DA14585/DA14531

External Processor Interface

This document provides all necessary information to create GTL-based applications. It includes detailed description of the Generic Transport Layer protocol (GTL), the messages exchanged as well as sequence flow diagrams for protocol operations. This document applies to DA14585/53x Renesas SoCs and covers changes for other devices.

Contents

Figures

Tables

1. Terms and Definitions

Bluetooth® LE	Bluetooth® Low Energy (Bluetooth Smart)
DTM	Direct Test Mode
GAP	Generic Access Profile
GAPC	Generic Access Profile Controller
GAPM	Generic Access Profile Manager
GATT	Generic Attribute Profile
GTL	Generic Transport Layer protocol
IAS	Immediate Alert Service
LE	Low Energy
LE_PSM	LE Protocol/Service Multiplexer
LECB	LE Credit Based
LLS	Link Loss Service
LTK	Long-term Key
MITM	Man in the Middle
MTU	Maximum Transmission Unit
SDK	Software Development Kit
TL	Transport Layer (standard 4-wire UART or other)
TXPS	TX Power Service

2. References

[1] Bluetooth SIG, Bluetooth Core Specification v5.0.

[2] DA14585, Datasheet, Renesas Electronics.

[3] UM-B-119, DA14585/DA14531 Software Platform Reference Manual, Renesas Electronics.

[4] Implementing Bluetooth LE Data Pumps White Paper.

[5] DA14531, Datasheet, Renesas Electronics.

3. Introduction

This manual describes how an external processor system, called "GTL Host" hereafter, communicates with DA14585/531 using the GTL messaging protocol; a GAP-like protocol, natively supported in DA14585/531. Using only GTL, the DA14585/531 application can be configured and controlled. All Bluetooth® LE operations, like advertising, disconnection, connection confirmation, security request and pairing/encryption, can be controlled by the GTL Host.

4. DA14585/531 GTL Project Software Architecture

The system consists of two separate devices:

- **GTL Host**
- **Bluetooth LE application.**

The Bluetooth LE application can be based on any Renesas Electronics' chip like DA14585/531 running GTL-enabled software (GTL project). The "GTL Host" device can be any uC which supports a 4-wire UART interface. This physical connection is called Transport Layer (TL).

The GTL Host contains the user application that communicates over the TL with the Bluetooth LE application. The GTL Host controls the Bluetooth LE application and initiates Bluetooth LE operations using messages referred also as commands.

The Bluetooth LE application communicates with the GTL Host with the use of messages that either are status updates or notifications. Data exchange like Bluetooth LE indications/notifications, GATT read and write operations are all executed using command and event messages. Bluetooth LE services and profiles usually are implemented in the GTL host while the Bluetooth LE application contains all the Bluetooth LE Stack functionality up to the GATT and GAP layers. Figure 1 shows an example of such a system.

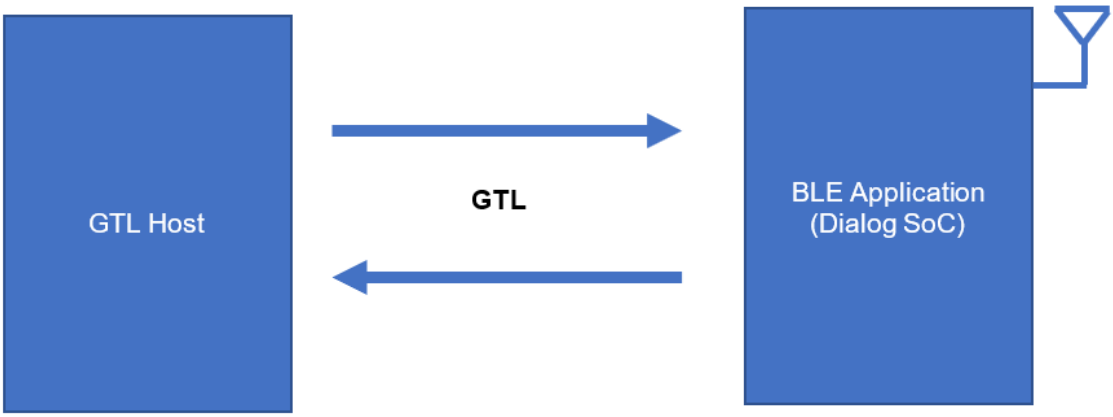


Figure 1. Example of a GTL-based system

The projects of the DA14585/531 SDK6 which support the GTL functionality are the prox_reporter_ext and prox_monitor_ext projects located in the projects\target_apps\ble_examples\prox_reporter_ext and projects\target_apps\ble_examples\prox_monitor_ext folders respectively. The "empty_template_ext" project, located in projects\target_apps\template\empty_template_ext, can also pose as a clean starting point for a GTL based application.

The software architecture of the GTL Project is shown in Figure 2 and is outlined in Ref. [4].

