handbook

Group
The Living room
In Kitchen
To Master bedroom
🕞 Secondary bedroom
The Balcony
To Bathroom
🕞 Hallway
in others
오 🖆 器 🌣 Group
Figure 33.2

# 33.4.1 On/Off Group

The Group interface enables user to On/Off the devices belonging to the corresponding group.

## 33.4.2 Group Setting

Long press the corresponding Group to enter the Group Setting interface.

	<	۲ Group	✓ Group Setting
	Living roor	Living room Devices:	Living room Devices:
		• •	• •
	16(Pid-01)	16(Pid-01) 18(Pid-01)	16(Pid-01) 18(Pid-01)
	Lum(100)(at	Lum(100)(at group adr:0xC000):	Lum(100)(at group adr:0xC000):
	Temp(100)(a	Temp(100)(at group adr:0xC000)	Temp(100)(at group adr:0xC000):
	Extend A	Extend Address Control:	Extend Address Control:
	Lum Level(	Lum Level(at group adr:0xD000):	Lum Level(at group adr:0xD000):
	Temp Level	Temp Level(at group adr:0xD001):	Temp Level(at group adr:0xD001):
	Hue Level(a	Hue Level(at group adr:0xD002):	Hue Level(at group adr:0xD002):
	Sat Level(a	Sat Level(at group adr:0xD003):	Sat Level(at group adr:0xD003):
			- HSL

Figure 33.25: Android & iOS Group setting

# 33.4.2.1 On/Off Group Devices Individually

Click the device icon in the Group setting interface to on/off the device (the blue icon is On status, the gray is Off status), the blue device name is the direct connection device.

# 33.4.2.2 Lum & Temp

Lum adjusts the brightness of the devices belonging to the group, and Temp adjusts the color temperature.

# 33.4.2.3 Extend Address Control

Extend Address Control supports to enable the Level control function under the group, you can control the Lum Level, Temp Level, Hue Level and Sat Level of the group separately. Before grouping nodes, you need to open Extend SubscriptionLevel Service Model ID in APP Home – setting – settings.

## 33.4.2.4 HSL

The color palette enables user to adjust the color of the GRB, or user can adjust the color by adjusting R, G, B or H, S, L individually, RGB corresponds to HSL color, and V below the palette can adjust the brightness. Note: The device is required to enable the LIGHT\_TYPE\_HSL macro.



Figure 33.26: Android & iOS HSL interface

# 33.5 Network Interface

In the version V4.1.0.0, it adds a Network interface, which is used to view and set the current Network's Mesh Info, Scenes, Direct Forwarding, Mesh OTA, and Private Beacon individually.

Network	
Default Mesh	>
* Scenes	>
Direct Forwarding	>
Mesh OTA	>
+	
Private Beacon	>
Q E K	\$

Figure 33.27: Android & iOS network interface

# 33.5.1 Mesh info

In Mesh Info interface, we can see the current Network's Mesh Name (click the Edit button on the upper right corner to edit the name), Mesh UUID, Iv Index, Sequence Number, Local Address, Net Keys/App Keys name and its index and key. Long press the key to copy the corresponding key (currently Android app has three built-in keys, in iOS app we need to manually add the Netkey List, Appkey List).

	<	Mesh Info Default Mesh
Ê	ן ן	Mesh Name
ID	1	Mesh UUID
#	I	"3053926-3835-1853-4764-511888821118 V Index 0x00000001
12		Sequence Number
Ģ	L	<b>_ocal Address</b> 0x0001
<b>6</b>	1	Net Keys
	<b>^</b>	<b>Name</b> Default Net Key
	ID	Index 0x00
	0-	<b>Key</b> B4261ABE9E70B3C8C5E203336FCBB610
	<u></u>	Name Sub Net Key 1
	ID	<b>Index</b> 0x01
	От	Key E8A682B6380EF0EE235CB7E94F3C2930
	÷	Name Sub Net Key 2
	ID	Index 0x02
	_	Key
	-0	23.28. Android Mash info interfac

Figure 33.28: Android Mesh info interface



Figure 33.29: iOS Mesh info interface & Steps of adding Netkey / APPkey

## 33.5.2 Scenes

By Scene, we can save the current state of the specified device as a scene, in order to quickly start the set scene.

#### 33.5.2.1 Create Scene

Click the + sign in the upper right corner of the iOS APP Scene interface to save the current state of the specified device as Scenes (Android version currently only creates Scenes, you need to tap the Edit button to configure Scenes).



#### Telink SIG Mesh SDK Developer Handbook

<	Scene List	+	×	Scene List	+	<	Scene Lis	t
						* r	ame: scene_1 D: 01	▶ @
	_		Cre	eate Scene				
			please i	nput content	- 1			
	List Empty!			CANCEL CO	NFIRM			
	ADD							

Figure 33.30: Android create Scene interface

<	Scenes	+	<	Scene Setting	$\checkmark$	<	Scenes		+
			Devices:		All	scenel	0:0X0001	$\bigcirc$	Ø
			adr:0x16 on/off:ON		$\checkmark$	adr:0	x16 f:ON		
			adr:0x18 on/off:ON		$\checkmark$	adr:0	x18 f:ON		
			adr:0x1A on/off:OFI	F		adr:0	x1A f:OFF		



# 33.5.2.2 Edit Scene

After creating a Scene, click Scene List interface to edit Scenes.

**Android edit scene**: in the scene setting interface the left icon of "address" shows the current node switch status, the right of "address" shows the corresponding address, it only supports one element node in the right box (another element shows not support), you can tick the box to add the node to the scene. If a node that has been in the scene before needs to update its status, you can click to update the node to the current status.

Scene Set scene_1	ting 🕜
address: 0006	
• element: 0x0006	C C
• element: 0x0007	scene not support
🂡 address: 0008	
• element: 0x0008	
• element: 0x0009	scene not support
address: 000A	
• element: 0x000A	
• element: 0x000B	scene not support
	0

Figure 33.32: Android scene edit interface

**iOS edit scene:** in the scene setting interface, the left icon and the bottom of "adr" shows the current node switch status, the right of "adr" shows the corresponding address, check the corresponding box and click on the upper right corner of the Save button to update the selected node scene to the current state.

	<		Scene S	Setting	~
< C	Devi	ces:			All
	۲	adr:0x16 on/off:ON			$\checkmark$
	•	adr:0x18 on/off:ON			$\checkmark$
	Ţ	adr:0x1A on/off:OFF			$\checkmark$

Figure 33.33: iOS scene edit interface

## 33.5.3 Direct Forwarding

Direct Forwarding reduces the number of packets forwarded over the air by engaging commands in forwarding at specified path nodes (routing tables).

**Controllable Flooding**: means that when a mesh message is propagated outward from the source, it is similar to the ripples spreading in all directions when a stone is thrown into the water. The range of transmission is controlled through ttl and relay feature controls the nodes involved in forwarding, this transmission is called controllable flooding.



The controllable flooding does not control the direction of message delivery and wastes bandwidth on parts of the network that are not related to the message. For example, if there are 2 switches in the middle of a large conference room that control the podium and the lights in the back row, when controlling the lights at the podium, messages are also retransmitted between the back row light nodes.



Figure 33.34: Directed Forwarding & Managed flooding

**Routing table:** is a command from the starting point to the end of the route intermediate nodes involved in the forwarding of a path identity, the end point can be unicast, multicast and virtual address, each routing node saves the path information through it, that is, routing table. A subset of routing nodes is selected to form a path in the network, a route may have 1 or more paths.

When a message is sent in the specified routing method, it will check if the path exists, and if it does, it will be sent as routed, otherwise it will be sent as flooding, and route establishment will be triggered. Messages sent by flooding are encrypted with network key and messages sent by routing are encrypted with directed key (derived from network key). In DF\_TEST\_MODE\_EN mode, the node flashes when forwarding messages encrypted with the directed key.

The path establishment message is directed key encrypted, so all nodes flash their lights to indicate that the node forwarded the path establishment message. After the path is established, only the nodes on the path will flash, indicating that only the nodes on the path forwarded the directive.

# 33.5.3.1 Fixed Routing

Fixed Routing is configured and managed by the provisioner to forward nodes on a specified path. You need to turn on Direct Forwarding(Main), Direct Relay, Direct Proxy, and Direct Friend to the nodes on the path in the Direct Forwarding–Direct Toggles interface in advance. Note: If you don't check Direct forwarding(main), you will be prompted "(relay) check direct forwarding first" at the bottom.

< Direct Toggle List	
<pre>adr-0x0002 cid-1102 pid-0100</pre>	2
Direct Forwarding(main)	
Direct Relay	
Direct Proxy	
Direct Friend	
<pre>adr-0x0004 cid-1102 pid-0100</pre>	
Direct Forwarding(main)	
Direct Relay	
Direct Proxy	
Direct Friend	
<pre>adr-0x0006 cid-1102 pid-0100</pre>	
Direct Forwarding(main)	
Di (relay)check direct forwarding first	
Direct Proxy	

Figure 33.35: Fixed Routing Directangle Toggle list interface

The Direct forwarding interface allows you to add a fixed-route path to the mesh network by clicking the Add Table button at the bottom. A path contains a start point and an end point, as well as the nodes through which the path passes. When a message is routed from the start point, all nodes on the path will participate in forwarding the message, and nodes not on the path will ignore the message. You can see that the LEDs of nodes 0x06, 0x0e, and 0x16 on the path are flashing, while the LEDs of other nodes not on the path are not flashing.

Add Forwarding Table	Forwarding Table Added	App Homepage
< Add Forwarding Table	< Direct Forwarding	C Device +
Select origin device: select >	Direct Toggles >	ALL ON ALL OFF CMD LOG
<b>P</b> Node-0006	Table List	• • • •
Select target device: select >	<ul> <li>Origin 0x0006</li> <li>Target 0x0016</li> <li>Backward Validated</li> <li>Nodes on route</li> <li>Node-000E</li> </ul>	02(Pid-01) 04(Pid-01) 05(Pid-01) 08(Pid-01) 04(Pid-01) 0C(Pid-01) 0E(Pid-01) 10(Pid-01) 04(Pid-01) 14(Pid-01) 16(Pid-01) 10(Pid-01) 12(Pid-01) 14(Pid-01) 16(Pid-01)
SAVE	ADD TABLE	오 r k 💠 Device

Figure 33.36: Adding a fixed route

# 33.5.3.2 Non-fixed Routing

The non-fixed routing does not need to be operated by Direct Forwarding in the APP, it is created and maintained by the sender (path origin), which sends the message in a controlled flood and triggers the route establishment. The route establishment message is Directed Key encrypted, so all nodes flash to indicate that the node forwarded the route establishment message. After the path is established, only the nodes on the path flash lights, indicating that only the nodes on the path forwarded the command.

**Non-fixed Route Establishment Rules:** The figure below illustrates the creation of two paths from PO (Path Origon) to PT (Path Target) by selecting the shortest path that meets the set energy threshold.



Figure 33.37: Non-fixed route establishment rules

No Extend

Extend GATT only

Extend GATT & ADV

# 33.5.4 Mesh OTA

Mesh OTA allows simultaneous OTA upgrades of multiple devices specified by the mesh network, and there are three ways for Mesh OTA to be loaded: (1) No Extend (All nodes short packet loaded); (2) Extend GATT Only (Long packet loading for directly connected nodes and short packet loading for non-directly connected nodes); (3) Extend GATT & ADV (All nodes long packet loading), setting path: APP home page click on the lower right corner Setting-Settings-Extend Bearer Mode.

Extend options: Extend Bearer Mode

0: all are short packets

1: long packets for directly connected nodes, short packets for non-directly connected nodes

2: all are long packets (LPN upgrade)

Figure 33.38: Introduction to loading methods

#### Note:

- (1) Mesh OTA is turned off by default, you need to turn on the macro of node MD\_MESH\_OTA\_EN, otherwise it will not support Mesh OTA, and you can't check this device as the object of mesh OTA. Open method: in the mesh\_config.h file,
  - Enable MD\_MESH\_OTA\_EN
  - It needs to turn on DISTRIBUTOR\_UPDATE\_SERVER\_EN if the directly connected node after testing is as distributor mode.
  - If it is needed to use Telink's Extended Broadcast Packet mode to speed up the OTA time, you need to set EXTENDED\_ADV\_ENABLE to 1.
- (2) After the Mesh OTA upgrade is complete, exit the mesh OTA interface and re-enter to read off the upgraded version.
- (3) For iOS, put the upgraded bin file through itunes file sharing TelinkSigMesh to show up in the mesh OTA upgrade interface.

# 33.5.4.1 Distributor: Phone mode upgrade (App as distributor mode)

The Distributor chooses the Phone method to upgrade, it directly transfers OTA data to the target device through the phone.

Specific operation steps:

#### Android version:

At the APP home page, click on the bottom right Network – Mesh OTA according to the following step by step operation.

Mesh OTA		< Mesh OTA	
es:	select >	Devices:	
		adr-0x0004 Fwld:01003	335
1. Select Upgrade	• Device	state: Update Success additional: No changes to CPS	
2、select Upgrade 3、Choose phone	e File		
loading method			
le	>		
	Device	bin version: pid-01:00 vid-42:00	De
	T Device	Distributor. O Finishe O connected	De
progress(%d): /	151011	Initiate progress(100):	
5、Start u	pgrade		
ute progress (%d):		Distribute progress(100):	
ute progress (%d):	~	Distribute progress(100): update complete - recheck complete	

# Figure 33.39: Android phone method upgrade steps and upgrade completion interface

#### iOS version:

At the APP home page click on the bottom right Network – Mesh OTA according to the following step by step operation.

#### Telink SIG Mesh SDK Developer Handbook

Mesh OTA         revice list 1, Select Upgrade Device         cose all         r:0x2 pid:0x1 vid:0x4100         r:0x4 pid:0x1 vid:0x4100         r:0x4 pid:0x1 vid:0x4100         r:0x6 pid:0x1 vid:0x4200         S8_mesh_V4.1.0.0 pid:0x1 vid:0x4200         S8_mesh_V4.2.0.0 pid:0x1 vid:0x4200         S8_mesh_V4.1.0.0 pid:0x1 vid:0x4200         S9_mesh_V4.1.0.0 pid:0x1 vid:0x4200         S0_btain the current device version         tributor Progress:         S. Start upgrade         Distributor         get fw inf0       stait		
vice list 1, Select Upgrade Device se all	Mesh OTA	
a all choose all adr.0x2 pic 4 pid:0x1 vid:0x4100 adr.0x4 pic 6 pid:0x1 vid:0x4100 adr.0x6 pic 6 pid:0x1 vid:0x4100 adr.0x6 pic 7 file 2, select Upgrade File mesh_V4.2.0.0 pid:0x1 vid:0x4200 mesh_V4.2.0.0 pid:0x1 vid:0x4200 mesh_V4.2.0.0 pid:0x1 vid:0x4200 3, Choose phone loading method Phone bbtain the current device version or Progress: putor Progress 5, Start upgrade get tw info start growthing additional addited additional addited additi	<sup>ice list</sup> 1、Select Upgrade Dev	ice
:0x2 pid:0x1 vid:0x4100       adr:0x2 pid         :0x4 pid:0x1 vid:0x4100       adr:0x6 pid         :0x6 pid:0x1 vid:0x4100       adr:0x6 pid         :0x6 pid:0x1 vid:0x4100       adr:0x6 pid         :0x6 pid:0x1 vid:0x4100       adr:0x6 pid         :0x7 pid:0x1 vid:0x4200       adr:0x8 pid         :0x8 pid:0x1 vid:0x4100       adr:0x8 pid         :0x8 pid:0x1 vid:0x1 vid:0x4100       adr:0x8 pid         :0x8 pid:0x1 vid:0x1 vid:0x1 vid:0x100       adr:0x8 pid         :0x8 pid:0x1 vid:0x1 vid:0x1 vid:0x1 vid:0x1 pid       adr:0x8 pid         :0x8 pid:0x1 pid       adr:0x8 pid         :0x8 pid       :0x8 pid       adr:0x8 pid         :0x8 pid       :0x8 pid       adr:0x8 pid         :	pose all	
r:0x4 pid:0x1 vid:0x4100 adr:0x4 pid r:0x6 pid:0x1 vid:0x4100 adr:0x6 pid 0TA file 2, select Upgrade File 58_mesh_V4.10.0 pid:0x1 vid:0x4300 258_mesh 58_mesh_V4.2.0.0 pid:0x1 vid:0x4200 258_mesh 58_mesh_V4.1.0.0 pid:0x1 vid:0x4100 258_mesh 58_mesh_V4.1.0.0 pid:0x1 vid:0x4100 258_mesh 3, Choose phone loading method tributor: ConnectedDevice Phone Obtain the current device version intor Progress: 5, Start upgrade get fw info start get	r:0x2 pid:0x1 vid:0x4100	
ibbs pid:0x1 vid:0x4100   TA file   2. select Upgrade File   i8_mesh_V4.10.0 pid:0x1 vid:0x4300   i8_mesh_V4.2.0.0 pid:0x1 vid:0x4200   i8_mesh_V4.2.0.0 pid:0x1 vid:0x4200   i8_mesh_V4.1.0.0 pid:0x1 vid:0x4100   3. Choose phone   loading method   Distributor: ConnectedDevice Phone   Obtain the current device version   ator Progress   5. Start upgrade   get fw info   Start	0x4 pid:0x1 vid:0x4100	
Affle       2. select Upgrade File       OTA file         mesh_V4.10.0 pid:0x1 vid:0x4300       Imesh_V4.20.0 pid:0x1 vid:0x4200       Imesh_V4.20.0 pid:0x1 vid:0x4100         mesh_V4.10.0 pid:0x1 vid:0x4100       Imesh_V4.10.0 pid:0x1 vid:0x4100       Imesh_V4.10.0 pid:0x1 vid:0x4100         3. Choose phone       Ioading method       Imesh_V4.10.0 pid:0x1 vid:0x4100       Imesh_V4.10.0 pid:0x1 vid:0x4100         butor:       ConnectedDevice       Phone       Imesh_V4.10.0 pid:0x1 vid:0x4100         butor:       ConnectedDevice       Phone       Imitiator Phone         Distributo       Start       Imitiator Phone         Distributo       Start       Imitiator Phone	x6 pid:0x1 vid:0x4100	
B_mesh_V4.10.0 pid:0x1 vid:0x4200       B258_mesh         B_mesh_V4.2.0.0 pid:0x1 vid:0x4200       Image: Comparison of the comparison	A file 2、select Upgrade File	
38_mesh_V4.2.0.0 pid:0x1 vid:0x4200     Image: Constraint of the current device version initiator Progress     B258_mvi       3. Choose phone loading method     Image: Constraint of the current device version initiator Progress     Distributor       Obtain the current device version initiator Progress     5. Start upgrade     Distributor       get fw info     start     get fw info	58_mesh_V4.1.0.0 pid:0x1 vid:0x4300	
58_mesh_V4.1.0.0 pid:0x1 vid:0x4100  3. Choose phone loading method  tributor: □ ConnectedDevice ♥ Phone  Obtain the current device version iator Progress: 5. Start upgrade  get fw info start  get fw info start	58_mesh_V4.2.0.0 pid:0x1 vid:0x4200	
3. Choose phone loading method       Distributo         stributor:       ConnectedDevice       Phone         Obtain the current device version iator Progress:       Initiator P         tributor Progress       5. Start upgrade       Distributo         get fw info       start       get	258_mesh_V4.1.0.0 pid:0x1 vid:0x4100	
Ioading method       uributor:     ConnectedDevice       Obtain the current device version ator Progress:     Initiator P       uributor Progress     5、Start upgrade       get fw info     start	3、Choose phone	
tributor: ☐ ConnectedDevice ⊘ Phone Distributo Obtain the current device version ator Progress: tributor Progress 5、Start upgrade get fw info start g	loading method	
tributor:       ConnectedDevice          Phone        Distributor          Obtain the current device version        Initiator P          tributor Progress:          5、Start upgrade        Distributo          get fw info       start          get		
Obtain the current device version iator Progress:     Initiator Progress:       tributor Progress:     5、Start upgrade       get fw info     start		10
Obtain the current device version iator Progress:     Initiator P       tributor Progress     5、Start upgrade       get fw info     start		
ributor Progress 5、Start upgrade Distributo get fw info start gr	Obtain the current device vers	ion
get fw info start gr	ributor Progress 5、Start up;	grade
get winno		
	get tw into start	

Figure 33.40: iOS phone method upgrade steps & upgrade completion interface

# 33.5.4.2 Distributor: Verify and Apply Mode Upgrade (Directly Connected Nodes as Distributor Mode)

For Distributor, select connected device, FOr Apply Policy, select Verify and Apply method of upgrading, upload the firmware to the directly connected node through the mobile phone, and then distributes it to the target node through the directly connected node, and then automatically applies the new version after loading is completed.