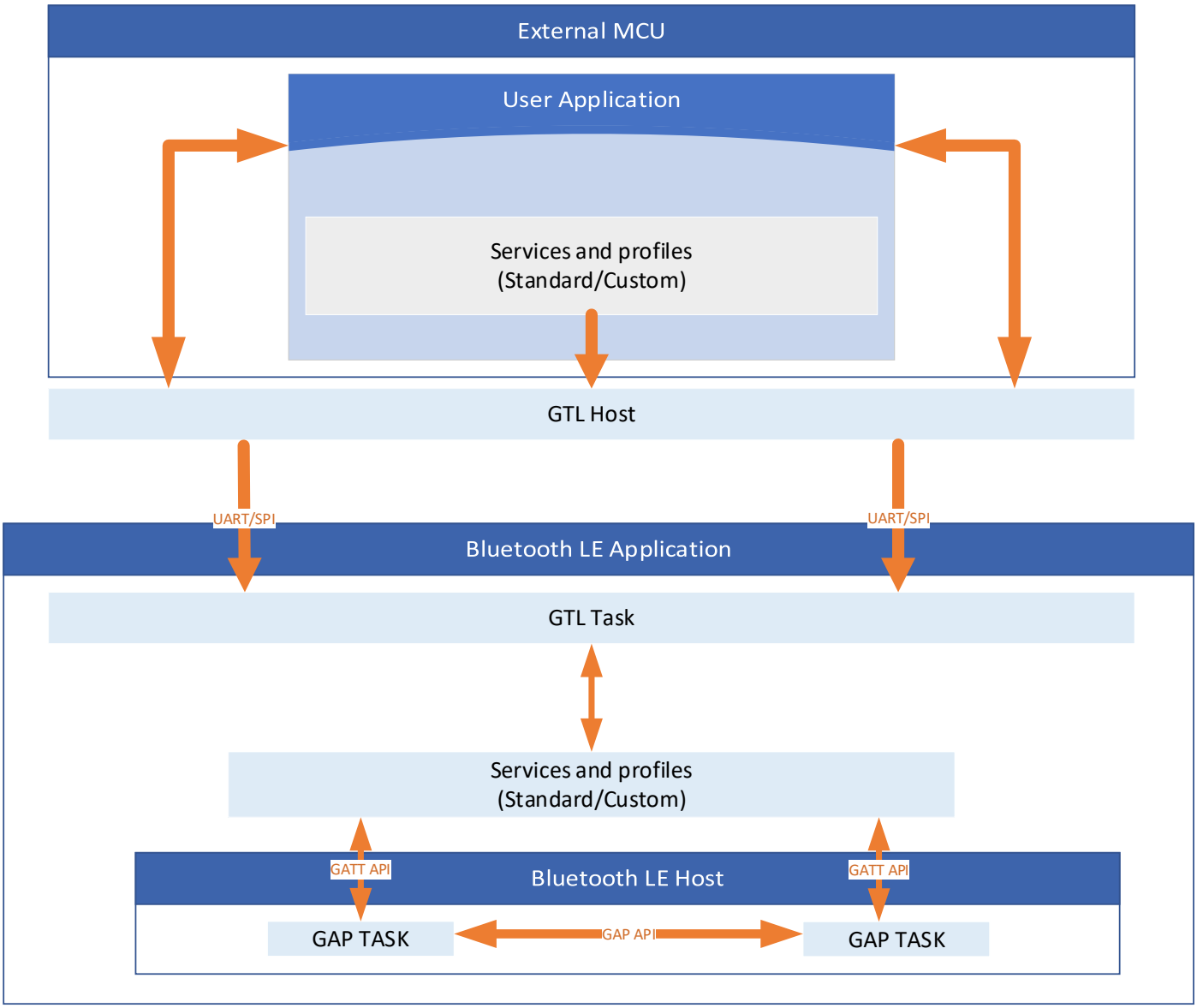


Figure 2. DA14585/531 GTL project software architecture

5. External Processor Interface

5.1 Message Format

The messages sent/received to/from the external processor follow the DA14585/531 GTL protocol format.

The format of the GTL message in DA14585/531 are outlined in Figure 3 and Table 1.

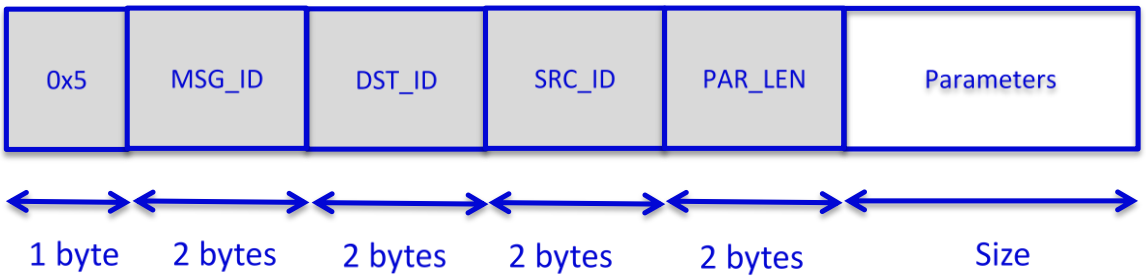


Figure 3. GTL message format

Table 1. GTL message parameters

Field	Bytes	Description	
GTL initiator	1	Proprietary GTL header Always set to 0x05	
MSG_ID	2	Message Identification A valid Message ID of a TASK describing a command, a status or a notification. API message IDs are listed in Section 7.	
DST_ID	2	Destination Task Identifier	A task ID of the Bluetooth LE stack tasks (that is, TASK_ID_GAPM, TASK_ID_GAPC, and others) or the standard GAP API (GAP-based API) The standard DA14585/531 GAP API is used for the standard Bluetooth LE procedures, like discovering Bluetooth devices, establishing connections, managing links, and establishing security.
SRC_ID	2	Source Task Identifier	
PAR_LEN	2	Message Parameter Length The size, in bytes, of the Parameters field.	
Parameters	N	Message Parameters The format of this field depends on the MSG_ID API message formats are described in Section 5.2 and Section 5.3.	

The DST_ID of the message must be created using the KE_BUILD_ID macro defined in DA14585/531 SDK header file "ke_msg.h". The type parameter of the macro is the task (GAPM, GAPC, GATTC, GATTM) and the index parameter is the connection index for the connection with the peer device. The connection index can be extracted by the SRC_ID of the incoming message by using the KE_IDX_GET macro defined in the same header file.

5.2 Messages

5.2.1 Standard GAP API (GAP-based API)

The standard DA14585/531 GAP API in Ref. [3] is used for the standard Bluetooth LE procedures like discovery of Bluetooth devices, connection establishment, link management, and security establishment.

5.3 Examples and Diagrams

5.3.1 GAP Initialization

The GAP initialization procedure must be executed at system start-up to configure Bluetooth LE stack operations. After this operation, the device is then usually set in an advertising state for GAP peripheral role or in

a scanning state for central GAP role. The system start-up of DA14585/531 is indicated with the reception of a GAPM_DEVICE_READY_IND message sent by GAPM task of the Bluetooth LE application.

The initialization of a Bluetooth LE device operating in GAP peripheral role and always advertising in general discoverable undirected mode is outlined in Figure 4.

The sequence of messages exchanged by the GTL Host and the Bluetooth LE application is shown in Figure 4.

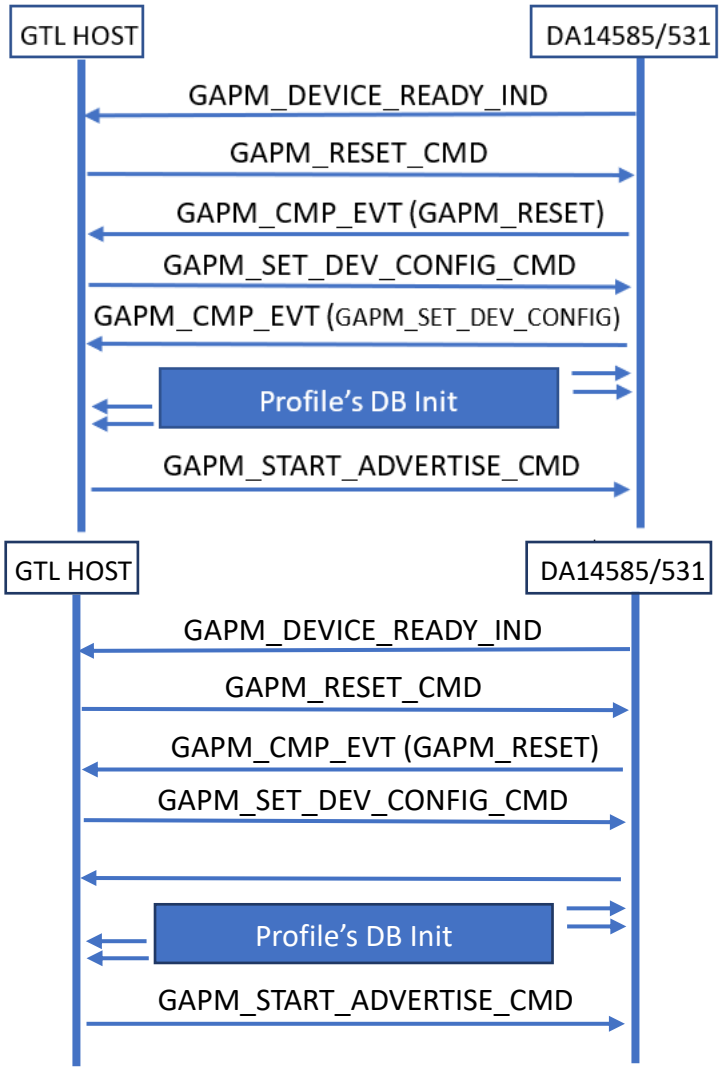


Figure 4. Device initialization procedure

The steps of the GAP initialization procedure are described in more details in the following subsections. For more information on database creation, see Section 5.3.8.

5.3.1.1 Step 1: Reception of the Device Ready Indication

The GAPM task sends a GAPM_DEVICE_READY_IND message at the completion of DA14585/531 stack initialization. It is sent each time the DA14585/531 application is restarted after power-on or reset. The message must be always responded with the GAPM_RESET_CMD message.

Table 2. GAPM_DEVICE_READY_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_DEVICE_READY_IND	0x0D01

Parameters	Bytes	Description	Data
Header			
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	No parameters	0x0000
Message parameters			
{None}			

Table 3. GAPM_DEVICE_READY_IND example

Message stream (GAP initialization – step 1)		
Symbolic message structure reference can be found in Table 2.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x0D	GAPM_DEVICE_READY_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x00 0x00	No parameters
Message string		
0x05 0x01 0x0D 0x10 0x00 0x0D 0x00 0x00 0x00		

5.3.1.2 Step 2: Send Reset Command

The GTL Host should always respond to GAPM_DEVICE_READY_IND with a GAPM_RESET_CMD message.

Table 4. GAPM_RESET_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_RESET_CMD	0x0D02
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	1	0x0001
Message parameters			
operation	1	GAPM_RESET	0x01 (See Table 374)

Table 5. GAPM_RESET_CMD example

Message stream (GAP initialization – step 2)		
Symbolic message structure reference can be found in Table 4.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x02 0x0D	GAPM_RESET_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x01	GAPM_RESET
Message string		
0x05 0x02 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0X01		

5.3.1.3 Step 3: Reception of Completion Event Message for the Reset Operation

Following the completion of the GAPM reset procedure, the DA14585/531 GAPM sends a completion event message (GAPM_CMP_EVT) of GAPM_RESET operation.

Table 6. GAPM_CMP_EVT example

Message stream (GAP initialization – step 3)		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x01	GAPM_RESET
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x01 0x00		

5.3.1.4 Step 4: Send Device Configuration Message

The GTL Host must send a GAPM_SET_DEV_CONFIG_CMD to set the Bluetooth LE stack configuration.

Table 7. GAPM_SET_DEV_CONFIG_CMD symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_SET_DEV_CONFIG_CMD	0x0D04
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	44	0x002C
Message parameters			
operation	1	**0x03**: GAPM_SET_DEV_CONFIG **0x1E:** GAPM_SET_SUGGESTED_DFLT_LE_DATA_LEN	See Table 374
role	1	**0x00:** GAP_ROLE_NONE: No role set yet **0x01:** GAP_ROLE_OBSERVER: Observer role **0x02:** GAP_ROLE_BROADCASTER: Broadcaster role **0x05:** GAP_ROLE_CENTRAL: Master/Central role **0x0A:** GAP_ROLE_PERIPHERAL: Peripheral/Slave role **0x0F:** GAP_ROLE_ALL: Device has all role, both peripheral and central	Gap role
renew_dur	2	Duration before regenerate device address when privacy is enabled	
addr	6	The BD address	
irk	16	Device IRK used for resolvable random BD address generation (LSB first)	