Note:

Make sure that the DISTRIBUTOR\_UPDATE\_SERVER\_EN macro is enabled for the directly connected node.

Specific operational steps:

#### Android version:

At the APP home page, click on the bottom right Network – Mesh OTA according to the following steps to operate.

Mesh O	Ā	< Mesh OTA	
evices:	select >	Devices:	sel
		adr-0x0004 cid-1102 pid-0100 Fwld:0	)1004200
1、Select U	Jpgrade Device	state: Update Success additional: CPS changed, remote pv is no	ot support
2、sel	ect Upgrade File		
3、Choose Con	nected		
device loading r	nethod		
. Select			
erify and apply policy			
e error	>		
n version: nul		bin version: pid-01:00 vid-42:00	
stributor: 🥊 Phone 🧿	Connected Device	Distributor: OPhone OCon	nnected D
pply Policy: 🜔 Verify And Apply	Only	Apply Policy: O Verify And	O Ver On
itiate progress(%d): 5	Obtain the current device version	Initiate progress(100):	
stribute progress(%d): <sup>6</sup> 、	Start upgrade	Distribute progress(100):	
LE	~	update complete - recheck complete	
GET FIRMWARE ID	START		

Figure 33.41: Android verify and apply upgrade steps & upgrade completion interface

#### iOS version:

At the App home page, click on the lower right Setting – Mesh OTA according to the following steps.

K Mesh OTA		<	Mesh	ΟΤΑ	
Device list 1、Select Upgrade Device		Device list			
choose all		choose all			
adr:0x2 pid:0x1 vid:0x4100		adr:0x2 pid:0x1 v	id:0x4100		
adr:0x4 pid:0x1 vid:0x4100		adr:0x4 pid:0x1 v	id:0x4200 succ	ess	~
adr:0x6 pid:0x1 vid:0x4200		adr:0x6 pid:0x1 v	id:0x4200		
OTA file 2、 select Upgrade File		OTA file			
8258_mesh_V4.1.0.0_20230926 pid:0x1 Vic.9v4100		8258_mesh_V4.1	.0.0_20230926	pid:0x1 vid:0x4100	
8258_mesh_V4.2.0.0 pid:0x1 vid:0x4200		8258_me	Hi	ts	~
8258_mesh_V4.1.0.0 pid:0x1 vid:0x4100		N 8258_me	lesh ota finish,	success:1,fail:0	
3、Choose Connected device load 4、Select Verify a	ing method and apply policy		Do	ne	
Distributor: 🖂 ConnectedDerce 🗌 Phone		Distributor:	Connected	Device Phone	
Policy: VerifyOnly VerifyAndApply		Policy: 🗌 Ve	erifyOnly 🖂	VerifyAndApply	
Initiator Progress: 5, Obtain the o	current device version	Initiator Progre	ess: 100%		
Distributor Progress: 6、Start upgra	ade	Distributor Pro	gress: 100%		
get fw info start		get fw	info	start	





# 33.5.4.3 Distributor: Verify Only Mode Upgrade (Directly Connected Nodes as Distributor Mode)

For Distributor, choose the Verify only method to upgrade, upload firmware to the directly connected node through the mobile phone, and then distribute it to the target node through the directly connected node, and after the loading is completed, you need to reconnect the node with the APP before applying the new version.

#### Note:

Make sure that the DISTRIBUTOR\_UPDATE\_SERVER\_EN macro is enabled for the directly connected node.

#### Specific operational steps:

#### Android version:

At the APP home page, click on the bottom right Network – Mesh OTA according to the following steps.

< Mesh O	ТА	< Mesh OTA	
Devices:	select >	Devices:	se
1. Select		edit-0x0002 cid-1102 pid-0100 state: Update Success additional: CPS changed, remote pv is not sup	00
2、se 3、Choose Connected	lect Upgrade File		
device loading metho	d		
ïle error	>		
oin version: null	4、Select Verify and apply p	blicy bin version: pid-01:00 vid-42:00	
Distributor: 🔵 Phone (	Connected Device	Distributor: O Phone O Connecto	ed
Apply Policy: O Verify An	d Only	Apply Policy: Verify And	Ve
nitiate progress(%d): 5	Obtain the current device version	n Initiate progress(100):	
Distribute progress( /₀d): <sup>6</sup>	、Start upgrade	Distribute progress(100):	
DLE	~	update complete - recheck complete	
GET FIRMWARE ID	START		

Figure 33.43: Android verify only method upgrade steps & upgrade completion interface

#### iOS version:

Click Setting – Mesh OTA at the bottom right of APP homepage and follow the steps in the figure below.

#### Telink SIG Mesh SDK Developer Handbook

K Mesh OTA		K Mesh OTA
Device list 1、Select Upgrade Devi	ce	Device list
choose all		choose all
adr:0x2 pid:0x1 vid:0x4100		adr:0x2 pid:0x1 vid:0x4200 success
adr:0x4 pid:0x1 vid:0x4200		adr:0x4 pid:0x1 vid:0x4200
adr:0x6 pid:0x1 vid:0x4200 OTA file 2、 select Upgrade File		adr:0x6 pid:0x1 vid:0x4200 OTA file
8258_mesh_V4.1.0.0_20230926 pid:0x1 vio.2x4100		8258_mesh_V4.1.0.0_20230926 pid:0x1 vid:0x4100
8258_mesh_V4.2.0.0 pid:0x1 vid:0x4200		8258_me Hits
8258_mesh_V4.1.0.0 pid:0x1 vid:0x4100		Mesh ota finish, success:1,fail:0 8258_me
4、Select 3、Cl Verify and apply policy load	hoose Connected devic ing method	e Done
Distributor: 🖂 ConnectedDevice 🗌 Phon	e	Distributor: 🕢 ConnectedDevice 🗌 Phone
Policy: VerifyOnly 🗌 VerifyAndApply		Policy: VerifyOnly VerifyAndApply
Initiator Progress: 5, Obtain the	e current device versior	Initiator Progress: 100%
Distributor Progress: 6、Start upg	grade	Distributor Progress: 100%
get fw info start		get fw info start

Figure 33.44: Android verify only method upgrade steps & upgrade completion interface

#### 33.5.5 Private beacon

Using private beacon can set to send the specified broadcast after the node is networked, enable MD\_PRIVACY\_BEA and PRIVATE\_PROXY\_FUN\_EN in the mesh\_config.h file.

## 33.5.5.1 Config GATT Proxy

Open Config GATT Proxy alone and always send only Network ID broadcasts with light blue APP broadcast type 0x00.

< Private Beacon S	Setting
Enable GATT Proxy Enable Private GATT Proxy	
Enable Node Identity Enable Private Node Identity	
Enable Beacon Enable Private Beacon	





## 33.5.5.2 Private GATT Proxy

Open private GATT Proxy alone, Network ID is private, send encrypted processed Network ID with new Mac address, light blue APP broadcast type 0x02.

< Private Beacon	Setting
Enable GATT Proxy Enable Private GATT Proxy	
Enable Node Identity Enable Private Node Identity	
Enable Beacon Enable Private Beacon	



## 33.5.5.3 Config Node Identity

Open Config Node Identity individually, send Node Identity information within 60 seconds (light blue APP broadcast type 0x01), and automatically switch to send the default Network ID broadcast after 60 seconds (light blue APP broadcast type 0x00).

< Private Beacon Setting	Send node identity within 60 seconds
Enable GATT Proxy	<b>Mesh Proxy:</b> Identification type: Node Identity (0x01) Hash: 0xC131CC6807C45FA2 Random: 0x66AFB74530695FC8
Enable Node Identity   Enable Private Node Identity	Sand notwork ID in 60 seconds
	Send network ib in 60 seconds
Enable Beacon	Mesh Proxy: Identification type: Network ID (0x00) Network ID: 6E:78:C0:08:90:26:BD:FB



## 33.5.5.4 Private Node Identity

Open Private Node Identity individually, Node Identity is in private state, send encrypted Node Identity broadcast (light blue APP broadcast type 0x03) with new Mac address within 60 seconds, and automatically switch to send default Network ID broadcast (light blue APP broadcast type 0x00) after 60 seconds.

< Private Beacon	Setting
Enable GATT Proxy Enable Private GATT Proxy	
Enable Node Identity Enable Private Node Identity	
Enable Beacon Enable Private Beacon	



# 33.5.5.5 Config GATT Proxy + Config Node Identity

Open Config GATT Proxy and Config Node Identity at the same time, send Node Identity information within 60 seconds (light blue APP broadcast type is 0x01), and automatically switch to send the default Network ID broadcast after 60 seconds (light blue APP broadcast type is 0x00).

< Private Beacon S	Setting	Send node identity within 60 seconds
Enable GATT Proxy Enable Private GATT Proxy		<b>Mesh Proxy:</b> Identification type: Node Identity (0x01) Hash: 0x546BFF1D41878218 Random: 0xEEB50C28EE8878DA
Enable Node Identity Enable Private Node Identity		Sand notwork ID in 60 coconds
		Send network ID In 60 seconds
Enable Beacon Enable Private Beacon		<b>Mesh Proxy:</b> Identification type: Network ID (0x00) Network ID: 6E:78:C0:08:90:26:BD:FB
	,	

Figure 33.49: Config GATT proxy + Config node identity & broadcast type

# 33.5.5.6 Config GATT Proxy + Private Node Identity

At the same time, open Config GATT Proxy and Private Node Identity, the Node Identity is private, the encrypted Node Identity broadcast will be sent with the new Mac address within 60 seconds (light blue APP broadcast type 0x03), and after 60 seconds, it will automatically switch to send the default Network ID broadcast (light blue APP broadcast type 0x00).

< Private Beacon S	etting
Enable GATT Proxy Enable Private GATT Proxy	
Enable Node Identity Enable Private Node Identity	
Enable Beacon Enable Private Beacon	

#### Figure 33.50: Config GATT proxy + Private node identity & broadcast type

## 33.5.5.7 Private GATT Proxy + Config Node Identity

At the same time open Private GATT Proxy + Config Node Identity, within 60 seconds send Node Identity information (light blue APP broadcast type is 0x01), after 60 seconds Network ID is private, with the new Mac address after encrypted processing and then send the broadcast to the outside world, light blue APP broadcast type is 0x02.

< Private Beacon S	etting	Send private node identity within 60 seconds
Enable GATT Proxy Enable Private GATT Proxy		<b>Mesh Proxy:</b> Identification type: Unknown (3) Payload: 0xB0C06E0C6826C2FF3C2802 5EDCE230D1
Enable Node Identity Enable Private Node Identity		Send network ID in 60 seconds
Enable Beacon Enable Private Beacon		<b>Mesh Proxy:</b> Identification type: Network ID (0x00) Network ID: 6E:78:C0:08:90:26:BD:FB
10		

Figure 33.51: Private GATT proxy + Config node identity & broadcast type

#### 33.5.5.8 Private GATT Proxy + Private Node Identity

Open Private GATT Proxy + Private Node Identity at the same time, Node Identity and Network ID are both in private state, send encrypted Node Identity broadcast (light blue APP broadcast type 0x03) with new Mac address within 60 seconds, send encrypted Network ID broadcast (light blue APP broadcast type 0x02) with new Mac address after 60 seconds.

Private Beacon Setting	Send private node identity within 60 seco
inable GATT Proxy	<b>Mesh Proxy:</b> Identification type: Unknown (3) Payload: 0x65F7923E6550DE3201E1CE 19E4147752
nable Node Identity 🕖 💭	Sand private potwork ID within 60 second
	Send private network ib within to second
Enable Beacon	<b>Mesh Proxy:</b> Identification type: Unknown (2) Payload: 0x5DAB35671E8DAB7701E1CE 19E4147752

Figure 33.52: Private GATT proxy + Private node identity & broadcast state

# 33.5.5.9 Config Beacon

After opening Config Beacon, the node will send Non-Comnecable undrected Adv Packet of Secure Network Beacon type every 10 seconds, which can be viewed through light blue APP (whether the phone can receive the beacon packet when the node sends the beacon packet every 10 seconds depends on the Bluetooth refresh rate of the mobile phone), or you can view it through the packet grabbed by Netkey and APPkey through ellisys.

#### light blue APP beacon packet

mesh beacon: Beacon type: Secure Network (0x01) Key Refresh flag: false IV Update flag: IV Update active Network ID: 6E:78:C0:08:90:26:BD:FB IV Index: 1 Authentication value: 0x74DB8A889DE4411C

ellisvs heacon packet

Energy Overview Message Log		set reset an induit	gate 💌 🛶 🕛 🕅	arkers + qui up i	Thering. O	my 01.00.00.00.0	.0.50 (i	vointesoivai	ole), 66:6 👻 🕎			
							1 Þ 🗙	Details				
l: Single 🖌 All layers 🔸 🛹 👓 🎰 💡 👘	🕑 🖻 🦂 🎝 🔯 1 item disp	played		7 selections / 1	m 06s 🝸 🔍	🕭 🔹 Search	•   😫	➤ All fields	📄 🛅 Show in overview	v 🛛 Display 👻 🛐	Search	
		con Type 🛛 🗸	Time delta 🛛 🗸	RF Channel I $ \smallsetminus $	Source A ~	Time 🕴	~ ^	Name		Value	Dec	ł
Mesh Secure Network Beacon (ADV, Key Refresh	Flag=False, IV Update Flag=N. Sec	ure Network Beacon		37 (adv)		3.957 741 125			lvertiser Address	66:66:66:33:33:04		
Non-Connectable Undirected Adv Packet (66:	66:66:33:33:04, Mesh Beac Sec	ure Network Beacon		37 (adv)		3.957 741 125		🗏 🕂 A(	lvertising Data			
👪 Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon		37 (adv)		3.957 741 125			Mesh Beacon			
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 465 125	38 (adv)		3.958 206 250			Length	23	23	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 464 250	39 (adv)		3.958 670 500			Data Type	Mesh Beacon	43	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	10.987 813 625	37 (adv)		14.946 484 125			Beacon Type	Secure Network Beacon	1	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 465 000	38 (adv)		14.946 949 125		E	🐴 Flags			
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Seo	ure Network Beacon	0.000 464 250	39 (adv)		14.947 413 375			Key Refresh Flag	False	0	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	10.996 273 625	37 (adv)		25.943 687 000			IV Update Flag	Normal	0	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 465 125	38 (adv)		25.944 152 125			Reserved	0	0	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 464 625	39 (adv)		25.944 616 750			Network ID	0x6E78C0089026BDFB	7'960'323'484'3	
Non-Connectable Undirected Adv Packet (	66:66:66:33:33:04, Mesh B Seq	ure Network Beacon	11.010 113 875	37 (adv)		36.954 730 625			IV Index	0x0000001	1	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Seo	ure Network Beacon	0.000 465 000	38 (adv)		36.955 195 625			Authentication Value	0x74D88A889DE4411C	8'420'476'247'6	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Seo	ure Network Beacon	0.000 464 375	39 (adv)		36.955 660 000		4	Non-significant Part	0 bytes		
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	10.986 273 250	37 (adv)		47.941 933 250		9 C	RC	Valid	12'581'830	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 465 000	38 (adv)		47.942 398 250		<				
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 464 625	39 (adv)		47.942 862 875		Data				
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	11.004 839 500	37 (adv)		58.947 702 375		Data type:	Raw Data		Search	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 465 000	38 (adv)		58.948 167 375			0 1 2 3 4	56789AB	0123456789	P
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 464 250	39 (adv)		58.948 631 625		0x0000:	22 1E 04 33 33 0	6 66 66 17 2B 01 00	"33fff.+	έ.
Non-Connectable Undirected Adv Packet (	66:66:66:33:33:04, Mesh B Sec	ure Network Beacon	10.994 391 625	37 (adv)		69.943 023 250		0x000C:	6E 78 CO 08 90 2	6 BD FB 00 00 00 01	nx&	
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 465 000	38 (adv)		69.943 488 250		0x0018:	74 DB 8A 88 9D B	4 41 1C C6 FB BF	tA	•
Non-Connectable Undirected Adv Packet (	56:66:66:33:33:04, Mesh B Sec	ure Network Beacon	0.000 464 375	39 (adv)		69.943 952 625	~					
							>					
							a 🗙					
	0.45	I Lordet Loren Lor		Let let								

Figure 33.53: Light blue APP & ellisys receive beacon packets after opening beacon

# 33.5.5.10 Private Beacon

To open Private Beacon, you need to open Private GATT Proxy at the same time, the node will change the Mac address, and send Reserved (0x02) Beacon type Non-Comnecable undrected every 10 seconds, which can be viewed through light blue APP (whether the phone can receive the beacon packet when the node sends it every 10 seconds depends on the Bluetooth refresh rate of the phone), and can also be viewed through Netkey and APPkey by grabbing packets through ellisys.