<table>
<tr><th>Parameters</th><th>Bytes</th><th>Description</th><th>Data</th></tr>
<tr><td colspan="4">Header</td></tr>
<tr><td>addr_type</td><td>1</td><td>0x00: GAPM_CFG_ADDR_PUBLIC: Static address
0x01: GAPM_CFG_ADDR_PRIVATE: Private Static Address
0x02: GAPM_CFG_ADDR_PRIVACY: Generated using Privacy feature
0x04: GAPM_CFG_ADDR_PRIVACY_CNTL: Generated using Privacy feature in controller</td><td>Device Address Type</td></tr>
<tr><td>att_cfg</td><td>1</td><td>
<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>DBG</td><td>RFU</td><td>SC</td><td>PCP</td><td colspan="2">APP_PERM</td><td colspan="2">NAME_PERM</td></tr>
</table>

Bit [0-1]: Device Name write permission requirements for peer device

0x00: GAPM_WRITE_DISABLE
0x08: GAPM_WRITE_ENABLE
0x10: GAPM_WRITE_UNAUTH
0x18: GAPM_WRITE_AUTH

Bit [2-3]: Device Appearance write permission requirements for peer device

0x00: GAPM_WRITE_DISABLE
0x08: GAPM_WRITE_ENABLE
0x10: GAPM_WRITE_UNAUTH
0x18: GAPM_WRITE_AUTH

Bit [4]: Slave Preferred Connection Parameters present:

1: Yes
0: No

Bit [5]: Service change feature present in GATT attribute database:

1: Yes
0: No

Bit [6]: CoC zero credit behavior
Bit [7]: Enable Debug Mode:

1: Yes
0: No

</td><td>Attribute database configuration</td></tr>
<tr><td>gap_start_hdl</td><td>2</td><td>GAP service start handle</td><td></td></tr>
<tr><td>gatt_start_hdl</td><td>2</td><td>GATT service start handle</td><td></td></tr>
<tr><td>max_mtu</td><td>2</td><td>Maximum MTU [23...512]</td><td></td></tr>
<tr><td>max_mps</td><td>2</td><td>Maximum MPS [23...512]</td><td></td></tr>
<tr><td>att_cfg_</td><td>2</td><td>Not used - must be set to zero</td><td>Not used</td></tr>
<tr><td>max_txoctets</td><td>2</td><td>Maximum TX octets [27…251]</td><td></td></tr>
<tr><td>max_txtime</td><td>2</td><td>Maximum TX time [328...2120]</td><td></td></tr>
<tr><td>priv1_2</td><td>1</td><td>Privacy 1.2</td><td></td></tr>
<tr><td>{padding}</td><td>1</td><td>{padding} Ignored. Any value is valid</td><td></td></tr>
</table>

Table 8. GAPM_SET_DEV_CONFIG_CMD example

Message stream (GAP initialization – Step 4)		
Symbolic message structure reference can be found in Table 7.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x04 0x0D	GAPM_SET_DEV_CONFIG_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x2C 0X00	44 bytes
operation	0x03	GAPM_SET_DEV_CONFIG
role	0x0A	GAP_ROLE_PERIPHERAL
renew_dur	0x00 0x00	
addr	0x00 0x00 0x00 0x00 0x00 0x00	
irk	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	
addr_type	0x00	GAPM_CFG_ADDR_PUBLIC
att_cfg	0x20	GAPM_MASK_ATT_SVC_CHG_EN (Service change feature present in GATT attribute database)
gap_start_hdl	0x00 0x00	Handle 0
gatt_start_hdl	0x00 0x00	Handle 0
max_mtu	0x00 0x02	0x200 = 512
max_mps	0x00 0x00	0
att_cfg_	0x00 0x00	0
max_txoctets	0xFB 0x00	251
max_txtime	0x48 0x08	2120
priv1_2	0x00	0
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x04 0x0D 0x0D 0x00 0x10 0x00 0x2C 0x00 0x03 0x0A 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x20 0x00 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00 0x00 0xFB 0x00 0x48 0x08 0x00 0x00		

5.3.1.5 Step 5: Reception of Completion Event for the Device Configuration Operation

The DA14585/531 GAPM sends a completion event message (GAPM_CMP_EVT) for the operation GAPM_SET_DEV_CONFIG_CMD.

Table 9. GAPM_CMP_EVT example

Message stream (GAP initialization – step 5)		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x03	GAPM_SET_DEV_CONFIG
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x03 0x00		

In case the status field is not GAP_ERR_NO_ERROR, the GTL Host should retry device configuration based on the received error code.

5.3.1.6 Step 6: Adding the DISS Profile Task

See Section 6.2.1.

5.3.1.7 Step 7: Send Start Advertise Command

Upon completion of the database initialization, the DA14585/531 device is ready to enter connectable mode. To achieve this, the GAPM_START_ADVERTISE_CMD must be sent to the DA14585/531 GAPM task.

Table 10. GAPM_START_ADVERTISE_CMD symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_START_ADVERTISE_CMD	0x0D0D
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	82 bytes	0x0052
Message parameters			
op.code	1	**0x0C**: GAPM_ADV_NON_CONN Non-connectable advertising. **0x0D**: GAPM_ADV_UNDIRECT Undirected connectable advertising. **0x0E**: GAPM_ADV_DIRECT Directed connectable advertising. **0x0F**: GAPM_ADV_DIRECT_LDC Directed connectable advertising using Low Duty Cycle.	See Table 374

Parameters	Bytes	Description	Data
Header			
op.addr_src	1	**0x00**: GAPM_STATIC_ADDR Public or Private Static Address according to device address configuration. **0x01**: GAPM_GEN_RSLV_ADDR Generated resolvable private random address. **0x02**: GAPM_GEN_NON_RSLV_ADDR Generated non-resolvable private random address.	
op.state	2	Dummy data used to retrieve internal operation state (should be set to 0).	
intv_min	2	Minimum interval for advertising.	
intv_max	2	Maximum interval for advertising.	
channel_map	1	Advertising channels map.	**0x01**: ADV_CHNL_37_EN **0x02**: ADV_CHNL_38_EN **0x04**: ADV_CHNL_39_EN **0x07**: ADV_ALL_CHNLS_EN
info.host.mode	1	**0x00**: GAP_NON_DISCOVERABLE Non-discoverable advertising. **0x01**: GAP_GEN_DISCOVERABLE General discoverable advertising. **0x02**: GAP_LIM_DISCOVERABLE Limited discoverable advertising. **0x03**: GAP_BROADCASTER_MODE Broadcaster mode which is a non-discoverable and non-connectable mode.	Advertising mode
info.host.adv_filt_policy	1	**0x00**: ADV_ALLOW_SCAN_ANY_CON_ANY Allow both scan and connection requests from anyone. **0x01:** ADV_ALLOW_SCAN_WLST_CON_ANY Allow both scan requests from Whitelist devices only and connection req from anyone. **0x02**: ADV_ALLOW_SCAN_ANY_CON_WLST Allow both scan requests from anyone and connection requests from Whitelist devices only. **0x03**: ADV_ALLOW_SCAN_WLST_CON_WLST Allow scan and connection requests from Whitelist devices only.	
info.host.adv_data_len	1	Advertising data length. 3 bytes are reserved to set Advertising AD type flags and should not be set in advertising data.	Maximum 28 bytes
info.host.adv_data	31	Advertising data string.	Advertising data string must be compatible with the Bluetooth LE spec requirements. See Vol 3, Part C, section 11 in Ref. [1]
info.host.scan_rsp_data_len	1	Scan response data length.	Maximum 31 bytes

Parameters	Bytes	Description	Data
Header			
info.host.scan_rsp_data	31	Scan response data string.	Scan response data string must be compatible with Bluetooth LE spec requirements. See Vol 3, Part C, section 11 in Ref. [1]
info.host.peer_info.addr	6	BD address of the peer device.	Used only when the address type of the local peer is set to 0x04: GAPM_CFG_ADDR_PRIVACY_CNTL. See parameter addr_type in Section 5.3.1.4
info.host.peer_info.addr_type	1	Address type of the peer device	**0x00**: public **0x01**: random

Table 11. GAPM_START_ADVERTISE example

Message stream (GAP initialization – step 7)			
Symbolic message structure reference can be found in Table 10.			
Hex data			
Data token	**Hex data (LSB)**	**Comments**	
Initiator	0x05	GTL Initiator	
MSG_ID	0x0D 0x0D	GAPM_START_ADVERTISE_CMD	
DST_ID	0x0D 0x00	TASK_ID_GAPM	
SRC_ID	0x10 0x00	TASK_ID_GTL	
PAR_LEN	0x52 0x00	82 bytes	
op.code	0x0D	GAPM_ADV_UNDIRECT	
op.addr_src	0x00	GAPM_STATIC_ADDR	
op.state	0x00 0x00	0x00	
intv_min	0xC8 0x00	0.625 × 200 = 125 ms	
intv_max	0xC8 0x00	0.625 × 200 = 125 ms	
channel_map	0x07	Advertise using all channels	
info.host.mode	0x01	GAP_GEN_DISCOVERABLE	
info.host.adv_filt_policy	0x00	ADV_ALLOW_SCAN_ANY_CON_ANY	
info.host.adv_data_len	0x1B	27 bytes	
info.host.adv_data	0x07 0x03 0x03 0x18 0x02 0x18 0x04 0x18 0x12 0x09 0x44 0x69 0x61 0X6C 0x6F 0x67 0x50 0x45 0x52 0x20 0x44 0x41 0x31 0x34 0x35 0x38 0x35 0x00 0x00 0x00 0x00	0x07	Length: 7 bytes
		0x03	AD Type: Complete list of 16-bit Service Class UUIDs
		0x03 0x18	UUID: Link Loss Alert
		0x02 0x18	UUID: Immediate Alert
		0x04 0x18	UUID: TX Power
		0x12	Length:18 byte
		0x09	AD Type: Complete local name
		0x44	D
		0x69	i
		0x61	a
		0x6C	l

Message stream (GAP initialization – step 7)			
		0x6F	o
		0x67	g
		0x50	P
		0x45	E
		0x52	R
		0x20	
		0x44	D
		0x41	A
		0x31	1
		0x34	4
		0x35	5
		0x38	8
		0x35	5
		0x00 0x00 0x00 0x00	Padding up to the 31st byte (can be any value)
info.host.scan_rsp_data_len	0x0D	13 bytes	
info.host.scan_rsp_data	0x0C 0xFF 0xD2 0x00 0x53 0x61 0x6D 0x70 0x6C 0x65 0x20 0x23 0x31 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	0x0C	Length: 12 bytes
		0xFF	AD Type: Manufacturer Specific
		0xD2 0x00	Manufacturer ID (MSB – in this example, dialog ID:0x00D2 is used)
		0x53 0x61 0x6D 0x70 0x6C 0x65 0x20 0x23 0x31	Manufacturer-specific data (in this example, the text "Sample #1" is used
		0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	Padding up-to the 31st byte (can be any value)
info.host.peer_info.addr	0x00 0x00 0x00 0x00 0x00 0x00	0x000000000000	
info.host.peer_info.addr_type	0x00	Public address	
Message string			
0x05 0x0D 0x0D 0x0D 0x00 0x10 0x00 0x52 0x00 0x0D 0x00 0x00 0x00 0xC8 0x00 0xC8 0x00 0x07 0x01 0x00 0x1B 0x07 0x03 0x03 0x18 0x02 0x18 0x04 0x18 0x12 0x09 0x44 0x69 0x61 0X6C 0x6F 0x67 0x50 0x45 0x52 0x20 0x44 0x41 0x31 0x34 0x35 0x38 0x35 0x00 0x00 0x00 0x00 0x0D 0x0C 0xFF 0xD2 0x00 0x53 0x61 0x6D 0x70 0x6C 0x65 0x20 0x23 0x31 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			

5.3.2 Connection Sequence

5.3.2.1 On the Central Peer's Side

The connection sequence is initiated when the central issues **GAPM_START_CONNECTION_CMD**. This command supports four operations:

1. **GAPM_CONNECTION_DIRECT:** Direct connection operation

   Start a connection with a specific device using its Bluetooth LE address and address type.

2. **GAPM_CONNECTION_AUTO:** Automatic connection operation

   Automatic connection is used to perform a connection with a specified device. Before the connection procedure starts, the central must perform a scan (in observer mode) to look for available peer devices. As soon as a device is detected, the scanning stops and a direct connection process with the detected device is initiated.