#### Open private beacon and private proxy

< Priv	vate Beacon S	etting
Enable GATT F	Proxy	
Enable Private	GATT Proxy	
Enable Node I	dentity	
Enable Private	Node Identity	
Enable Beacor	1	
Enable Private	Beacon	

# light blue app beacon package

mesh beacon: Beacon type: Unknown (2) Payload: 0xE216A92B900FEC31E2CCA5 9A46FEB8BF5AE49ECDD742EA46CA27

ellisys beacor	n package		
- APP说明文档 private beacon .btt - Ellisvs Bluetooth Analvzer		- a	×
File View Lavout Search Record Tools Help		🗖 123 🗐 Analysis 📑 Add	
🖹 🗃 🗃 🕅 🖒 🕨 Record 👻 🗏 Stop 💷 Restart 👼 Save & Continue 🦙 🕮 🖙 Mavigate 👻 🖫 Markers 🕶 🖓 👘 Hereir	ng: Only 13:57:8D:6C:87:1C (Non-Resolvable)	a 66:6 ▼ 😚	_
	4.6	× Details	9 X
Protocol: Single + All lavers + + + + + + + + + + + + + + + + + + +	🍸 🔍 🦣 🗸 Search , 🛛	X All fields R Show in overview Display - Da	
X Show: Item x = "Non-Connectable Undirected" x		Name Value	E De A
Trans Trans L. Research Trans L. Herdesneded better	Time delta	Overall Static Title     1/8.015 518 8/5	1/8
		<ul> <li>Overal Duration</li> <li>1.37 ms</li> </ul>	1'36
• VINT-CONNECTABLE UNITABLE (1):57:301-56:79:10 ( RESERVED (XXZ)) = 21 64 02 59 00 FE C 31 22 CC AS 94 40 FE 03 F5 A5 49 50 CD 74 35 A4 50 A	27 11 002 526 500 178 615 518 875	Ouration 1.369 ms	1'36
* V Non-Connectable Undirected (13:57:80:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 10.997 868 000 189.613 386 875	. Devices	<b>.</b>
* 3 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 10.999 743 500 200.613 130 375	E At Mesh Beacon	
* 🚼 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE 88 BF 5A E4 9E CD D7 42 EA 46 CA	27 11.001 635 625 211.614 766 000	Data Type Mesh Beacon	43
* 🐉 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 11.000 188 875 222.614 954 875	A Beacon Type Reserved (0x02)	2
3 3 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	27 11.000 793 500 233.615 748 375	A Undecoded bytes E2 16 A9 2B 90 0F EC	3
3 🕃 🐉 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 10.998 062 000 244.613 810 375	* Indication	
Constant in the second se	27 10.998 844 875 255.612 655 250		
* J Non-Connectable Undirected (13:57:80:6C:87:1C ( Reserved (0x02)) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 10.999 453 875 266.612 109 125	<sup>(1)</sup> <sup></sup>	
a Non-connectable Undirected (13:57/80/56.03/11C)	2/ 11.001 /91 250 2/7.613 900 3/5	😑 🔩 Sniffer Radio	
a Non-connectable Undirected (1357:80:06:07/11C)	27 11.002 866 250 288.616 766 625	RSSI -21.5 dBm	-21
*** Non-Connectable Indirected (13:57:60-66:97:1C) Reserved (0x02) E2 16 A2 29 00 FE 31 E2 CCA 3 A4 0FE BB FE 34 E4 0FE DD 74 2E 44 0CU **********************************	27 11.002.473.250 310.618.509.250	RX Quality High	-6 ~
Non-Connectable Indirected (13:57:80:60:87:10 ( Reserved (0x2))     F 16 A9 20 90 0F FC 31 E2 CC 39 A 45 FF 88 F5 A F4 9F CD 77 42 FA 45 CA	27 10.992 553 125 321.611 152 375	<	>
Non-Connectable Undirected (13:57:80:56:87:1C ( Reserved (0x02)     E 2 16 49 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 # 5A E4 9E CD 77 42 E4 46 CA	27 11.008 404 875 332.619 557 250	Data	9 <b>X</b>
* 37 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE 88 8F 5A E4 9E CD D7 42 EA 46 CA	27 10.997 995 250 343.617 552 500	Data type: Raw Data • Search	-
* 🐉 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 10.998 199 625 354.615 752 125	0 1 2 3 4 5 6 7 01234567	
* 🕄 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE 88 BF 5A E4 9E CD D7 42 EA 46 CA	27 11.003 501 500 365.619 253 625	0x0000: 62 23 1C 87 6C 8D 57 13 b1.W.	
* 🐉 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 10.991 083 625 376.610 337 250	0x0010: 0F EC 31 E2 CC A5 9A 461F	
* 😲 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 11.008 409 875 387.618 747 125	0x0018: FE B8 BF 5A E4 9E CD D7Z	
* 🕑 Non-Connectable Undirected (13:57:8D:6C:87:1C ( Reserved (0x02) E2 16 A9 28 90 0F EC 31 E2 CC A5 9A 46 FE B8 BF 5A E4 9E CD D7 42 EA 46 CA	27 10.999 719 250 398.618 466 375	V 0x0020: 42 EA 46 CA 27 F0 91 3F B.F.'? 0x0020;	
	>		
Timing	ģ	×	
🔪 🖓 🔍 🔟 🛣 v origin: 178.52 s v span: 0.16 s v Bluetooth v WiFi HCI WCI WPAN Logic Misc v Display v Lo	əgic inputs 🛛 🖓 🐪		
aluetooth 🔅			
66:66:33:33:04			
13:57:80:6C:87:1C (NonR			
1 Non-Connectable Und.			
50 s 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.06 0.09 170.60 s 0.01 0.02 0.03 0.04	0.05 0.06 0.07 0.08		
Zoom bar		A that Granded A Granded 101 parts	
ming whenets and audo		Mesh Security Security all Data	
Keady			P 022 45

#### Figure 33.54: Open private beacon after light blue APP & ellisys receive beacon packets

# 33.5.5.11 Beacon + Private Beacon

Open Beacon and Private Beacon at the same time (Private GATT Proxy needs to be opened at the same time), the node will send Non-Comnecable undrected Adv Packet of Secure Network Beacon type every 10 seconds with the original Mac address, at the same time, the node will change the Mac address and send a Non-Comnecable undrected Adv Packet of type Reserved (0x02) Beacon every 10 seconds, it can be viewed by light blue APP (whether the phone can receive the beacon packet when the node sends it every 10 seconds depends on the Bluetooth refresh rate of the phone), and it can also be viewed by Netkey and APPkey through ellisys packet grabbing.

# Open beacon and private beacon

< Private Beacon S	etting
Enable GATT Proxy Enable Private GATT Proxy	
Enable Node Identity Enable Private Node Identity	
Enable Beacon Enable Private Beacon	•

## light blue app beacon & private beacon package

#### mesh beacon:

Beacon type: Secure Network (0x01) Key Refresh flag: false IV Update flag: IV Update active Network ID: 6E:78:C0:08:90:26:BD:FB IV Index: 1 Authentication value: 0x74DB8A889DE4411C

#### mesh beacon:

Beacon type: Unknown (2) Payload: 0x068AB85EDC9ADE12CAB3E 5081093141542E86278D205D050AC38

#### Figure 33.55: Beacon packets received by light blue APP after opening beacon + private beacon

relink semiconductor

|  
  | embys seace  
   |   
   |  | ~~~B  | •  |   
  |  
   |   |  |
---
--
---	--
--
--|---|--|
| 🛞 Recording from BTR1-23087 - Ellisys Bluetooth Analyzer   
  |  
   |   
   |  |   |  |   
  |  
   | - 0   | ×  |
| <u>File View Layout Search Record Tools H</u> elp  
  |  
   |   
   |  |   |  |   
  | 123  
   | 🛯 🗔 Analysis 🛛 📑 Ad   | d  |
| 🗋 🚔 🛃 🎇 👫 🕨 Record 👻 🖬 Stop 🖬 Restart 😼 Save & <u>C</u> or   
  | ntinue 🏾 htter ht  
   | 🝷 🖳 💼 Markers 👻 💭 🔩 🖓 Filterin  | ng: Only 02:FE:32:C8  
  | :EF:A8 (Non-Resolv  | able), 66  | :66 👻 🌏   
  |  |   
   |  |
| Low Energy Overview Message Log  
  |  
   |   
   |  |   | 4 • ×  | Details   
  |  
   |   | φ×   |
| Protocol: Single - All layers + 🕫 📾 📽 🕼 😒 🦻 🏭  
  | 1 item displayed   
   | 7 sele  
   | ctions / 1m 06s ¥  | Q 🧐 - Search  | • ( <u>*</u>   | ➤ All field   
  | s 💾 Show in overview   
   | Display 🔻 📄   |  |
| Item 4 Verse Beacon Type   
  | · · · · Network ID · · ·   
   | Authentication Value V IV Index V Tir   
   | ne delta V RF C  | V Time  | - ~ ^  | Name  
  |  
   | Value   | Der ^  |
| Mesn Secure Network Beacon (ADV, Key Refresh Flag=r., Secure Netw     Secure Network Beacon (ADV, Key Refresh Flag=r., Secure Netw     Secure Network Beacon (ADV, Reg Refresh Flag=r., Secure Netw  
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001<br>0x74DB8A889DE4411C 0x00000001  
   | 37 (ad<br>37 (ad   | <ul> <li>v) 2.544 002 625</li> <li>v) 2.544 002 625</li> </ul>  |  |   
  | dvertiser Address<br>dvertising Data   
   | 66:66:66:33:33:04   |  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001   
   | 37 (ad   | v) 2.544 002 625  |  |   
  | 3 Mesh Beacon  
   |   |  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 465 125 38 (ad   | v) 2.544 467 750  |  |   
  | Length   
   | 23  | 23   |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 464 125 39 (ad   | v) 2.544 931 875  | _  |   
  | Data Type  
   | Mesh Beacon   | 43   |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 1   
   | 000 465 125 38 (ad   | <ul> <li>v) 13.548 912 750</li> <li>v) 13.547 377 875</li> </ul>  |  | 1   
  | Beacon Type Hags   
   | Secure Network Beacon   | · •  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 464 250 39 (ad   | v) 13.547 842 125   |  |   
  | Key Refresh Flag   
   | False   | 0  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 10  
   | 0.997 427 625 37 (ad   | v) 24.545 269 750   |  |   
  | IV Update Flag   
   | Normal  | 0  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 465 125 38 (ad   | v) 24.545 734 875   |  |   
  | Reserved   
   | 0<br>0x6E78C0089026BDEB   | 0  |
| Non-Connectable Undrected Adv Packet (66:66 Secure Netw  
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74D88A889DE4411C 0x00000001 0.  
   | 000 464 250 39 (ad<br>999 980 500 37 (ad   | v) 24.546 199 125 v) 35 546 179 625   | -  |   
  | IV Index   
   | 0x00000001  | 1  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 465 000 38 (ad   | <ul> <li>v) 35.546 644 625</li> </ul>   |  |   
  | Authentication Value   
   | 0x74DB8A889DE44110  | 8'42   |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 464 250 39 (ad   | v) 35.547 108 875   |  |   
  | Non-significant Part   
   | 0 bytes   | 10/1   |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 1   
   | ).999 656 625 37 (ad   | v) 46.546 765 500   |  | < 4   
  | .RC  
   | Vaid  | >  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 465 125 38 (ad<br>000 464 125 39 (ad   | v) 46.547 230 625 v) 46.547 694 750   |  | Data  
  |  
   |   | a x  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 10  
   | 000 404 125 35 (ad<br>0.994 337 500 37 (ad   | v) 57.542 032 250   | -  | Data type:  
  | Raw Data   
   | Search  |  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 465 125 38 (ad   | v) 57.542 497 375   |  |   
  | 0 1 2 3 4 5  
   | 6 7 0123456   | 7  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 464 750 39 (ad   | v) 57.542 962 125   | _  | 0x0000:   
  | 22 1E 04 33 33 66<br>17 2B 01 00 6E 78   
   | 66 66 "33ff:  | f  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw<br>(19) Non-Connectable Undirected Adv Packet (66:66 Secure Network)  
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 10  
   | 0.997 250 500 37 (ad   | v) 68.540 212 625 v) 68 540 677 750   |  | 0x0010:   
  | 90 26 BD FB 00 00  
   | 00 01 .6  |  |
| Non-Connectable Undirected Adv Packet (66:66 Secure Netw   
  | ork Beacon 0x6E78C0089026BDFB  
   | 0x74DB8A889DE4411C 0x00000001 0.  
   | 000 463 123 38 (ad<br>000 464 750 39 (ad   | <ul> <li>v) 68.540 677 730</li> <li>v) 68.541 142 500</li> </ul>  | ~  | 0x0018:   
  | 74 DB 8A 88 9D E4  
   | 41 1C tA  | •  |
| <  
  |  
   |   
   |  | .,  | >  | 020020.   
  | CO ID DI   
   |   |  |
| Timing   
  |  
   |   
   |  |   | 9 <b>X</b>   |   
  |  
   |   |  |
| 🖤 🔍 🔳 🚡 🗸 origin: 68.44 s 🔹 🔹 span: 0.16 s 🔹   
  | Bluetooth 👻 WiFi HCI WCI   
   | WPAN Logic Misc - Display - Lo  
   | gic inputs   | <b></b>   | }i <b>* </b> + -   |   
  |  
   |   |  |
| M  
  |  
   |   
   | •  |   | ^  |   
  |  
   |   |  |
| 66:66:33:33:04   
  |  
   |   
   |  |   | - 14   |   
  |  
   |   |  |
| Undetermined   
  |  
   |   
   |  |   |  |   
  |  
   |   |  |
| ( (re)   
  |  
   |   
   |  |   |  |   
  |  
   |   |  |
|  
  | 001 002  
   | 0.03 0.04 0.05 0.06   
   | 0.07 0.08  | 0.00  | -  |   
  |  
   |   |  |
| Zoom bar   
  | 68.50 s  
   |   
   | 0.07 0.00  | 68.60 s   | ~  |   
  |  
   |   |  |
| 🛀 Timing 😴 Piconets 🎝 Audio  
  |  
   |   
   |  |   |  | 👌 Mesh Sec  
  | curity 👌 Security 👬 Data   
   |   |  |
| Recording data   
  |  
   |   
   |  |   |  |   
  |  
   | ₩ ,   | 🖣 🖽 👪  |
|  
  |  
   |   
   |  |   |  |   
  |  
   |   |  |
|  
  |  
   |   
   |  |   |  |   
  |  
   |   |  |
| Percenting from PTP1 22007 Elling Physics th Analyzer  
  |  
   |   
   |  |   |  |   
  |  
   |   | . ~  |
| Recording from BTR1-23087 - Ellisys Bluetooth Analyzer     Ello View Larout Coards Bacard Tools Halp   
  |  
   |   
   |  |   |  |   
  | <b>I</b> 123   
   | – O   | x a  |
| Recording from BTR1-23087 - Ellisys Bluetooth Analyzer     Elle View Layout Search Record Tools Help     Search Score Store Record For St  
  | ations 🖿 📲 = 🐄 🕅 Navigata  
   | - 🖾 lê Markara - 🖉 🕞 📝 Eilteoi  |   
  | EE:AR (Non Parah)   | able) 66   |   
  | 123  | - 🛛   
   | u ×<br>d   |
| Image: Recording from BTR1-23087 - Ellisys Bluetooth Analyzer         File       View       Layout       Search       Record       Iools       Help         Image: Im  
   | ntinue  🍟 🕶 🗱 💭 Navigate  
  | 🕶 🖏 💼 Markers 👻 🛹 👒 🏹 Filterin  | ng: Only 02:FE:32:C8   
   | :EF:A8 (Non-Resolv  | able), 66  | :66 🔻 🜏  
   | 122  | - 🖸  
  | u x<br>d   |
| Recording from BTR1-23087 - Ellisys Bluetooth Analyzer Ele View Layout Search Record Iools Help      Search Record - Stop Restart & Save & Cor      Low Energy Overview Message Log      Protropic Single - All Navec + All Protoce - Single - All Navec - Al  
  | ntinue 🛛 🎲 🖛 🛣 💭 Navigate<br>28 items kent. 7 filtered   
   | 🕶 🖏 🕼 Markers 💌 🐢 🎭 🏹 Filterin  | 1g: Only 02:FE:32:C8  
  | :EF:A8 (Non-Resolv  | able), 66<br>4 ▶ <b>×</b>  | i:66 ▼ 🕢  
  | E Chow in superior   | - a   
   | ₽ ×<br>d   |
| Image: Second in the second secon  
   | ntinue 뉟 🐄 🛩 🎇 🚚 Navigate<br>28 items kept, 7 filtered  
  | 💌 🖏 🌔 Markers 🕶 🥔 🍕 Filterin  | ng: Only 02:FE:32:C8<br>Y  
   | EF:A8 (Non-Resolv   | able), 66<br>∢ ▶ ★<br>- 《≩   | c:66 ▼<br>Details<br>¥ All field   
   | s 🖹 Show in overview   | - C  
  | 0 X<br>d /   |
| Image: Second in the secon   
  | ntinue 🔭 🐨 🗸 🎲 Navigate<br>28 items kept, 7 filtered   
   | 🕶 🖏 🍋 Markers 🕶 📣 🖓 Filterin  | rg: Only 02:FE:32:C8  
  | EF:A8 (Non-Resolv   | able), 66<br>4 ▶ <b>×</b><br>- (€ <u>₹</u>   | i:66 •<br>Details<br>Vall field<br>Name<br>Adverti  
  | s Show in overview   | - C<br>Analysis 🗈 Ad<br>Display 🕶 🏹<br>Value  
   | 0 ×<br>d /<br>0 ×  |
| Recording from BTR1-23087 - Ellisys Bluetooth Analyzer  File View Layout Search Record Iools Help   Vew Layout Search Record > Stop I Restart > Save & Cor  Low Energy Overview / Message Log  Protocol: Single ~ All layers + e* = * © © ? A 7 0  × Show: Item = "Yon-Connectable" ×  Item  	 Beconvertable Indexnet (0015:270-2015:56)  Records and 10015:270-2015:560  Records and 10015:270-2015:560  Records and 10015:270-2015:560  Records and 10015:270-2015:560  Records and 10015  Records and 10015  Records and 1001  Records and  
  | ntinue 🔭 🐨 🕶 📰 🛹 Navigate<br>28 items kept, 7 filtered<br>V Undecoded bytes  
   | • 15 20 BERS 56 00 AS ON 56 75 75 48 56 55  | rg: Only 02:FE:32:C8  
  | EF:A8 (Non-Resolv<br>Q 3 - Search<br>Time 1<br>2 552 487 635  | able), 66<br>4 ▶ ×<br>- (€<br>+ ∽ ^  | i:66 ▼<br>Detais<br>¥ All field<br>Name<br>Adverts  
  | s Show in overview   | - C   
   | 0 ×<br>d<br>0 ×  |
| Recording from BTR1-23087 - Ellisys Bluetooth Analyzer  Elle View Layout Search Record Tools Help  View Layout Search Record - Broom Restart in Save & Con  Low Energy Overview Message tog  Protocol: Single - All Layers + et - O O O O O O O O O O O O O O O O O O  
  | ntinue 📘 🎲 📲 💭 Navigate<br>28 items kept, 7 filtered<br>- Undecoded bytes<br>0. C5 4 C6 OF B 10 C5 FE B 80 10 03 3A<br>0. C5 4 C6 OF B 10 FE FE B 91 00 33 A   
   |   
   | rg: Only 02:FE:32:C8   | EF:A8 (Non-Resolv<br>Q 3 - Search<br>Time 1<br>2.553 487 625<br>12.555 981 125  | able), 66<br>4 ▷ X<br>- (¥<br>+ ✓ ^  | i:66 ▼ 《<br>Detais<br>All field<br>Name<br>Advertis<br>M Adv  
  | s Show in overview   
   | - □<br>Analysis Ad<br>Display - □<br>Value  | 0 ×<br>d<br>9 ×<br>Des   |
| Image: Second in the second secon  
   | ttinue № 1  
  | ▼ ■ ○ Markers ▼ ● ● ■ ■ Filterin<br>SS 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AE 0<br>S9 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AE 0<br>S9 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AE 0   | ag: Only 02:FE:32:C8<br>Time deta<br>A<br>11.002 493 500<br>A 10.998 712 625   
   | EF:A8 (Non-Resolv<br>Q 3 - Search<br>Time 1<br>2.553 487 625<br>13.555 981 125<br>24.554 693 750  | able), 66  | i:66 ▼<br>Detais<br>¥ All field<br>Name<br>Advertis<br>B 22 Adv<br>↓ 1   
   | s Show in overview<br>ement<br>ertisement<br>ype   | - □<br>Analysis 計Ad<br>Display → La<br>Value   
  | 4 ×<br>0 ×<br>0 ×<br>0 ×<br>0 ×  |
| Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Jools Help I was a start of the second a start of the second secon   
  | ntinue 1 28 i cm 2 28 i cm Navigate<br>28 items kept, 7 filtered<br>∨ Undecode bytes<br>20 C5 4 C 60 F8 10 5F EE 80 10 03 3A<br>30 C5 4 C 60 F8 10 5F EE 80 10 03 3A<br>30 C5 4 C 60 F8 10 5F EE 80 10 03 3A   
   | <ul> <li>Markers</li> <li>Markers</li></ul>  | g: Only 02:FE:32:C8<br>♀<br>Time deta ↓<br>A<br>A<br>11.002.493 500<br>4 10.999 712 625<br>4 11.002 383 750  
   | EF:A8 (Non-Resolv<br>Q 2 - Search<br>Tme 1<br>2.553 487 625<br>13.555 981 125<br>24.554 687 591 125<br>24.554 697 500   | able), 66  | i:66 ▼<br>Detais<br>¥ All field<br>Name<br>Advertis<br>Ø Advertis<br>Ø 1<br>Ø 1<br>Ø 1<br>Ø 1<br>Ø 1<br>Ø 1<br>Ø 1<br>Ø 1  
   | s Show in overview ement ertisement ype contrauty  | Analysis Ad   | U X<br>d<br>9 X<br>2<br>0<br>0<br>0  
   |
| Recording from BTR1-23067 - Ellisys Bluetooth Analyzer<br>Elle View Layout Search Record Iools Help<br>View Layout Search Record Iools Help<br>Towe Energy Overview Message Log<br>Protocol: Single - Alllayers + et al. In the Search of the Search   
  | Itimu         Image: Imag  
   | <ul> <li>■ 10 Markers &lt; 20 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</li></ul>   | g: Only 02:FE:32:C8<br>✓<br>Time delta ✓<br>A<br>11.002 493 500<br>A 11.0998 712 625<br>A 11.0998 712 625<br>A 11.0998 726 550<br>A 10.3999 236 500  | EF:A8 (Non-Resolv<br>Q 2 - Search<br>Time 1<br>2.553 487 625<br>13.555 981 125<br>24.554 693 750<br>35.557 077 500<br>36.555 314 000   
  | able), 66  | x66 ▼ 《<br>Detais<br>▼ All field<br>Name<br>≈ Adverts<br>■ ¾ Adv<br>1<br>0<br>0<br>0<br>0   
  | Show in overview ement ertisement ype contrusty ming d Owerd Start Time  | Analysis Ad     Display      Vaue     Non-Connectable Under     Start     13 555 981 125  | U X<br>d<br>Q x<br>Det ^<br>ected 9<br>0  
  |
| Recording from BTR1-23087 - Ellisys Bluetooth Analyzer      File View Layout Search Record Jools Help      Search Record Jools      Search Record Record Record Jools      Search Record Record Jool   
   | Ifiliare         Image  
  |   | Gordy 02:FE:32:C8  
   | EF:A8 (Non-Resolv<br>C 3 + Search<br>Time 1<br>2.553 487 625<br>13.555 901 125<br>24.554 637 500<br>35.557 007 500<br>46.555 314 000<br>46.555 314 000<br>57.550 608 625<br>C 905 00 86 625   | able), 66  | C66 ▼<br>Detais<br>X All field<br>Name<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Advertis<br>Ad   
   | s Show in overview<br>ement<br>ertisement<br>ype<br>ontrauty<br>iming<br>0 Overal Duration   | Analysis Ad     Analysis Ad     Display      Analysis Ad     Non-Connectable Under     Start     13.555 981 125     1.37 ms   | <ul> <li>A x</li> <li>A x</li></ul>  
   |
| Recording from BTR1-23067 - Ellisys Bluetooth Analyzer      File View Layout Search Becord Jools Help      Search Becord Jools Help      Search Joons      Search Joons Help   | ttinue   | Markers      Markers      Part  | y<br>y<br>y<br>y<br>y<br>y<br>y<br>y<br>y<br>y<br>y<br>y<br>y<br>y   | EF:A8 (Non-Resolv<br>C 2 - Search<br>Time 1<br>2.553 487 625<br>13555 981 125<br>24554 637 50<br>35.557 077 500<br>46.556 314 000<br>46.556 314 000<br>46.556 314 000<br>46.556 314 000<br>46.556 319 500   | able), 66  | C66 ▼<br>Detais<br>× All field<br>Name<br>× Advertis<br>3  | s Show in overview<br>ement<br>eritsement<br>ype<br>contructy<br>imag<br>> Overal Start Time<br>> Overal Duration<br>> Overal Duration   | - C<br>Display Analysis Ad<br>Display A<br>Value<br>Non-Connectable Under<br>Start<br>13.555 981 125<br>1.37 ms<br>1.369 ms   | <ul> <li>A x</li> <li>A x</li></ul>  |
| Recording from BTR1-23087 - Ellisys Bluetooth Analyzer<br>Elle View Layout Search Record Tools Help<br>View Layout Search Record - State Search Search Search<br>Internet Search Search Search Search Search Search Search Search<br>Protocol: Single - All Layers + + + + + + + + + + + + + + + + + + +   
  | ntinue   
   | Comparison of the second  | g: Only 02:FE:32:C8<br>Time deta<br>1.0099 712 635<br>4.10.099 712 635<br>4.10.099 712 635<br>4.10.099 236 500<br>4.10.999 236 500<br>4.10.999 236 500<br>4.10.999 236 500<br>4.10.999 236 507<br>4.10.999 210 125<br>4.10.999 120 125  
  | EF:A8 (Non-Resolv<br>Q 2 - Search<br>Tme 2<br>2.553 487 625<br>13555 981 125<br>24.554 693 750<br>46.556 314 000<br>57.550 608 625<br>68.550 159 500<br>79.548 329 625<br>68.551 1250   | able), 66  | ic66 ♥ ④<br>Detais<br>♥ All field<br>Name<br>Advertis<br>Adv<br>↓ 1<br>↓ 1  
  | s Show in overview<br>ement<br>eritement<br>Vpe<br>orothuty<br>ming<br>9 Overal Start Time<br>9 Overal Start Time<br>9 Overal Start Time<br>9 Overal Start Time  | Analysis Ad     Analysis Ad     Display      Au   | 4 × 4   
  |
Recording from BTR1-23067 - Ellisys Bluetooth Analyzer File View Layout Search Record Jools Help File View Layout Search Record Jools Help File View Layout Resage to Protocol Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Allyzers + et al. State Search Control Single - Non-Connectable Underected (02:FE332CBEFA8 ( Reserved (0x0) = 3; Non-Connectable Underected (02:FE332CBEFA8 ( Reserved	Intinue         Image: Ima	<ul> <li>Markers</li> <li>Markers</li></ul>	g: Only 02:FE:32:C8 ✓ ✓ Time deta ✓ 4 4 11.002 493 500 11.002 493 500 11.0098 712 625 A 11.0098 726 507 10.0994 294 627 A 10.999 226 507 A 10.999 226 507 A 10.999 200 125 A 10.998 200 125 A 10.998 1567 000	EF:A8 (Non-Resolv Q 2 - Search Tme 2.553 487 625 13.555 981 125 24.554 693 750 35.557 077 500 46.556 314 000 77.550 608 625 66.550 159 500 79.548 359 625 90.553 181 250 10.548 348 250	able), 66	<ul> <li>i:66 ▼ Q</li> <li>Detais</li> <li>X All field</li> <li>Name</li> <li>Advertis</li> <li>2 Advertis</li> <li>3 Adv</li> <li>3 Advertis</li> <li>4 Advertis&lt;</li></ul>	S Show in overview ement ertsement ype overal Start Time Overal Start Time Overal Start Time Overal Start Time Evercise Advertiser	Analysis Ad Display - Analysis Ad Display - Analysis Ad Non-Connectable Undre Start 1.355 981 125 1.37 ms 1.369 ms 02:FE:32:C8:EF.AS (No	9 × 9 × 0 De(^ 13% 1'36 1'36 0'
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Jools Help View Layout Search Record Jools Help View Entry VoerNew Message top Protocol: Single      All Layers      +	Number         No.         No.<	Comparing the second seco	Ig: Only 02:FE:32:C8 Time deta Time deta A 10:090 712 625 A 10:099 712 625 A 10:099 236 500 10:099 236 500 10:099 236 500 10:099 236 500 10:099 550 875 A 10:099 550 875 A 10:090 550 8	EF:A8 (Non-Resolv Q 2 - Search Tme 2 2553 487 625 24.554 693 750 24.555 801 125 24.555 801 125 24.554 693 750 46.555 314 005 57.550 686 625 68.555 159 500 79.548 356 625 90.553 181 250 101.548 348 250 101.12.550 409 625	able), 66	i:66 • Q Detais All field Name Advertis Advertis Advertis Q	s Show in overview ement ertisement ype ontruty yourda Start Time > Overal Start Time	− □     □ Analysis Ad     □ Analysis     □ An	<ul> <li>x</li> <li>x</li> <li>z</li> <li>z</li> <li>be</li> <li>De</li> <li>13*</li> <li>1'3*</li> <li>1'3*</li> <li>1'3*</li> <li>1'3*</li> <li>1'3*</li> </ul>
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help View Layout Search Record Tools Help View Layout Search Record Tools Help View Entry VoerNew Messae Song Tools Help Protocol: Single      All layers      +      e      =	ntinue	Si Ale 105 56 9A AE 00 F6 75 C6 48 56 AE 05 56 9A AE 00 F6 75 C6 48 56 AE 05 56 9A AE 00 F6 75 C6 48 56 AE 05 56 9A AE 00 F6 75 C6 48 56 AE 05 56 9A AE 00 F6 75 C6 48 56 AE 05 56 9A AE 00 F6 75 C6 48 56 AE 05 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 67 55 56 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 56 95 AA E 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 53 AB 60 55 9A AE 00 F6 75 C6 48 56 AE 05 55 9A AE 00 F6 75 C6 48 56 AE 05 55 9A AE 00 F6 75 C6 48 56 AE 05 55 9A AE 00 F6 75 C6 48 56 AE 05 75 76 76 76	g: Only 02:FE:32:C8 Time deta 1.0029 372 635 4.10.099 712 635 4.10.099 712 635 4.10.099 226 550 4.10.999 226 550 4.10.999 236 550 4.10.999 250 6875 4.10.999 150 6875 4.10.999 150 6875 4.10.099 150 6875 4.10.0	EF:A8 (Non-Resolv Q 2 - Search Tme 2 2.553 487 625 13555 941 125 24.554 693 750 46.556 314 000 57.550 608 625 68.550 159 500 79.548 339 625 101.548 348 250 101.548 348 250 101.5557 496 625 122.557 496 625	able), 66	<ul> <li>ic66 ▼ </li> <li>Detais</li> <li>✓ All field</li> <li>Name</li> <li>☆ Advertis</li> <li>※ Adv</li> <li>♥ 1</li> <li>♥ 1<td>s Show in overview ement erisement Vype orden Start Time &gt; Overal Start Time</td><td>Analysis Ad     Analysis Ad     Display      Au     Analysis Ad     Display      Au     Au</td><td><ul> <li>x</li> <li>4</li> <li>z</li> <li>be ^</li> <li>De ^</li> <li>13%</li> <li>13%</li> <li>13%</li> <li>13%</li> <li>13%</li> <li>13%</li> </ul></td></li></ul>	s Show in overview ement erisement Vype orden Start Time > Overal Start Time	Analysis Ad     Analysis Ad     Display      Au     Analysis Ad     Display      Au	<ul> <li>x</li> <li>4</li> <li>z</li> <li>be ^</li> <li>De ^</li> <li>13%</li> <li>13%</li> <li>13%</li> <li>13%</li> <li>13%</li> <li>13%</li> </ul>
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Jayout Search Record Jools Help File View Jayout Search Record Jools Help File View Jayout Search Record Jools Help Fortocols Single - All Jayers + et al. Stop III Restart III Save & Cord Fortocols Single - All Jayers + et al. Stop III Restart III Save & Cord Show: Rem × = 'Iton-Connectable' × Item T V Beacon Type Item T V Beacon Type Item T V Beacon Type Item Connectable Understed (02:FE332C8:FFA6 ( Reserved (00:0) Item Connectable Understed (02:FE332C	Intimue         Image: Ima	Sta BF 05 56 9A AE 00 F6 75 C6 4B 56 AE 0 55 3A BF 05 56 9A AE 00 F6 75 C6 4B 56 AE 0 55 3A BF 05 56 9A AE 00 F6 75 C6 4B 56 AE 0 55 3A BF 05 56 9A AE 00 F6 75 C6 4B 56 AC 0 55 3A BF 05 56 9A AE	g: Only 02:FE32:C8 ✓ ✓ Time deta ✓ × 11.002 493 500 × 10.099 712 625 × 10.099 236 500 × 10.999 520 625 × 10.999 200 821 625 × 10.999 200 821 625 × 10.099 201 625 × 10.099 216 625 × 10.099 5167 000 × 10.002 61375 × 10.099 170 625 × 10.099 170 625 × 10.099 170 625 × 10.099 216 700 × 10.002 61375 × 10.099 170 625 × 10.099 170 625 × 10.091 10.095 105 × 10.001 10.095 105 × 10.001 10.055 105 × 10.055	EF:A8 (Non-Resolv Q	able), 66 4 ♦ x • 1 € # > ^	C66 ↓ Detais S All field Name Advertie A	S Show in overview ement ement emet of the secon of the	Analysis Ad     Analysis Ad     Display -      Analysis Ad     Display -      Analysis     Start	9     ×       9     ×       0     0       13%     13%       13%     13%       8     2       2     ×
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Jools Help View Layout Search Record Jools Help View Layout Search Record Jools Help View Entry Vorview Message top Protocol: Single      All Layers      +	ttinue	Markers	rg: Only 02:FE:32:C8 Time deta Time deta	EF:A8 (Non-Resolv	able), 66	<ul> <li>i:66 ▼ </li> <li>Detais</li> <li>➢ All field</li> <li>Name</li> <li>➢ Advertis</li> <li>➢ Advertis</li> <li>➢ Adv</li> <li>○ </li> <li>○</li></ul>	s Show in overview ement erisement ype contructy mng Overal Start Time Overal Start Time Overal Start Time Advertiser Advertiser Bash Beacon Ype Advertiser Advertis	<ul> <li>Analysis Ad</li> <li>Analysis Ad</li> <li>Display A</li> <li>Value</li> <li>Non-Connectable Under Start</li> <li>3.555 981 125 1.37 ms</li> <li>1.369 ms</li> <li>02:FE:32:C8:EF:A6 (No</li> <li>Mesh Beacon</li> <li>Reserved (Into2)</li> <li>C5 4C 60 FB 10 5F EE</li> </ul>	9     ×       9     ×       0     0       13*     1'3*       1'3*     1'3*       6     8       43     2       5     6
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Jools Help View Layout Search Record Jools Help View Layout Search Record - State State Search Search Jools Help View Entry VoerNew Message top Protocol: Single - All Layers + P = P = P = V = V = V = V = V = V = V =									
  | Itimue         Image         <   
   | Si Al El OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 9A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 9A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 9A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 9A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 9A EE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE 3A EE OS SE 9A AE OU FE 75 CE 48 SE AE O<br>SE AE OU SE SE AAE OU FE 75 CE 48 SE AE O<br>SE AE OU SE SE AAE OU FE 75 CE 48 SE AE O<br>SE AE OU SE SE AAE OU FE 75 CE 48 SE AE O<br>SE AE OU SE SE AAE OU FE 75 CE 48 SE   | ag: Only 02:FE:32:C8<br>Time deta<br>11.002 403 500<br>11.002 403
500<br>11.002 403 500<br>10.099 712 625<br>10.099 226 500<br>10.099 226 500<br>10.099 250 875<br>10.099 550 875<br>10.099 550 875<br>10.099 551 67 000<br>11.007 065 750<br>11.007 565 525   | EF:A8 (Non-Resolv<br>Q 2 - Search<br>13555901 125<br>24.554 693 750<br>13555901 125<br>24.554 693 750<br>46.556 314 000<br>7.556 086 625<br>68.550 159 500<br>101.548 348 250<br>101.548 348 250<br>101.548 348 250<br>112.555 496 623<br>134.5564 496 623<br>136.558 130 500   | able), 66  | is66 • Constant of the second seco   
  | S Show in overview ement erdsement yopt voral Start Time y Overal  | Analysis Ad<br>Display - Analysis Ad<br>Value<br>Non-Connectable Undre<br>Start<br>13.555 981 125<br>1.37 ms<br>1.369 ms<br>02:FE-32:C8:EF-A8 (No<br>Mesh Beacon<br>Reserved (0x02)<br>C5 4C 60 FB 10 5F EI   | 4<br>9 ×<br>0 = 2<br>0 = 2<br>0 = 2<br>134<br>136<br>136<br>136<br>136<br>136<br>136<br>136<br>136  
  |
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Resage too Protocols Single - Allayers + e* Song Restart Searce & Cort Fortocols Single - Allayers + e* Song View Resage too Protocols Single - Allayers + e* Song View Resage too Protocols Single - Allayers + e* Song View Resage too Protocols Single - Allayers + e* Song View Resage too Protocols Single - Allayers + e* Song View Resage too Protocols Single - Allayers + e* Song View Resage too Protocols Single - Allayers + e* Reservel (000) Song View Resage too Non-Connectable Underceted (02:FE:32:CB:FFA ( Reservel (000) Song View Connectable Underceted (02:FE:32:CB:FFA ( Reservel (000) Non-Connectable Underceted (02:FE:32:CB:FFA ( Reservel (000) <	Intimue         Image: Ima	Si Aler 05 56 9A AE 00 F6 75 C6 48 95 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 95 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 55 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 55 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 55 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 55 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 55 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 55 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 55 A6 15 53 A BF 05 56 9A AE 00 F6 75 C6 48 55 A6 15 53 AB F0 55 59 AE 00 F6 75 C6 48 55 A6 15 53 AB F0 55 59 AE 00 F6 75 C6 48 55 A6 15 53 AB F0 55 59 AAE 00 F6 75 C6 48 55 A6 15 55 59 AAE 00 F6 75 C6 48 55 A6 15 53 AB 50 50 50 50 50 50	g: Only 02:FE:32:C8 Time deta 1.002 493 500 1.00298 712 625 1.00298 725 625 1.0099 236 500 1.0399 208 520 875 1.0399 200 125 1.0399 120 625 1.0399 120 625 1.0399 120 625 1.0399 120 625 1.0399 120 625 1.0399 689 230 1.0394 625 1.0399 5167 000 1.0399 120 625 1.0399 689 255 1.0394 689 125 1.0394 689 125 1.03	EF:A8 (Non-Resolv Q 2 - Search Tme 2 2.553 487 625 13.555 901 125 24.554 633 750 35.557 077 500 35.557 077 500 35.557 077 500 35.557 077 500 35.557 077 500 35.557 077 500 35.557 078 500 101.548 348 250 112.550 469 255 112.555 446 325 112.555 446 325 113.4550 667 000 145.554 496 625 156.548 184 875 167.555 730 500	able), 666	<ul> <li>c66 • C</li> <li>Detais</li> <li>All field</li> <li>Name</li> <li>Advertis</li> <li>Nature</li> <li>Advertis</li> <li>Nature</li> <li>Nature</li></ul>	S Show in overview ement entent ype contrusty ming Overal Start Time Overal Start T	Analysis Ad Display - A Non-Connectable Under Start 13.555 981 125 1.37 ms 1.369 ms 02:FE:32:C8:EFJAB (No Mesh Beacon Reserved (no2) C5 4C 60 FB 10 5F EE	a × a × a × a × a × a × a × a ×