3. **GAPM_CONNECTION_SELECTIVE**: Selective connection operation

   Selective connection can be also used to connect to a specific device. After a scan is executed and a peer device is detected, an advertise report is created in the application. The application must then select which device it connects to by sending the GAPM_CONNECTION_CFM message with specific connection parameters. When the GAPM_CONNECTION_CFM message is received, the scan procedure stops, and a direct connection starts.

4. **GAPM_CONNECTION_NAME_REQUEST:** Name Request operation

   This mode is used to discover the name of a specific peer device. A direct connection is established and the central executes a name request over GATT. When the name is acquired, the central disconnects without the application being involved.

A typical connection sequence for the central peer's side is shown in Figure 5.

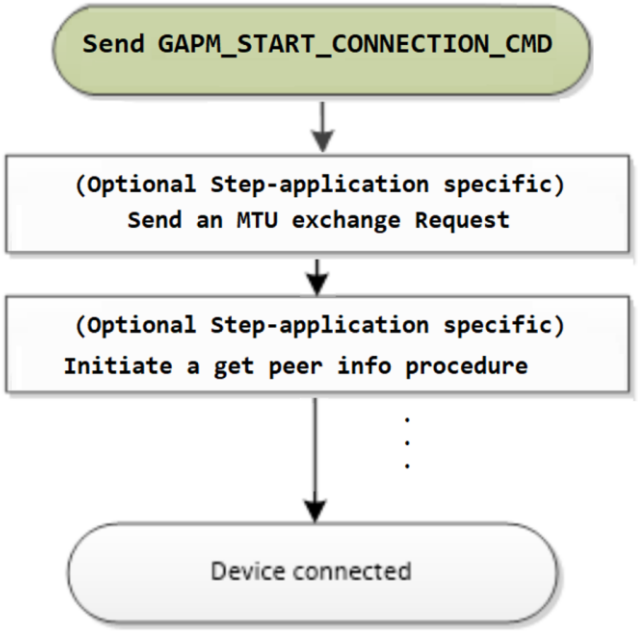


Figure 5. Connection sequence on a central side

Table 12. GAPM_START_CONNECTION_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_START_CONNECTION_CMD	0x0D11
DST_ID	2	TASK_ID_GAPM	0x000D

Parameters	Bytes	Description	Data
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Depending on number of peers	0xxxxx
Message parameters			
op.code	1	GAPM connection operation	**0x13**: GAPM_CONNECTION_DIRECT **0x14**: GAPM_CONNECTION_AUTO **0x15**: GAPM_CONNECTION_SELECTIVE **0x16**: GAPM_CONNECTION_NAME_REQUEST
op.addr_src	1	Own BD address type	**0x00**: GAPM_STATIC_ADDR: Public or Private Static Address according to device address configuration **0x01**: GAPM_GEN_RSLV_ADDR: Generated resolvable private random address **0x02**: GAPM_GEN_NON_RSLV_ADDR: Generated non-resolvable private random address
op.state	2	Dummy data used to retrieve internal operation state (should be set to 0)	0x00 0x00
scan_interval	2	Scan interval	Expressed in 0.625 ms slots
scan_window	2	Scan window size	Expressed in 0.625 ms slots
con_intv_min	2	Minimum connection interval	Expressed in 1.25 ms slots
con_intv_max	2	Maximum connection interval	Expressed in 1.25 ms slots
con_latency	2	Connection latency	
superv_to	2	Link supervision timeout	Multiply by 10 ms
ce_len_min	2	Minimum CE length	Expressed in 0.625 ms slots
ce_len_max	2	Maximum CE length	Expressed in 0.625 ms slots
nb_peers	1	Number of peer device information present in message	**0x01**: GAPM_CONNECTION_DIRECT or GAPM_CONNECTION_NAME_REQUEST operations. Greater than 0 for other operations
peers[0].addr.addr	6	6-byte array address value	
peers[0].addr_type	1	Address type of the device	**0x00:** Public **0x01:** Private random
peers[1].addr.addr	6	6-byte array address value	
peers[1].addr_type	1	Address type of the device	**0x00:** Public **0x01:** Private random
…		…	…
peers[n].addr.addr	6	6-byte array address value	
peers[n].addr_type	1	Address type of the device	**0x00:** Public **0x01:** Private random
{padding}	1	{padding} Ignored.	Any value is valid.

5.3.2.2 On the Peripheral Peer's Side

The connection sequence on a peripheral role device is initiated by the connection of a central peer on the device (GAPM_START_CONNECTION_CMD as shown in Section 5.3.2.1). The Bluetooth LE stack of the DA14585/531 (GAPC task) sends a GAPC_CONNECTION_REQ_IND message to the GTL Host, upon the connection event.

The connection sequence consists of the following parts:

- A standard sequence of messages exchanged upon the reception of the GAPC_CONNECTION_REQ_IND.
- (Optional, application specific) Send Security Request Command to the peer.
- (Optional, application specific) Identify whether the peer device is known (bonding data for the specific host exists) and retrieve the associated bonding information or proceed with bonding. The message sequence can include resolution of the random resolvable address of the peer device.
- (Optional, application specific) Other messages, including the following:
  - Start non-connectable advertising
  - Send MTU exchange request
  - Send parameters update request.

The connection sequence is shown in Figure 6.