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help View Layout Search Record View Message Log Protocol Single View Layout Search Record View Record Vie	Itimue         Image         <	Comparing the second seco	g: Only 02:FE:32:C8 ✓ ✓ Time deta 4 10:099 1712 625 A 10:099 126 625 A 10:099 226 500 10:099 226 500 10:099 226 500 10:099 226 500 A 10:099 226 500 A 10:099 226 500 A 10:099 216 700 A 10:099 120 621 A 10:098 120 A 10:098 250 A 10:098 120 625 A 10:098	EF:A8 (Non-Resolv <b>Q ()</b> - Search Tme <b>1</b> 2.553 487 625 <b>13.555</b> 981 125 24.554 693 750 <b>35.557</b> 077 500 <b>46.556</b> 3184 000 79.548 359 625 90.553 181 250 112.550 409 625 112.550 409 625 123.557 667 000 145.554 184 875 165.548 184 165.548 184 165.548 184 165.548 184 165.548 185 165.548 185 165.558 185 165.558 185 165.558 185 165.558 185 165.558 185 165.558 185 165.558 1858 1858 1858 15	able), 666	<ul> <li>is66 • I Petals</li> <li>All field</li> <li>Name</li> <li>Advertis</li> <li>I Advertis</li> <l< td=""><td>S Show in overview ement ertisement ype contructy mng      Overal Duration     Duration</td><td>− □     □ Analysis Ad     □     □ Analysis Ad     □     □     □ Analysis Ad     □</td><td>4     9     ×       9     ×     2       Dee     0     13?       1'34     1'34     1'34       1'35     1'36     1'37       1'36     •     ×       7     •     •</td></l<></ul>	S Show in overview ement ertisement ype contructy mng      Overal Duration	− □     □ Analysis Ad     □     □ Analysis Ad     □     □     □ Analysis Ad     □	4     9     ×       9     ×     2       Dee     0     13?       1'34     1'34     1'34       1'35     1'36     1'37       1'36     •     ×       7     •     •
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help View Layout Search Record Tools Help View Layout Search Record Tools Help View Entry Vorthew Message too Protocol: Single      All Layers      All system      Von Connectable Undereded 02:FE32:CBEFA8 ( Reserved (0x0);     Non-Connectable Undereded 02:FE33:CBEFA8 ( Reserved (0x0);     Non-Connectable Undereded 02:FE33:C	ttinue	Si Ale 05 56 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 05 56 9A AE 00 F6 75 C6 48 56 A6 A 55 3A 8F 05 56 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 A6 C 55 3A 8F 06 55 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 95 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 9A AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 59 AA AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 59 AA AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 59 AA AE 00 F6 75 C6 48 56 AC C 55 3AF 00 56 59 AA AE 00 F6 75 C6 48 56 AC C 55 59 AE 00 56 59 AA AE 00 F6 75 C6 48 56 AC C 55 59 AE 00 F6 75 C6 48 56 AC C	ag: Only 02:FE32:C8	EF:A8 (Non-Resolv	able), 66 4 • • × • • • •	<ul> <li>c66 • •</li> <li>Detais</li> <li>× All field</li> <li>Name</li> <li>Advertis</li> <li>Nata</li> <li>• •</li> &lt;</ul>	s Show in overview ement ertisement ype overal Start Time Overal Start Time Overal Start Time Overal Duraton Overoids Advertiser Advertiser Advertiser Advertiser Start		Image: 2 minipage     Image: 2 minipage       Image: 2 minipage
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help Fortocols Single - All Nescoe Log Protocols Single - All Nescoe Log Protocols Single - All Nescoe Log Protocols Single - All Nescoe Log Resource Comparison of the Searce Log Protocols Single - All Nescoe Log Resource Comparison of the Searce Log Resource Log <p< td=""><td>ntinue</td><td>Si Ale 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 106 55 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 106 55 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 400 55 9A AE 00 F6 7</td><td>g: Only 02:FE:32:C8 Time deta 1.0029 437 500 1.0099 712 635 1.0099 712 635 1.0099 712 635 1.0099 236 500 1.0099 236 500 1.0099 236 500 1.0099 157 000 1.009 167 000 1.009 167 000 1.009 685 25 1.0099 689 125 1.009 489 350 1.009 489 350 1.009 489 350 1.009 489 350 1.009 489 350 1.000 489 350 1.000 489 350 1.000 204 450 1.000 204 500 1.000 204 500 1.00</td><td>EF:A8 (Non-Resolv <b>Q 2</b> - Search <b>Tme</b> <b>2</b>.553 487 625 <b>13555 941 125</b> <b>24.554 633 750</b> <b>35.557 077 500</b> <b>35.557 077 500</b> <b>35.557 077 500</b> <b>35.557 077 500</b> <b>35.557 078 000</b> <b>57.556 668 625</b> <b>68.550 159 500</b> <b>79.548 348 250</b> <b>101.548 348 250</b> <b>101.548 348 250</b> <b>101.548 348 260</b> <b>101.548 348 260</b> <b>101.548 348 260</b> <b>101.555 7496 625</b> <b>155.558 166 7000</b> <b>178.550 539 625</b> <b>138.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>131.554 873 0000</b> <b>131.554 873 0000 <b>131.554 873 0000</b> <b>131.554 873 0000 <b>131.554 873 0000</b> <b>131.554 873 0000</b> <b>131.554 873 0000</b> <b>131.554 873 00000 <b>131.554 873 00000 <b></b></b></b></b></b></td><td>able), 66</td><td><ul> <li>c66 ▼ </li> <li>Detais</li> <li>X All field</li> <li>Name</li> <li>Advertie</li> <li>3 Advertie</li> <li>4 Advertie</li> <li>4 Advertie</li> <li>3 Advertie</li> <li>4 Advertie</li> <li>3 Advertie</li> <li>4 Ad</li></ul></td><td>S Show in overview ement erisement Vpe output Outp</td><td>Analysis Add     Analysis Add     Display      Analysis Add     Display      Au     Analysis Add     Display      Analysis     Ana</td><td>9     ×       9     ×       0     2       13*     1'3*       1'3*     1'3*       1'3*     1'3*       1'3*     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*</td></p<>	ntinue	Si Ale 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 105 56 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 106 55 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 106 55 9A AE 00 F6 75 C6 48 56 AE 0 59 AB 400 55 9A AE 00 F6 7	g: Only 02:FE:32:C8 Time deta 1.0029 437 500 1.0099 712 635 1.0099 712 635 1.0099 712 635 1.0099 236 500 1.0099 236 500 1.0099 236 500 1.0099 157 000 1.009 167 000 1.009 167 000 1.009 685 25 1.0099 689 125 1.009 489 350 1.009 489 350 1.009 489 350 1.009 489 350 1.009 489 350 1.000 489 350 1.000 489 350 1.000 204 450 1.000 204 500 1.000 204 500 1.00	EF:A8 (Non-Resolv <b>Q 2</b> - Search <b>Tme</b> <b>2</b> .553 487 625 <b>13555 941 125</b> <b>24.554 633 750</b> <b>35.557 077 500</b> <b>35.557 077 500</b> <b>35.557 077 500</b> <b>35.557 077 500</b> <b>35.557 078 000</b> <b>57.556 668 625</b> <b>68.550 159 500</b> <b>79.548 348 250</b> <b>101.548 348 250</b> <b>101.548 348 250</b> <b>101.548 348 260</b> <b>101.548 348 260</b> <b>101.548 348 260</b> <b>101.555 7496 625</b> <b>155.558 166 7000</b> <b>178.550 539 625</b> <b>138.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>139.551 389 125</b> <b>131.554 873 0000</b> <b>131.554 873 0000 <b>131.554 873 0000</b> <b>131.554 873 0000 <b>131.554 873 0000</b> <b>131.554 873 0000</b> <b>131.554 873 0000</b> <b>131.554 873 00000 <b>131.554 873 00000 <b></b></b></b></b></b>	able), 66	<ul> <li>c66 ▼ </li> <li>Detais</li> <li>X All field</li> <li>Name</li> <li>Advertie</li> <li>3 Advertie</li> <li>4 Advertie</li> <li>4 Advertie</li> <li>3 Advertie</li> <li>4 Advertie</li> <li>3 Advertie</li> <li>4 Ad</li></ul>	S Show in overview ement erisement Vpe output Outp	Analysis Add     Analysis Add     Display      Analysis Add     Display      Au     Analysis Add     Display      Analysis     Ana	9     ×       9     ×       0     2       13*     1'3*       1'3*     1'3*       1'3*     1'3*       1'3*     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Jools Help File View Layout Search Record Jools Help File View Layout Search Record Jools Help File View Layout Resource Search Record View Resource Comparison of the Save & Comp	Itimue         Image         Image <t< td=""><td>■ ■ ■ Markers ■ ■ ■ ■ ■ Filteria S5 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 5</td><td>g: Only 02:FE32:C8  Time deta  Ti</td><td>EF:A8 (Non-Resolv Q</td><td>able), 66</td><td>c66</td><td>E 122     S Show in overview ement ertisement ype Overal Duration     Overal     Overal     Overal Duration     Overal Duration     Overal D</td><td>- □     - □    □    </td><td>0         ×           0         x           0         x           0         x           2         x           134         134           135         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x</td></t<>	■ ■ ■ Markers ■ ■ ■ ■ ■ Filteria S5 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0 S6 3A BF 05 5	g: Only 02:FE32:C8  Time deta  Ti	EF:A8 (Non-Resolv Q	able), 66	c66	E 122     S Show in overview ement ertisement ype Overal Duration     Overal     Overal     Overal Duration     Overal Duration     Overal D	- □     - □    □	0         ×           0         x           0         x           0         x           2         x           134         134           135         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x           0         x
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help View Commetable Undereded (02:FE3:20:EF:A8 ( Reserved (0x0) View Commetable Undereded (02:FE3:20:EF:A8 ( Reserve	Number         No         No <th< td=""><td>Markers • P P P P P P P P P P P P P P P P P P</td><td>ag: Only 02:FE32:C8</td><td>EF:A8 (Non-Resolv</td><td>able), 666</td><td>C66      Catala     Catalaa     Catalaaa     Catalaaa     Catalaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</td><td>s Show in overview ement ertisement ype contrauty mmg Overal Start Time Overal Sta</td><td>- □     - □   □   </td><td>a × d a × be cted 9 0 13° 1°4° 1°4°</td></th<>	Markers • P P P P P P P P P P P P P P P P P P	ag: Only 02:FE32:C8	EF:A8 (Non-Resolv	able), 666	C66      Catala     Catalaa     Catalaaa     Catalaaa     Catalaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	s Show in overview ement ertisement ype contrauty mmg Overal Start Time Overal Sta	- □     - □   □   □	a × d a × be cted 9 0 13° 1°4° 1°4°
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Resould Search Record Tools Help Fortocols Single - Allayers + et al. State & Control Cools Help Fortocols Single - Allayers + et al. State & Control Cools Help Fortocols Single - Allayers + et al. State & Control Cools Help Show: Tem = "Non-Connectable Undereded (02:FE332C8:EFA8 ( Reserved (0x0) For Non-Connectable Undereded (02:FE332C8:EFA8 ( Reserved (0x0)	Intimue         Image: Ima	Comparison of the second se	g: Only 02:FE:32:C8 Time deta 1.0029 472 500 1.0099 712 625 1.0099 712 625 1.0099 712 625 1.0099 226 550 1.0099 226 550 1.0099 226 550 1.0099 206 125 1.0099 150 675 1.0099 167 000 1.000 645 625 1.0094 659 125 1.0094 649 375 1.0094 649 375 1.0094 642 30 1.009 777 625 1.000 777 625	EF:A8 (Non-Resolv <b>Q 2</b> - Search <b>Tme</b> <b>2.553 487 625</b> <b>3.555 017 500</b> <b>46 556 314 000</b> <b>57 550 668 625</b> <b>68 550 159 500</b> <b>79.548 348 250</b> <b>01.548 348 250</b> <b>01.548 348 250</b> <b>01.548 348 250</b> <b>134 555 667 000</b> <b>134 555 496 625</b> <b>155 559 625</b> <b>155 559 625</b> <b>134 555 539 625</b> <b>156 555 73 625</b> <b>158 551 539 625</b> <b>189 551 389 125</b> <b>189 551 389 125</b> <b>189 551 389 125</b> <b>135 548 73 000</b> <b>131 548 73 000</b> <b>131 544 730 667</b> <b>131 547 67</b> <b>135 67</b> <b>135 551 59 625</b> <b>133 551 59 625</b> <b>133 551 59 625</b> <b>133 551 59 625</b> <b>133 551 59 625</b> <b>134 555 539 625</b> <b>138 551 389 125</b> <b>135 551 887 3000</b> <b>131 548 673 000</b> <b>131 548 673 000 <b>131 548 673 000</b> <b>131 548 673 000</b> <b>131 548 673 000 <b>131 559 568 675</b> <b>131 559 568 6</b></b></b></b></b></b>	able), 66 4 b x • 1 2 2 1 2 0 1 2	Constant of the second s	S Show in overview ement erisement Vpe oursel Duraton Oursel Start Time Overal Star	Analysis ↑ Ad     Analysis ↑ Ad     Display ◆      Display ◆	9     ×       9     ×       0     2       13*     1'3*       1'3*     1'3*       1'3*     1'3*       1'3*     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*       1'3     1'3*
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help View Rem # "Non-Connectable Undereded 02:FE3:20:EF748 ( Reserved (0x0) View Connectable Undereded 02:FE3:20:EF748 ( Reserved (0x0) View Conne	Itimue         Image: Ima	S3 A BF 05 56 9A AE 00 F6 75 C6 48 56 AC 0           S5 3A BF 05 56 9A AE 00	g: Only 02:FE:32:C8  Time deta  T	EF:A8 (Non-Resolv Q	able), 666 ↓ × × •   • 2 ↓ × × × × • ×	<ul> <li>c66 ▼ 0</li> <li>Detais</li> <li>¥ All field</li> <li>Name</li> <li>Advertis</li> <li>Name</li> <li>Nam</li> <li>Nam</li> <li>Nam</li></ul>	S Show in overview ement ertisement ype contructy mmg      Overal Duration     Duraton     Duraton     Outar Type     Advertser     Advertser     Mesh Beacon     Duraton     Durator     Durato	- □     - □    -	0         ×           0         ×           0         ×           0         ×           0         ×           0         ×           10         ×           13%         13%           13%         ×           43         ×           2         ×           43         ×           7         ×           2         ×
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Jools Help View Layout Search Record Jools Help View Layout Search Record - Stop Restart & Save & Cord Low Energy Workiew Message top Protocol: Single < All layers + et al. (Stop Restart & Save & Cord Protocol: Single < All layers + et al. (Stop Restart & Save & Cord Protocol: Single < All layers + et al. (Stop Restart & Save & Cord Protocol: Single < All layers + et al. (Stop Restart & Save & Cord Protocol: Single < All layers + et al. (Stop Restart & Save & Cord Restruction Restart & Restruct (Stop Restart & Save & Cord Protocol: Single < All layers + et al. (Stop Restart & Save & Cord Restruction Restart & Restruct (Stop Restart & Save & Cord Restruction Restart & Restruct (Stop Restart & Save & Cord Restruction Restart & Restruct (Stop Restart & Save & Cord Restruction Restart & Restruct (Stop Restart & Save & Restruct (Stop Restart & Save & Cord Restruction Restart & Restart (Stop Restart & Save & Restruct (Stop Restart & Save & Restruct (Stop Restart & Restruct (Stop Restar	Navigate           28 items kept, 7 filtered           28 items kept, 7 filtered           28 items kept, 7 filtered           29 C3 4C 60 F8 10 5F EE 80 10 03 A           20 C3 4C 60 F8 10	So         Markers         So         Filteria           So         Markers <t< td=""><td>g: Only 02:FE32:C8</td><td>EF:A8 (Non-Resolv</td><td>able), 66</td><td>Content of the second s</td><td>s Show in overview ement ertsement ype contrauty mmg Overal Start Time Overal Star</td><td>- □      - □   </td><td>J × d / 4 x × 5 c 6 c 137 138 139 139 139 139 139 139 139 139</td></t<>	g: Only 02:FE32:C8	EF:A8 (Non-Resolv	able), 66	Content of the second s	s Show in overview ement ertsement ype contrauty mmg Overal Start Time Overal Star	- □      - □	J × d / 4 x × 5 c 6 c 137 138 139 139 139 139 139 139 139 139
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Search Record Tools Help Fortocols Single - Allayers + et al. State & Control Cools Help Fortocols Single - Allayers + et al. State & Control Cools Help Fortocols Single - Allayers + et al. State & Control Cools Help Fortocols Single - Allayers + et al. State & Control Cools Help Show: Tem = "Non-Connectable Understell (02:FE332CBEFA8 ( Reserved (000) State Connectable Understell (02:FE332CBEFA8 ( Reserved (000) Non-Connectable Understell (02:FE332CBEFA8 ( Reserved (000) </td <td>Huture       Image: Control of the second seco</td> <td>So         Markers         Image: So         Image:</td> <td>ag: Only 02:FE:32:C8  Time deta  I 1.002:403 500  1 1.099 712 625  1 1.099 236 500  1 0.999 236 500  1 0.999 236 500  1 0.999 236 500  1 0.999 236 500  1 0.999 250 675  1 0.999 250 625  1 0.999 5167 000  1 1.004 625  1 0.999 689 250  1 1.007 665 750  1 0.099 689 250  1 0.099 689 250  1 0.099 689 250  1 0.099 689 250  1 0.009 489 250  1 1.001 245 625  1 0.099 689 250  1 1.001 245 625  1 0.099 489 250  1 1.001 245 625  1 0.000 24</td> <td>EF:A8 (Non-Resolv</td> <td>able), 666</td> <td>c66     ▼     0       Detais     &gt;     All field       Name     Advertie     3     Advertie       □     3     Advertie     1       □     3     Advertie     1       □     3     1       <td< td=""><td>s Show in overview ement erisement ype oursel Duraton Oursel Duraton Oursel Start Time Overal Start Time Overal Duraton Works Advertiser Advertiser Advertiser Advertiser Beson Type Advertiser C 2 B 4 2 5 4 6 5 55 F E 8 0 1D 03 35 00 55 F E 8 0 1D 03 35 00 55 F E 8 0 1D 03 35 00 55 6 2 6 3 6 5 6 8 6 5 8</td><td>Analysis Add     Analysis Add     Display ●      Analysis Add     Display ●      Analysis     Analysis</td><td>3 × 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 ×</td></td<></td>	Huture       Image: Control of the second seco	So         Markers         Image: So         Image:	ag: Only 02:FE:32:C8  Time deta  I 1.002:403 500  1 1.099 712 625  1 1.099 236 500  1 0.999 236 500  1 0.999 236 500  1 0.999 236 500  1 0.999 236 500  1 0.999 250 675  1 0.999 250 625  1 0.999 5167 000  1 1.004 625  1 0.999 689 250  1 1.007 665 750  1 0.099 689 250  1 0.099 689 250  1 0.099 689 250  1 0.099 689 250  1 0.009 489 250  1 1.001 245 625  1 0.099 689 250  1 1.001 245 625  1 0.099 489 250  1 1.001 245 625  1 0.000 24	EF:A8 (Non-Resolv	able), 666	c66     ▼     0       Detais     >     All field       Name     Advertie     3     Advertie       □     3     Advertie     1       □     3     Advertie     1       □     3     1 <td< td=""><td>s Show in overview ement erisement ype oursel Duraton Oursel Duraton Oursel Start Time Overal Start Time Overal Duraton Works Advertiser Advertiser Advertiser Advertiser Beson Type Advertiser C 2 B 4 2 5 4 6 5 55 F E 8 0 1D 03 35 00 55 F E 8 0 1D 03 35 00 55 F E 8 0 1D 03 35 00 55 6 2 6 3 6 5 6 8 6 5 8</td><td>Analysis Add     Analysis Add     Display ●      Analysis Add     Display ●      Analysis     Analysis</td><td>3 × 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 ×</td></td<>	s Show in overview ement erisement ype oursel Duraton Oursel Duraton Oursel Start Time Overal Start Time Overal Duraton Works Advertiser Advertiser Advertiser Advertiser Beson Type Advertiser C 2 B 4 2 5 4 6 5 55 F E 8 0 1D 03 35 00 55 F E 8 0 1D 03 35 00 55 F E 8 0 1D 03 35 00 55 6 2 6 3 6 5 6 8 6 5 8	Analysis Add     Analysis Add     Display ●      Analysis Add     Display ●      Analysis     Analysis	3 × 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 × 2 ×
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Resource Search Record Tools Help Fortocols Single - Allangers + et Searce Search Record View Resource Colling File Col	Hiture       Image: Control of the second seco	So So So So A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 6F 05 56 9A 42 00 F6 75 C6 48 55 46 C           So A 48 00 F6 75 C6 48 55 40 A2 00 F6 75 C6 48 55 46 C           So A 48 00 F6 75 C6 48 55 40 A2 00 F6 75 C6 48 55 40 C           So A 48 00 F6 75 C6 48 55 40 A2 00 F6 75 C6 48 55 40 C           So A 48 00 F6 75 C6 48 55 40 A2 00 F6 75 C6 48 55 40 C           So A 48 00 F6 75 C6 48 55 40 A2 00 F6 75 C6 48 55 40 C           So A 48 00 F6 75 C6 48 55 40 A2 00 F6 75 C6 48 55 40 C           So A 48 00 F6 75 C6 48 55 40 A2 00 F6 75 C6 48 55 40 C           So A 48 00 F6 75 C6 48 55 40 A2 00 F6 75 C6 48 55 40 AC           So A 48 00 F6 75 C6 48 55 40 AC           So A 48 00 F6 75 C6 48 55 40 AC           So A 48 00 F6 75 C6 48 55 40 AC           So A 48 00 F6 75 C6 48 55 40 AC           So A 48 00 F6 75 C6 48 55 40 AC           So A 48 00 F6 75 C6 48 55 40 AC           So A 48 00 F6 75 C6	g: Only 02:FE:32:C8 Time deta 4 10:099 712 625 11:002 493 500 10:099 216 625 11:002 493 500 10:099 236 500 10:099 236 500 10:099 236 500 10:099 236 200 10:099 550 875 10:007 086 750 10:007 086 750 10:007 086 750 10:007 086 750 10:007 685 625 10:007 68	EF:A8 (Non-Resolv Q	able), 66 4 b x •   6 2 4 b x •	<ul> <li>c66 • • • • • • • • • • • • • • • • •</li></ul>	S Show in overview ement enternet Ype Contrusty Iming Overal Start Time Overal Duration Duration Werkes Advertiser Adver	Analysis ↑ Ad     Analysis ↑ Ad     Display ← 🕞     Value     Non-Connectable Under     Start     13.555 981 125     1.369 ms     02:FE:32:C8:EFA8 (No     Mesh Beacon     Reserved (No2)     C5 4C 60 FB 10 5F EF     Search	a × d 9 × 2 × 0 0 13°: 1'3¢ 1'4¢ 1'4
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Message too Protocol: Single • All layers + et al. Stop III Restart III Save & Control View Message too Protocol: Single • All layers + et al. In the Searce View Message too Protocol: Single • All layers + et al. In the Searce View Message too Protocol: Single • All layers + et al. In the Searce View Message too Protocol: Single • All layers + et al. In the Searce View Message too Protocol: Single • All layers + et al. In the Searce View Message too Non-Connectable Understed (02:FE:32:CB:FAA ( Reserved (00:0) Non-Connectable U	Hitnue       Image: Image	• • • • • • • • • • • • • • • • • • •	g: Only 02:FE32:C8	EF:A8 (Non-Resolv <b>Q</b> • Search Tme 2 2,553 487 6493 750 2,255 487 6493 750 13,555 001 125 24,554 6493 750 145,554 693 750 10,548 348 250 112,550 495 6425 90,553 181 250 10,548 348 250 112,550 495 6425 123,557 495 6750 134,550 657 530 500 145,554 81 64 275 156,548 148 4255 156,548 148 4255 121,554 873 0000 222,549 019 250 231,549 556 875 <b>Call</b>	able), 666	cc6 • • • • • • • • • • • • • • • • • •	S Show in overview ement ertsement ye outeruby mng     Overal Start Time     Overal Start Time     Overal Duraton     Du	- □     - □    -	a → a →
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Reside Log Status View Resarct View	Hitmue       No       Image: A mark       Image: A mark         28 items kept, 7 filtered       Image: A mark       Image: A mark         20 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         21 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         22 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         23 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         24 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 D 03 A       Image: A mark       Image: A mark         25 G + C GO FB 10 SF EE 80 10 D 03 A       Image: A mark       Image: A mark         <	• • • • • • • • • • • • • • • • • • •	ag: Only 02:FE:32:C8	EF:A8 (Non-Resolv 2	able), 66 4 <b>b x</b> • <b>e ±</b> 1 <b>v ^</b> <b>q x</b> <b>q x</b> <b>q x</b> <b>a</b>	<ul> <li>c66 ◆ <ul> <li>Petais</li> <li>All field</li> <li>All field</li> <li>Adverti</li> <li>Adverti<!--</td--><td>s Show in overview ement ertisement yope Overal Start Time &gt; Overal Start Time &gt; Overa</td><td>Analysis Add     Analysis     Analysis</td><td>a → a →</td></li></ul></li></ul>	s Show in overview ement ertisement yope Overal Start Time > Overal Start Time > Overa	Analysis Add     Analysis     Analysis	a → a →
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Search Record Tools Help File View Layout Resage too Protocols Single - Allayers + et al. Stop The Resard I to Save & Cort Low Energy Overview Message too Protocols Single - Millayers + et al. Stop The Reserved (000) Show: Rem × "Non-Connectable Underected (02:FE332C8:FFA6 ( Reserved (000) Store The Connectable Underected (02:FE332C8:FFA6 ( Reserved (000) Store Connectable Underected (02:FE332C8:FFA6 ( Reserved (000) Store Connectable Underected (02:FE332C8:FFA6 ( Reserved (000) Non-Connectable Underected (02:FE33	Hiture         Image: Imag	• • • • • • • • • • • • • • • • • • •	g: Only 02:FE:32:C8  Time deta  I 10:098 712 625  I 10:098 712 625 I 10:099 726 75 I 10:999 206 500 I 10:999 206 500 I 10:999 206 500 I 10:999 206 500 I 10:999 507 102 I 10:999 507 102 I 10:999 507 102 I 10:09 167 000 I 10:00 164 500 I 10:00 164 520 I 10	EF:A8 (Non-Resolv	able), 66 • • • • • • • • • • • • • • • • • • •	c66 ▼  Detais All field Name Advertis I Ad	S Show in overview ement erisement Vpe oursel Duraton Overal Start Time Overal Star	Analysis ↑ Ad     Analysis ↑ Ad     Display ◆      Dis	a → a →
Recording from BTR1-23087 - Ellisys Bluetooth Analyzer File View Layout Search Record Tools Help View Layout Search Record Tools Help View Layout Search Record Tools Help View Entry Overview Message top Protocol Single - All layers + P and P a									
  | Name       Name       Name         28 items kept, 7 filtered         28 items kept, 7 filtered         9 G4 660 F8 10 SF EE 80 10 03 3A         9 G4 660 F8 10 SF EE 80 10 03 3A         0 G4 66  
   | • • • • • • • • • • • • • • • • • • •   | g: Only 02:FE32:C8  
  | EF:A8 (Non-Resolv<br>C 2 - Search<br>Tme 2<br>2553 487 c2<br>2553 487 c3<br>2553 487 c3<br>2553 487 c3<br>2553 487 c3<br>2555 493 750<br>46.555 501 125<br>24.554 693 750<br>46.555 501 125<br>90.553 181 259<br>101.548 348 250<br>101.548 348 250<br>101.548 348 255<br>101.548 348 255<br>101.548 348 255<br>101.555 481 84 255<br>167.555 749 500<br>173.550 539 625<br>189.551 389 125<br>200.552 838 500<br>211.554 873 000<br>222.549 019 250<br>331.549 596 825<br>200.552 838 500<br>222.549 019 250<br>331.549 596 825<br>200.551 200<br>200.510 200<br>200 200   | able), 66<br>• • • • • • • • • • • • • • • • • • •   | c66 • • • • • • • • • • • • • • • • •   
  |  | - □    □    -   |
a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a<br>a   |