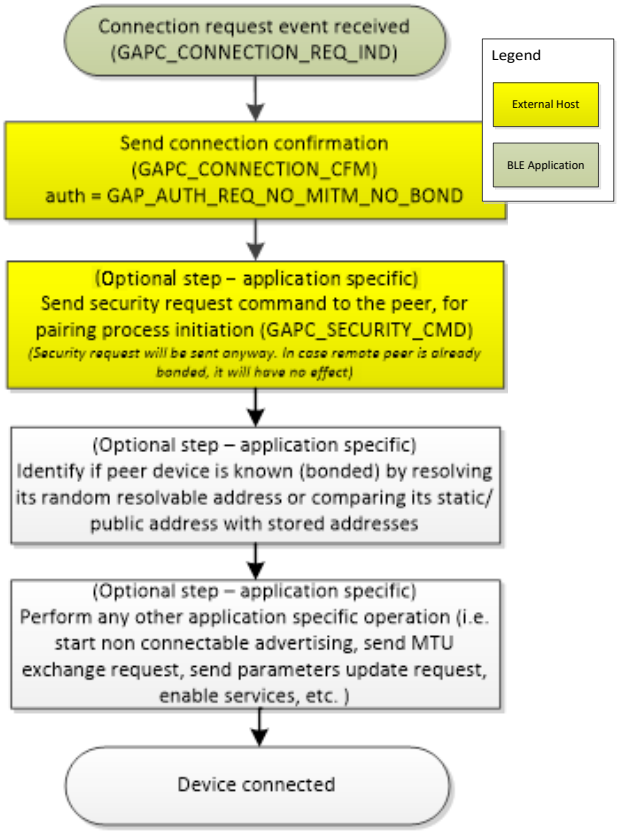


Figure 6. Connection sequence outline

Table 13. GAPC_CONNECTION_REQ_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_CONNECTION_REQ_IND	0x0E01
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)

Parameters	Bytes	Description	Data
Header			
PAR_LEN	2	16	0x0010
Message parameters			
conhdl	2	The handle of the current connection	0x0000
con_interval	2	Connection Interval	Expressed in 1.25 ms slots
con_latency	2	Connection latency	
sup_to	2	Supervision timeout for the connection	Multiply by 10 ms
clk_accuracy	1	Clock accuracy of the peer device	
peer_addr_type	1	The peer device type of address	**0x00**: ADDR_PUBLIC **0x01:** ADDR_RAND
peer_addr	6	The peer device BD address (LSB)	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

5.3.2.3 Connection Sequence on the Peripheral Side – Example with Security

Figure 7 shows the essential message exchange sequence of the connection procedure.

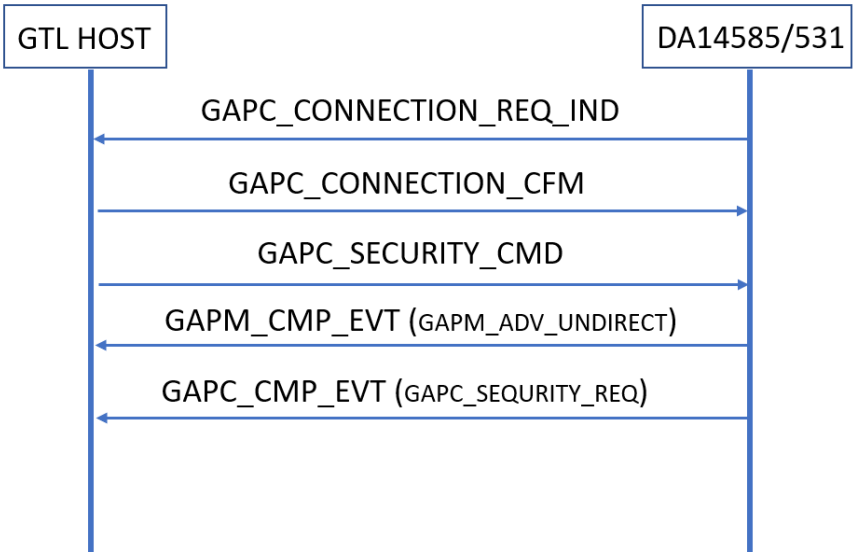


Figure 7. Connection event sequence – standard sequence

1. Step 1: Reception of a connection request.

   The central sends the GAPM_START_CONNECTION_CMD that creates a GAPC_CONNECTION_REQ_IND. This message is a request because it is waiting for GAPC_CONNECTION_CFM message to:

   - Set connection bond data.
   - Authentication and authorization link configuration.

Table 14. GAPC_CONNECTION_REQ_IND example #1

Message stream (connection sequence on a peripheral's side – step 1)		
Symbolic message structure reference can be found in Table 13.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x0E	GAPC_CONNECTION_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x10 0x00	16 bytes
conhdl	0x00 0x00	handle 0
con_interval	0x24 0x00	0x24 = 36 (36 × 1.25 ms = 45 ms)
latency	0x00 0x00	No latency
sup_to	0xF4 0x01	0x01F4 = 500 (500 x 10 = 5000 ms = 5 s)
clk_accuracy	0x00	Clock accuracy of the peer device
peer_addr_type	0x00	0: PUBLIC
peer_addr	0x02 0xEE 0x70 0xCA 0xEA 0x80	Peer BD address: 0x80EACA70EE02
Message string		
0x05 0x01 0x0E 0x10 0x00 0x0E 0x00 0x10 0x00 0x00 0x00 0x24 0x00 0x00 0x00 0xF4 0x01 0x00 0x00 0x02 0xEE 0x70 0xCA 0xEA 0x80		

2. Step 2: Transmission of the message for connection confirmation.

   The GTL Host sends the connection confirmation command (GAPC_CONNECTION_CFM) to the DA14585/531. The peer device is not known, so the authentication field is set to GAP_AUTH_REQ_NO_MITM_NO_BOND. The confirmation message then enables the attribute database and security manager to process requests from the peer device. Before sending the confirmation message, the application can perform address resolution to ascertain whether it is a known device and start some services.

Table 15. GAPC_CONNECTION_CFM symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_CONNECTION_CFM	0x0E02
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	44 bytes	0x002C
Messenger parameters			
lcsrk	16	Local CSRK value	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
lsign_counter	4	Local signature counter value	0x00 0x00 0x00 0x00
rcsrk	16	Remote CSRK value	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
rsign_counter	4	Remote signature counter value	0x00 0x00 0x00 0x00
auth	1	Authentication requirements	See Table 376

Parameters	Bytes	Description	Data
svc_changed_ind_enable	1	Enable sending of Service Changed Indication (Note 1)	**0x00**: Disabled **0x01**: Enabled
{padding}	2		{padding} Ignored. Any value is valid.

Note 1 It is a prerequisite to include the Service Changed Indication in the GATT database to enable the Service Changed Indication. See parameter att_cfg in Section 5.3.1.4 (bit 5 must be set to 1).