# ellisus beacon & private beacon package

Figure 33.56: Beacon packets received by ellisys after opening beacon + private beacon

# 33.6 Setting Interface

The Setting interface allows for Manage Network, OOB Database, Root Cert, settings (Enable Log, Enable Private Mode, Provision Mode, Enable Subscription Level Service, Enable Log, Enable Private Mode, Provision Mode, Enable Subscription Level Service, Extend Bearer Mode, Use No-OOB Automatically, Share Import Complete Action, Online Status, Reset Settings), How To Import Bin File, Get More Telink Apps.

	Settin	g	
E Manage	Network ate/Delete, Imp	ort/Export	>
00B Da	tabase		
used in sta	tic-oob provisio	oning	>
Root Ce used in cer	r <b>t</b> tificate-based µ	provisioninę	g >
Settings	as.		>
Q	ē	윦	Setting

Figure 33.57: Android & iOS setting interface

# 33.6.1 Manage Network

**Difference between Manage Network and the Network interface at the bottom of the app:** in Manage Network we can view, manage, apply and share different networks, while the Network at the bottom of the app is for the current network to perform Mesh OTA, Scene and other operations. The network selected in the blue box is the current network, click the + sign in the upper right corner to create a new network.



Figure 33.58: Android & iOS manage network interface

#### 33.6.1.1 Show Detail

Telink

т

Click on the Network you want to view and select Show detail to view the details of the Network (see section 4.1 for the Mesh Info interface).



Figure 33.59: Show detail interface

# 33.6.1.2 Share Export

#### (1) Export by File

At Network List interface, click on a shared Network and select Share Export, select the Net Key you need to share, select Json File for the sharing method and click on the export button to export the json file, the Android version of the exported json file is saved in storage/emulated/0/TelinkBleMesh directory, and the iOS version of the exported json file is shared through the iTunes file inside.



Android	export json file	iOS exp	oort json file
< Network List +	K Share Export	< Network List +	K Share Export network-2
Name           Default Mesh           UUID           15845521-7e78-558a-02c9-b8bf1e70fcd0           Timestamp           2023-11-13T10:07:49+08:00              Name           network-1           UDD           20394e4-e15a-7850-a483-13a3c3bbffc1           Timestamp           2023-11-15T14:14:23+08:00	Export mesh storage to json 1.5dect at least one network key 2.5dect destination 3.Click EXPORT Select Net Keys 10 0:00 Fill Science 2007 Select Net Key 1 10 0:01 Sol Net Key 2 Sub Net Key 2	None           Default Mesh           UND           115:2796-166A-315F-A6EA-837AD3371007           Tommer           2023-11-13706:53:16Z           None           network-1           USD3           13608FFF-D283-CDE6-3357-E43DF493289D           Unrestance           2023-11-11706:50:24Z	Export mesh storage to joon 3.54ect destination(SKN He of ORCodd); 3.24ect destination(SKN He of ORCodd); 3.24ect destination(SKN He of ORCodd); 4.24ect destination(SKN He o
Name network-2 UUD0 9df75bff-dc8d-3f20-3b91-b7a6b0021205 select action for : network-2 ×	10 0.d2 ♥ 617AC325ED9F4F48FD6375A169DF9FA8 Select destination ○ QRCode ① ● JSON File ①	etwork-2 UUD 80240443-430C-6098-87DE-0D17CEEAA8D5 Timestamo 2023-11-15T06:20:37Z	
문 Show Detail >		select action for: network-2	Select destination
Share Export > ↔ Switch To This Network >		Show Detail >	QRCode + Cloud Transfer JSON () JSON File ()
DELETE	EXPORT	DELETE	EXPORT

Figure 33.60: Android & iOS export json file

#### (2) Export by QR Code

Telink

T

At Network List interface, click on a shared Network and select Share Export, select the Net Key to be shared, select QRCode for the sharing method and click on the export button to display the json information in the form of QR code (the QR code has a time limit and will expire after 300 seconds).

<ul> <li>Network I</li> <li>Name</li> <li>Default Mesh</li> <li>UUD</li> <li>f584552f-7e78-558a-02c9</li> <li>Timestamp</li> <li>2023-11-13T10:07:49+08:</li> </ul> Name <ul> <li>network-1</li> <li>UUD</li> <li>2a7a94e4-e15a-7850-a48:</li> <li>Timestamp</li> <li>2023-11-15T14:14:23+08:</li> </ul>	_ist + P-b8bf1e70fcd0 S 00 (		<	Share-QRCode
Name Default Mesh UUD f584552f-7e78-558a-02c9 Timestamp 2023-11-13T10:07:49+08: Name network-1 UUD 2a7a94e4-e15a-7850-a48: Timestamp 2023-11-15T14:14:23+08:	-b8bf1e70fcd0 S	Export mesh storage to json         1.Select at least one network key         2.Select destination         3.Click EXPORT         Select Attentation         Select Net Keys         Default Net Key         10 0x00         * 81AC208C3C701FD7C9CE793F8BE7B203         Sub Net Key 1		
Name           network-1           UUID           2a7a94e4-e15a-7850-a483           Timestamp           2023-11-13T10:07:49+085	2-b8bf1e70fcd0 S	Select Net Keys           Default Net Key           10 0x00           \$1AC208C3C701FD7C9CE793F8BE7B203           3 Sub Net Key 1		
Timestamp 2023-11-13T10:07:49+08: Name network-1 UUID 2a7a94e4-e15a-7850-a48: Timestamp 2023-11-15T14:14:23+08:	00	Default Net Key 0x00 National States		
2023-11-13T10:07:49+08: Name network-1 UUID 2a7a94e4-e15a-7850-a48: Timestamp 2023-11-15T14:14:23+08:		0 0x00 ✿ 81AC208C3C701FD7C9CE793F8BE7B203 Î Sub Net Key 1		
Name network-1 UUD 2a7a94e4-e15a-7850-a48: Timestamp 2023-11-15T14:14:23+08:		<ul> <li>81AC208C3C701FD7C9CE793F8BE7B203</li> <li>Sub Net Key 1</li> </ul>		
Name network-1 UUID 2a7a94e4-e15a-7850-a48: Timestamp 2023-11-15T14:14:23+08:		Sub Net Key 1		
network-1 UUID 2a7a94e4-e15a-7850-a48 Timestamp 2023-11-15T14:14:23+08:				
UUID 2a7a94e4-e15a-7850-a48 Timestamp 2023-11-15T14:14:23+08		ID 0x01		▋▌▝▛▝▎`▅┓▔▌▋▇▌
Timestamp 2023-11-15T14:14:23+08:	2-13-3-3-3bbffc1			
2023-11-15T14:14:23+08:		Sub Net Key 2		
	00			
		617AC325ED9F4F48FD6375A169DF9FA8		
	, and the second se	elect destination		
network-2				
9df75bff-dc8d-3f20-3b91	-b7a6b0021205			
elect action for : netwo	ork-2 ×	JSON File (!)		a ha fhair a bhfair a
Show Detail	>			
Share Export	>			2R-Code available in 500 seconds
Switch To This Network	« <b>&gt;</b>			
DELETE				





# 33.6.1.3 Switch To This Network

Telink

т

Click the specified Network to select Switch to this network to switch to the selected Network (the blue box synchronizes the switch to the specified Network).



Figure 33.63: Switch to this network

# 33.6.1.4 Import mesh

Click the Import button at the bottom right of the Network List interface to import a network.

<	Network List	+
Name		
Default	Mesh	
UUID f58/55	2f-7o78-5582-02c9-b8bf1o70fc	40
Timestan		uu
2023-1	1-13T10:07:49+08:00	•••
Name		
networ	k-1	
UUID	a6 a15a 7850 a682 12a2a2bbf	fa1
Za/a74 Timestan	np	
2023-1	1-15T14:14:23+08:00	•••
networ UUID 9df75b Timestan 2023-1	k-2 ff-dc8d-3f20-3b91-b7a6b00212 <sup>np</sup> 1-15T14:14:36+08:00	205
	REMOVE ALL	
	· V /	

Figure 33.64: Network list interface import button

#### (1) Import by Json File

#### Android APP:

At Network List interface, click the Import button in the lower right corner, Select source to choose Json File, Select File to choose the json file to be imported, select the json file, Preview button to preview the information of the json file, Import button to import Network, after successful import, return to Network List interface by default pop-up asking whether to switch to the just imported Network, according to the need not to switch or switch (the previous Network will be retained), you can also set the import automatically switches to the imported Network, the setting path: APP Home – Setting in the right bottom corner – Settings – Share Import Complete Action in the lower right corner, select Auto Switch.

# ד Telink

#### Telink SIG Mesh SDK Developer Handbook



Figure 33.65: Android json file import

#### iOS APP:

Put the json file into TelinkSigMesh APP through iTunes, click import button in the lower right corner of Network List interface, select the json file inside the select sorce, click Import button to select the json file to be imported, click import, a prompt box will pop up to indicate whether to import the json data or not, clicking cancel not to import network, clicking confirm to import network. After importing network, the default popup will ask whether to switch to the just-imported network, you can not switch or switch according to your needs (the previous network will be retained), you can also set the network to automatically switch to the imported network after importing, the setting path: APP home page – Setting in the right bottom corner – Settings – Share Import Complete Action and select Auto Switch.



Figure 33.66: iOS json file import

#### (2) Import by QRCode

#### Android APP:

At Network List interface, click on the lower right corner of the import button, Select source choose QRCode, click Import button, scan the QR code of the other party's shared Network, it will pop up whether to import Network, after the import is successful, the default automatic pop-up whether to switch to the just imported Network, not switch or switch according to the needs (the previous Network will be retained), you can also set the import automatically switch to the imported Network, set the path: APP home page – Settings in the right bottom corner – Setting – Share Import Complete Action, choose Auto Switch.



Figure 33.67: Android QR code import network interface



Figure 33.68: iOS QR code import network interface
#### 33.6.1.5 Delete Network

At Network List interface, click to select the network to be deleted, select DELETE, a prompt box will pop up whether to delete the network, you can delete the network according to the need, need to pay attention to is the current network can not be deleted, you can switch to other network and then delete it.



Figure 33.69: Delete the network interface

#### 33.6.1.6 Clear All Network

At Network List, click Remove All at the bottom to clear all Networks.



Figure 33.70: Clear all network interface

#### 33.6.2 OOB Database

The OOB Database is used for the App to find the corresponding Auth Value of the device when the device supports static-oob method for provisioning.

When the App is looking for Auth Value, it will use deviceUUID as the key to look up the table from database. If writes OOB data to the the device, you need to enter the corresponding UUID and OOB data in APP in order to normalize networking, and vice versa, the networking will fail.

Currently it supports 16-bit and 32-bit OOB data, which can be written at flash location 77800 as needed.

#### 33.6.2.1 Add an OOB Database Manually

At OOB List interface, click the + sign in the upper right corner and select Manual input to enter UUID and OOB data.

The UUID and OOB queries are as follows:

OOB: Burn 8258 mesh, 8269 mesh, 8278 mesh and other projects, write 16 or 32 bit OOB data at location 77800.

UUID: The device to be networked state with the universal light blue APP to view the device's UUID, specific operation: APP scans the device that needs to obtain the UUID (shown as below) – view the Complete list of 16-bit Service UUID, the yellow area in the figure is UUID.





#### 33.6.2.2 Import OOB Database via Txt File

Create a new txt document – enter 16 bytes UUID, an empty space to enter 16 or 32 bytes of OOB data and save – at OOB List interface click on the upper-right corner of the + sign to select import from file – select the txt file just saved.

# 

#### Figure 33.72: OOB data in TXT format

#### 33.6.2.3 Delete OOB Database

Long press one of the OOB data to delete the OOB data individually, and click the trash can button in the upper right corner to empty all OOB data.





#### 33.6.2.4 Use No-OOB Automatically

Use No-OOB Automatically can be added to devices that have OOB data written at the 77800 location but not recorded in the APP (provided that the ENABLE\_NO\_OOB\_IN\_STATIC\_OOB macro is turned on, and the Use No-OOB Automatically switch is turned on in the APP home page – setting – settings).



#### 33.6.3 Root Cert

When networking, the certificate is verified to determine whether the device is allowed to join the mesh network. When the certificate passes, the network will be successful, but if it does not pass, the network will fail and prompt certificate recordcheck error.

#### 33.6.3.1 Networking by Default Certificate

The V4.1.0.0 version APP comes with a default certificate, when the device uses the default certificate to group the network, it will use the current effective default certificate for verification, you can check it in APP home page – Setting – Root Cert.

<	Cert List 🔋 🗎	+	Root Certificate
			Last select name:Default Root Certificate
e <b>y</b>	pubKey: EC Public Key [80:8e:e8:9f:9c:7b:94:2f:e7:9d:2a:68:3c:fb :b3:4e:47:76:56:a3] X: 41dbf54a701efafc88d34a7233c383a640 6f815ae7ba9d52bb4625a123036995 Y: 95a3b139d35eedc1e65e33c128a69ab0d 037bfff985aa23a62b3218baeb17e3f Time-NotBefore:Wed Oct 11 14:00:48 GMT+08:00 2023 Time-NotAfter:Sat Oct 08 14:00:48 GMT+08:00 2033 Alg:SHA256WITHECDSA (Default)	>	root.der

Preset conditions:

- (1) In the mesh\_config.h file turn on the CERTIFY\_BASE\_ENABLE macro;
- (2) The CERT\_TYPE in the certify\_base\_crypto.c file is set to CERTIFY\_OOB\_BY\_DEFAULT\_CERT (as shown in the figure below), and the firmware is compiled and burned into the device.



Figure 33.75: Open CERTIFY\_BASE\_ENABLE





Operation Steps:

#else <mark>#defin</mark> #endif

Manual networking mode:

OOB BY DEFAULT CERT

The certificate icon will be displayed on the upper left corner when the device networking that needs to verify the certificate, and the network will be successfully after the certificate verification passes. If the current device is equipped with other certificates, the APP is validated by other certificates, or the certificate corresponding to the device is deleted, it will prompt that the verification fails: Provision – intermediate cert verify fail.



Figure 33.77: Android & iOS manual networking certify tips

Automatic Networking Mode

Compared to devices that do not verify certificates, the devices that need to verify certificates will display the certificate icon in the upper left corner of the device when networking in the Device Scan (Auto) interface. The networking will be successful after the certificate verification passes, while the verification fails will prompt Provision – intermediate cert verify fail.



Figure 33.78: Android & iOS auto-networking certify tips

#### 33.6.3.2 Generate and Import New Certificate for Networking

Generate new certificate pre-conditions:

- (1) The computer needs to be updated to the latest version of git and TortoseGit.
- (2) In the mesh\_config.h file, turn on the CERTIFY\_BASE\_ENABLE macro, and set the CERT\_TYPE in the certify\_base\_crypto.c file to CERTIFY\_OOB\_BY\_READING\_FLASH.

Generate a new certificate procedure:

- (1) Right-click and select Open Git Bash here in the sdk\_git\_lab/tools/bash-certifybase directory.
- (2) Type ./gen-root.bash hit enter to generate the "root.der" certificate.



#### Figure 33.79: Generated "root.der" certificate

# 🗉 Telink

#### (3) APP import root.der certificate

In Android version, import root.der certificate: at APP home page click Setting – root cert – click the upper right corner of the + sign – select the root.der file.

Setting V4.1.0.0@RC01	< Cert List 📋 +	< select cert(.der) :
E Manage Network >	3 [80:8e:e8:9f:9c:7b:94:2f:e77:d:2a:68:3c:fb :b3:4e:47:76:56:a3] X:	<pre>/storage/emulated/0/AAAA 4  root.der</pre>
GOB Database >	41db14a/01e1atc8d34a/233c33a640 6f815ae7ba9d52bb4625a123036995 Y 95a5139d35eedc1e65e33c128a69ab0d 037bff985aa23a62b3218baeb17e3f	
Root Cert 2	Time-NotBefore:Wed Oct 11 14:00:48 GMT+08:00 2023 Time-NotAfter:Sat Oct 08 14:00:48 GMT+08:00 2033 Alg:SHA256WITHECDSA (Default)	
<pre>Settings other settings &gt;</pre>		
$\mathbf{X}$		
<u>`</u>		
오 🖻 윪 🔅		

#### Figure 33.80: Import root.der certificate for Android

Import root.der certificate for iOS version: in iOS system, it needs to put the root.der file into TelinkSigMesh APP through iTunes, and then click Setting – Root Cert – Check the certificate file – Save (Note: you need to save it after checking it to make it effective).



Figure 33.81: Import root.der certificate for iOS

(4) Type ./gen-intermediate.bash – hit enter to generate the intermediate certificate





#### Figure 33.82: Generat intermediate certificates

(5) Edit UUID, CID, PID in gen-device.config



[rea] prompt = no distinguished\_name = req\_distinguished\_name [req\_distinguished\_name] #Country C=CN #State ST=ShangHai #Organization 0=Telink-Semi #Organization Unit OU=Telink #CN(common name): use device uuid, CID(Company ID, big endianness), PID(Product ID, big endianness) of unprovision node. CN=001BDC08-1021-0B0E-0A0C-000B0E0A0C01 BCID:0211 BPID:0001 CN=001BDC08-1021-0B0E #emailAddress emailAddress=support@telink-semi.com [v3\_req] authorityKeyIdentifier = keyid subjectKeyIdentifier = hash basicConstraints = CA:FALSE keyUsage = Certificate Sign, CRL Sign #TODO: static oob #2.25.234763379998062148653007332685657680359 = DER:31:7a:6f:16:58:44:72:74:15:10:33:62:5a:fb:c4:f1 certificatePolicies = critical, @pol [po1] policyIdentifier = 2.16.840.1.101.3.2.1.48.1

#### Figure 33.83: Change the UUID and the corresponding CID and PID

#### (6) Type ./gen-device.bash – hit enter to generate a 4K size "device.bin" device certificate.



Figure 33.84: Generate device certificate bin file

(7) Burn the device certificate "device.bin" to flash 78000 and use APP to network the device.

#### 33.6.3.3 Switch Certify Base Certificates

Long press the specified certificate, select Set As ROOT Cert to switch to the certificate, the purple certificate icon is the current effective certificate, gray is the saved but not effective certificate.

ios select cert

()

#### **Cert List** Î Last select name:root.der pubKey: EC Public Key [80:8e:e8:9f:9c:7b:94:2f:e7:9d:2a:68:3c:fb :b3:4e:47:76:56:a3] root-1.der X: 41dbf54a701efafc88d34a7233c383a640 6f815ae7ba9d52bb4625a123036995 root-3.der root-2.der Y: 95a3b139d35eedc1e65e33c128a69ab0d >> 037bff985aa23a62b3218baeb17e3f Time-NotBefore:Wed Oct 11 14:00:48 GMT-06:00 2023 Time-NotAfter:Sat Oct 08 14:00:48 root-4.der GMT+08:00 2033 Alg:SHA256WITHECDSA (Default) select action at: 1 pubKey: EC Public Key [88:37:7f:1d:c0:99: ee:55:70:9c:cc:6c:60:ba:65:f4:80:0d:fd:52] delete cert x: e4972a8cf5151da19feb6a69d8edc46e4a 01cd0d2299d046d683d69d9440a9a2 Y: Y: d72e369d7250ab1dca2a5bcba3ec043ab 19e1328e9c394f8abdcc25cd028e41 Yme-NotBefore:Sat Nov 04 14:58:37 GM 108:00 2023 GM 108:00 2033 YME SAUGUETED SA set as ROOT cert 256WITHECDSA Ala

#### Android select cert

Figure 33.85: Switch the certify base certificate interface

#### 33.6.3.4 Delete Certify Base Certificate

Long press the specified certificate and select Delete Cert to delete the certificate, and click the trash can button in the upper right corner to clear all certificates.

#### 33.6.4 Settings

In the Settings, we can process Enable Log, Enable Private Mode, Enable Provision Mode, Enable Subscription Level Servicemodel Id, Extend Bearer Mode, Use No -Oob Automatically, Share Import Complete Action, Online Status, Reset Settings.

**Note:** There are differences between Android and iOS, see below for details:



Figure 33.86: Android & iOS Setting/Settings interface

#### 33.6.4.1 Enable Log

#### Android APP:

Turn on Enable Log to record the log information when controlling the mesh, this item is turned off by default, and can be turned on as needed (please refer to the section 2.5).

#### iOS APP:

There is no Log switch, the app turns on the Log function by default (please refer to the section 2.5).

#### 33.6.4.2 Enable Privare Mode (Default Bound)

Enable Default Bound is the default binding mode, which needs to be supported by the device. In this mode, the app key binding process can be completed only if the app key add is executed successfully, and the device will automatically bind the app key to all the modes that need to be bound.

#### 33.6.4.3 Provision Mode

In Provision Mode, the default setting is Normal (Selectable) manual networking mode (please refer to section 1.1), Normal (Auto) automatic networking mode (please refer to section 1.2), remote provision and fast provision, which can be turned on according to your needs, and the following is an introduction to remote provision and fast provision.

#### (1) Remote Provision



Remote provision is to add multi-hop range devices one by one when networking, able to add devices at a longer distance, and with relay function.

Specific operation: Normal (selectable or auto) network a device that supports Remote Provision (i.e., open the MD\_REMOTE\_PROV macro) – App home page click on the setting – settings – Select Remote Provision in Provision Mode – Click the + sign on the home page of the app to carry out Remote Provision.

**Note:** Remote provision is turned off by default and requires the device to open the MD\_REMOTE\_PROV macro.

#### (2) Fast Provison

In Fast provision batch networking mode, you can network multiple devices that are not networked within the multi-hop range at the same time. The device key used is generated according to certain rules based on the mac address and does not need to be assigned individually. Steps: App home page click on Setting – Settings – Provision Mode and select Fast Provision – at APP home page click on the + sign for Fast Provision.

**Note:** Fast provision is turned off by default and requires the device to open FAST\_PROVISION\_ENABLE macro.

#### 33.6.4.4 Enable Subscription Level Service model ID

Enable Subscription Level Servicemodel ID can support to enable the Level control function of the grouping, you can control the Lum Level, Temp Level, Hue Level, Sat Level of the grouping individually. Before grouping nodes, you need to turn on Extend SubscriptionLevel Service Model ID in APP Home – setting – settings (there is a note in section 3.2.3).

#### 33.6.4.5 Enable DLE Mode Extend Bearer

Enable DLE Mode Extend Bearer is an option for sending long packets, requires device support. When enabled, the maximum length of short packets at the access layer is changed from 11 to 225.

#### 33.6.4.6 Online Status

Online Status allows you to check whether the current connected-only device supports the Online Status function and report the status when the device status changes.

#### 33.6.4.7 Reset Settings

Reset settings restores all options in the settings screen to their default state.



# 34 Common API

This chapter introduces the commonly used APIs for mesh SDK development. For an introduction to the parameters of the APIs listed in this chapter, please refer to the comments on the API functions in mesh SDK.

# 34.1 Provisioning Callbacks

#### 34.1.1 Provision Event Callback

#### 34.1.1.1 void mesh\_node\_prov\_event\_callback(u8 evt\_code)

This function is callback function for provision event of mesh node in each provision state.

#### 34.1.1.2 u8 is\_provision\_success()

This function get whether the node is at provision success state.

#### 34.1.1.3 rf\_link\_light\_event\_callback (u8 status)

This function is callback function to define LED indication event, such as how to do LED indication when provision success.

#### 34.1.2 Provisioning Message Handle

#### 34.1.2.1 PB\_ADV

void mesh\_node\_rc\_data\_dispatch(pro\_PB\_ADV \*p\_adv)

This function is for a unprovisioned device to be provisioned through PB\_ADV bearer, and handle all messages during provision flow.if a provison message need to be assembled, it was assembled in mesh\_provison\_process() before.

#### 34.1.2.2 PB\_GATT

void dispatch\_pb\_gatt(u8 \*p ,u8 len )

This function is for a unprovisioned device to be provisioned through PB\_GATT bearer, and handle all messages during provision flow.if a provison message need to be assembled, it was assembled in pkt\_pb\_gatt\_data() before.

#### 34.2 Proxy Server API

#### 34.2.1 Provision Service

#### 34.2.1.1 Int pb\_gatt\_Write (void \*p)

This function server to process Proxy PDU of provision service from GATT master, such as cell phone.

#### 34.2.2 Proxy Service

#### 34.2.2.1 Int proxy\_gatt\_Write(void \*p)

This function server to process Proxy PDU of proxy service from GATT master, such as cell phone.

### 34.3 Configuration Callbacks API

This section introduce APIs of all opcodes of configuration server and client model.

Most of them are located within config\_model.c. and processing callback function when receiving a opcode are defined by mesh\_cmd\_sig\_func[].cb.

Take opcode of "APPKEY\_ADD" for example:

#### 34.3.1 Int mesh\_cmd\_sig\_cfg\_appkey\_set()

int mesh\_cmd\_sig\_cfg\_appkey\_set(u8 \*par, int par\_len, mesh\_cb\_fun\_par\_t \*cb\_par)

## 34.4 model\_enable

This section introduce APIs of all opcodes of other SIG server and client model.

all processing callback function when receiving a opcode are defined by mesh\_cmd\_sig\_func[].cb.

#### 34.4.1 MD\_SAR\_EN

SAR Configuration Server model and client model. SAR means segmentation and reassembly.

#### 34.4.2 MD\_ON\_DEMAND\_PROXY\_EN

On Demand Private Proxy Server model and client model.

#### 34.4.3 MD\_OP\_AGG\_EN

Opcodes Aggregator Server model and client model.

#### 34.4.4 MD\_LARGE\_CPS\_EN

Large Composition Data Server model and client model.

#### 34.4.5 MD\_SOLI\_PDU\_RPL\_EN

Solicitation PDU RPL Configuration Server model and client model. RPL means Replay Protection List.

#### 34.4.6 MD\_DF\_CFG\_SERVER\_EN and MD\_DF\_CFG\_CLIENT\_EN

directed forwarding server model and client model.

#### 34.4.7 MD\_SBR\_CFG\_SERVER\_EN and MD\_SBR\_CFG\_CLIENT\_EN

subnet bridge server model and client model.

#### 34.4.8 MD\_REMOTE\_PROV

Remote Provisioning Server model and client model.

#### 34.4.9 MD\_PRIVACY\_BEA

Mesh Private Beacon Server model and client model

#### 34.4.10 MD\_BATTERY\_EN

Generic Battery Server and client model

#### 34.4.11 MD\_LOCATION\_EN

Generic Location Server and client model

#### 34.4.12 MD\_LEVEL\_EN

Generic Level Server and client model

#### 34.4.13 MD\_DEF\_TRANSIT\_TIME\_EN

Generic Default Transition Time Server and client model

#### 34.4.14 MD\_POWER\_ONOFF\_EN

Generic Power OnOff Server model and client model

#### 34.4.15 MD\_SCENE\_EN

Scene Server and client model

#### 34.4.16 MD\_TIME\_EN

Time Server model and client model

#### 34.4.17 MD\_SCHEDULE\_EN

Scheduler Server model and client model

#### 34.4.18 MD\_SENSOR\_EN

Sensor Server model and client model

#### 34.4.19 MD\_MESH\_OTA\_EN

device firmware update Server model and client model Mesh Binary Large Object Transfer Server model and client model

#### 34.4.20 MD\_LIGHTNESS\_EN

Light Lightness Server model and client model

#### 34.4.21 MD\_LIGHT\_CONTROL\_EN

Lighting control models model and client model

#### 34.4.22 LIGHT\_TYPE\_CT\_EN

Light CTL Server model and client model

#### 34.4.23 LIGHT\_TYPE\_HSL\_EN

Light HSL Server model and client model

#### 34.4.24 LIGHT\_TYPE\_XYL

Light xyL Server model and client model

#### 34.4.25 LIGHT\_TYPE\_POWER

Generic Power Level Server model and client model

#### 34.4.26 MD\_PROPERTY\_EN

Generic User Property Server model and client model Generic Admin Property Server model and client model Generic Manufacturer Property Server model and client model Generic Client Property Server model and client model

# 34.5 Light CT and RGB PWM Output API

#### 34.5.1 Void light\_dim\_refresh(int idx)

Refresh the light status once the current lightness, CT and HSL, etc is changed, include during transition process.

In this function, user can get the lightness and CT, HSL, etc, and then user can redefine how to driver PWM output which is depend on hardware, if it is needed.

## 34.6 Vendor Model Client and Server API

This section introduce APIs of all opcodes of other Vendor server and client model.

all processing callback function when receiving a opcode are defined by mesh\_cmd\_vd\_func[].cb.

VENDOR\_OP\_MODE\_SEL is set to VENDOR\_OP\_MODE\_DEFAULT as default. Include model of VEN-DOR\_MD\_LIGHT\_S which is server model and VENDOR\_MD\_LIGHT\_C which is client model

# 34.7 Firmware Update and Blob Transfer API

This section introduce APIs of all opcodes of firmware update and blob transfer server and client model. all processing callback function when receiving a opcode are defined by mesh\_cmd\_sig\_func[].cb. And they are enable by set MD\_MESH\_OTA\_EN to 1.

reint semiconductor

# 35 QA

This section contains some frequently asked questions.

#### Q1. Where is the callback for successful networking?

Inside the LED indication callback function rf\_link\_light\_event\_callback(), use the LGT\_CMD\_SET\_MESH\_INFO branch. LGT\_CMD\_SET\_MESH\_INFO which is the same as LGT\_CMD\_PROV\_SUC\_EVE, indicates the flashing event on the light node side, after the provision is successful.

#### Q2. How to determine if a node is in a provision success state?

is\_provision\_success(); Returns 1 to indicate that the app has been provisioned. Note that this refers to the provisioning process, not include the subsequent app key add and key bind processes.

There is no single way to judge the completion of a key bind that is applicable to all scenarios. This is because logically, it is not necessary to perform app key add and key bind immediately after the app networking process. If we take our app process as an example, after the app networking process is completed, app key add and key bind will be performed immediately after the app key add and key bind, and at this point, we can judge node\_binding\_tick inside main\_loop() as follows. tick, as shown below:

LPN is the judgement by the above logic.

#### Q3. Can I use the group number as the destination address when deleting or kicking out nodes?

To remove a device from the network, it is necessary to kick out one by one. The reason is that kick out sends the command "NODE\_RESET", which belongs to the configuration model and requires device key to encryption and decryption, and the device key of each node is not the same, so there is no way to send the command using multicast, unless the client customises the vendor command to do the remove action.

#### Q4. Do 16k and 32k retention need to switch cstartup.S and boot.link files?

Not required. See the "Startup file cstartup.s and link file boot.link" section for more information.

# **Q5.** After grouping the onoff model of a node and sending onoff set or lightness set command to the destination address 0xc000, the node is controlled normally but the vendor command is not, what is the reason?

See "Grouping Features and Share-modell" section for a detailed description of the theory analysis.

The UI operation of the sig\_mesh\_tool.exe only sends "Config Model Subscription Add" to the onoff model by default, so we need to send Subscription Add to the vendor model via INI, etc. in addition to the onoff

model. If you want to add private practices, such as sharing the group number between onoff model and vendor model, you can turn on SHARE\_ALL\_LIGHT\_STATE\_MODEL\_EN. See the code for this macro.

# **Q6.** What is the reason for the error "get ut tx buffer failed: tx segment busy" when sending long packets (segment) continuously?

When a long packet is sent, it needs to be segment. If the destination address is a unicast address, after sending all the sub-packets, you need to wait for the receiver to reply block Ack to see if all the segment packets have been received, and if not, the missing packet will be retransmited. Until the other party has finished receiving or exceeded the maximum number of retransmissions.

Currently, the SDK only supports one state machine to manage the segment process, so when the previous segment sends a long packet without completing it, the send packet function sends a long packet again, and it will report an error "get ut tx buffer failed: tx segment busy".

Even if the destination address is a multicast address, you need to wait until all the sub-packets have been sent before you are allowed to send the next long packet. Because the process of sending packets is to send a packet first, and when the packet is finished (about 200ms), then send the next packet, until all the packets are sent, then the segment busy will be cleared to zero. The reason why we don't continuously press the packets into the send fifo is that our tx buffer is not set that big, and if we continuously press it, the buffer will probably be insufficient to cause an error.

The suggested solution is to judge the return value of the send packet function, if it is not 0, wait for some time before sending, or call is\_busy\_mesh\_tx\_cmd() before sending to judge if it is currently in busy state. Or enable the private extended broadcast packet mode, see "Telink Customized Mode for Sending Mesh Messages via Extended Broadcast Package extend\_adv" for details.

#### Q7. How to get the rssi of the current message?

In a message callback function, such as mesh\_cmd\_sig\_g\_onoff\_set(), to read the global variable rssi\_pkt to get rssi of the current message, which is assigned a value before calling app\_event\_handler().

#### Q8. How to get the ttl of the current message?

In a message callback function, such as mesh\_cmd\_sig\_g\_onoff\_set(), to read cb\_par->p\_nw.ttl to get the ttl of the current message.

# **Q9.** Do the step resolution and number of steps in the transition time commands, such as on/off command, define the amount of change each time the light fades?

No, Transition Time just defines how long it takes for the light to finish changing. The amount of each transition is customised by the hardware, if you need to change it, just change the value of LIGHT\_ADJUST\_INTERVAL.

#### Q10. Why add AS\_PWM\_SECOND type inside driver?

In the mesh SDK, the PWM property only retains AS\_PWM and adds AS\_PWM\_SECOND, because we hope that when the customer modifies the PWM port of the light to another IO port, the customer only needs to modify the GPIO, and the rest is done automatically by the code, no need to configure the PWM ID, invert property and so on.

Since some pins support two PWM IDs, such as GPIO\_PC1, GPIO\_PC4, GPIO\_PC5 of B85m. Therefore, when customers configure the PWM attribute in mesh SDK, they need to check the GPIO table in the datasheet, if the GPIO does not belong to one of these 3 pins, then all of them will be set to AS\_PWM, because all of



them have only one PWM index for use. If the GPIO belongs to one of these three, then if you select the first PWM index, then set it to AS\_PWM, if you select the second PWM index, then set it to AS\_PWM\_SECOND.

#### Q11. Which compilation option should I choose for low-power devices?

For lower power product, if you want to only receive commands during the provision process or during receive mode which is actively entered, and other times just need to send command, it is recommended to use the 8258\_mesh\_Switch compilation option.

If you want to receive data at irregular intervals, i.e. another node may send data from time to time and require a low-power node to receive the data, then you need to use the LPN mode defined in the SIG mesh spec, which corresponds to the compilation option 8258\_mesh\_LPN, and which requires a Friend node. You can also consider the private spirit LPN compilation option, which does not require a Friend node, but requires that the sender sends commands continuously for 1 second at short intervals. This is because spirit LPNs wake up periodically for a short period of time to receive data. The power consumption is higher than standard mesh LPN. For details, please refer to Spirit LPN.

#### Q12. Suggested improvement for flash over 192KB?

When the firmware is optimised and still exceeds 192KB, the first recommendation is to replace the chip with a flash of 1MB.

If you don't want to change it, and you don't exceed it by much, say a dozen k sizes, you can consider the following optimisation directions at this point:

- Rearrange the flash map and remove the parameter areas corresponding to some unused functions (but evaluate whether they will be used later), such as FLASH\_ADR\_MD\_TIME\_SCHEDULE, which is not used by default. FLASH\_ADR\_MD\_LIGHT\_LC, FLASH\_ADR\_MD\_SENSOR, FLASH\_ADR\_MD\_LIGHT\_HSL, FLASH\_ADR\_MD\_PROPERTY(FLASH\_ADR\_MD\_DF\_SBR), FLASH\_ADR\_MD\_G\_POWER\_ONOFF, FLASH\_ADR\_MD\_SCENE, FLASH\_ADR\_MESH\_TYPE\_FLAG, FLASH\_ADR\_MD\_MESH\_OTA, FLASH\_ADR\_MD\_MISC\_PA
- The md5 algorithm for removing device uuid, the specific function is uuid\_create\_md5\_from\_name of uuid\_create\_by\_mac(), which is about 2k.

#### Note:

After rearranging the flash, you need to modify FW\_SIZE\_MAX\_K and check if factory\_reset() needs to be adjusted.

#### Q13. How do we get a node's provisioning information?

The provisioning information mainly contains netkey, device key, appkey, iv index, unicast address, which includes:

```
typedef struct{
    u8 net_work_key[16];
    u16 key_index;
    union{
        mesh_ctl_fri_update_flag_t prov_flags;
        u8 flags;
    };
    u8 iv_index[4];
    u16 unicast_address;
```



}

```
u8 device_key[16];
```

u8 app\_key[16];

We can get netkey, device key, appkey from variable mesh\_key; get iv index from iv\_cur in iv\_idx\_st; get unicast address from ele\_adr\_primary.

#### Q14. How do we send a customized advertise packet?

To send a customized advertise packet, you can change the data where p points to in the gatt\_adv\_prepare\_handler function, such as pre\_set\_beacon\_to\_adv(), which returns 1 to send, or returns 0 indicating of no data to send.

#### Q15. Why is the maximum number of schedulers 16?

Because the index field size of Scheduler in mesh model spec occupy only four bits, so the maximum number of schedulers is 16. Please refer mesh\_model\_spec for details, such as section 5.2.3.4 of "MshMDL\_v1.1.pdf". Note that each node supports 16 schedulers, rather than the entire mesh network having only 16 schedulers, so it is also sufficient.

#### **Q16.** How do we delete scheduler?

The scheduler can be removed by sending the Scheduler Action Set command and setting the action field of the Scheduler to SCHD\_ACTION\_NONE which is 0xOF.

typedef	struct{												
u64	valid_flag_o	r_	idx	: 4	;	//	fla	g: I	when	save;	index:	in m	nessage
u64	уеаг	:	7;										
u64	month	:	12;										
u64	day	:	5;										
u64	hour	:	5;										
u64	minute	:	<mark>6</mark> ;										
u64	second	:	<mark>6</mark> ;										
u64	week	:	7;	//	bit0	тес	ins i	mon	day,				
u64	action	:	4;										
u64	trans_t	:	8;	//	tran	siti	ion	time	9				
u16	<pre>scene_id;</pre>												
u8 r	and_hour;												
u8 r	and_min;												
u8 r	and_sec;												
u8 r	sv;												
}schedul	ler_t;												