Table 16. GAPC_CONNECTION_CFM example

Message stream (GAPC_CONNECTION_CFM)		
Symbolic message structure reference can be found in Table 15.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x02 0x0E	GAPC_CONNECTION_CFM
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x2C 0x00	44 bytes
lcsrk	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	Local CSRK: 0x00000000000000000000000000000000
lsign_counter	0x00 0x00 0x00 0x00	0
rcsrk	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	Remote CSRK: 0x00000000000000000000000000000000
rsign_counter	0x00 0x00 0x00 0x00	0
auth	0x00	No MITM No Bonding (0x00) GAP_AUTH_REQ_NO_MITM_NO_BOND (= GAP_AUTH_NONE)
svc_changed_ind_enable	0x00	Sending of Service Changed Indication disabled.
{padding}	0x00 0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x02 0x0E 0x0E 0x00 0x10 0x00 0x2C 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

3. Step 3: Transmission of the message to initiate the security request procedure.

   The GTL Host sends the security request command (GAPC_SECURITY_CMD) to the Bluetooth LE application to initiate security.

Table 17. GAPC_SECURITY_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_SECURITY_CMD	0x0E1A
DST_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
SRC_ID	2	TASK_ID_GTL	0x0010

Parameters	Bytes	Description	Data
PAR_LEN	2	2 bytes	0x0002
Message parameters			
operation	1	request type	**0x0C**: GAPC_SECURITY_REQ (See Table 372)
auth	1	authentication level	**0x0D**: GAP_AUTH_REQ_SECURE_CONNECTION (**0x08**) | GAP_AUTH_REQ_MITM_BOND (**0x05**) (See Table 376)

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 18. GAPC_SECURITY_CMD example

Message stream (connection sequence – step 3)		
Symbolic message structure reference can be found in Table 17.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1A 0x0E	GAPC_SECURITY_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0x00	2
operation	0x0C	GAPC_SECURITY_REQ
auth	0x0D	**0x08**: GAP_AUTH_REQ_SECURE_CONNECTION **0x05**: GAP_AUTH_REQ_MITM_BOND
Message string		
0x05 0x1A 0x0E 0x0E 0x00 0x10 0x00 0x02 0x00 0x0C 0x0D		

4. Step 4: Reception of completion event message for undirected advertising.

   When the connection is established, the undirected advertising stops, and the Bluetooth LE application informs the GTL Host with the completion event message (GAPM_CMP_EVT) of the GAPM_ADV_UNDIRECT operation.

Table 19. GAPM_CMP_EVT - GAPM_ADV_UNDIRECT – example

Message stream - GAPM_CMP_EVT of GAPM_ADV_UNDIRECT operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x0D	GAPM_ADV_UNDIRECT
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x0D 0x00		

5. Step 5: Reception of completion event message for security request.

The Bluetooth LE application informs the GTL Host with the completion event message (GAPC_CMP_EVT) of the GAPC_SECURITY_REQ operation.

Table 20. GAPC_CMP_EVT - GAPC_SECURITY_REQ – example

Message stream - GAPC_CMP_EVT of GAPC_SECURITY_REQ operation		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x0C	GAPC_SECURITY_REQ
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x0C 0x00		

5.3.2.4 Optional Actions on Connection

5.3.2.4.1 Send Start Advertise Command to Start non-Connectable Advertising

The GTL Host sends GAPM_START_ADVERTISE_CMD to DA14585/531 to start non-connectable advertising.

Table 21. GAPM_START_ADVERTISE_CMD - GAPM_ADV_NON_CONN – example

Message stream – GAPM_START_ADVERTISE_CMD with operation GAPM_ADV_NON_CONN			
Symbolic message structure reference can be found in Table 10.			
Hex data			
Data token	**Hex data (LSB)**	**Comments**	
Initiator	0x05	GTL Initiator	
MSG_ID	0x0D 0x0D	GAPM_START_ADVERTISE_CMD	
DST_ID	0x0D 0x00	TASK_ID_GAPM	
SRC_ID	0x10 0x00	TASK_ID_GTL	
PAR_LEN	0x52 0x00	82 bytes	
op.code	0x0C	GAPM_ADV_NON_CONN	
op.addr_src	0x00	GAPM_STATIC_ADDR	
op.state	0x00 0x00	0x00	
intv_min	0xC8 0x00	0.625 × 200 = 125 ms	
intv_max	0xC8 0x00	0.625 × 200 = 125 ms	
channel_map	0x07	Advertise using all channels	
info.host.mode	0x01	GAP_GEN_DISCOVERABLE	
info.host.adv_filt_policy	0x00	ADV_ALLOW_SCAN_ANY_CON_ANY	
info.host.adv_data_len	0x1B	27 bytes	
info.host.adv_data	0x07 0x03 0x03 0x18 0x02 0x18 0x04 0x18 0x12 0x09 0x44 0x69 0x61 0X6C	0x07	Length: 7 bytes
		0x03	AD Type: Complete list of 16-bit Service Class UUIDs

Message stream – GAPM_START_ADVERTISE_CMD with operation GAPM_ADV_NON_CONN			
	0x6F 0x67 0x50 0x45 0x52 0x20 0x44 0x41 0x31 0x34 0x35 0x38 0x35 0x00 0x00 0x00 0x00	0x03 0x18	UUID: Link Loss Alert
		0x02 0x18	UUID: Immediate Alert
		0x04 0x18	UUID: TX Power
		0x12	Length:18 byte
		0x09	AD Type: Complete local name
		0x44	D
		0x69	i
		0x61	a
		0x6C	l
		0x6F	o
		0x67	g
		0x50	P
		0x45	E
		0x52	R
		0x20	
		0x44	D
		0x41	A
		0x31	1
		0x34	4
		0x35	5
		0x38	8
		0x35	5
		0x00 0x00 0x00 0x00	Padding up-to the 31st byte (can be any value)
info.host.scan_rsp_data_len	0x00	Not used in non-connectable advertising	
info.host.scan_rsp_data	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		
info.host.peer_info.addr	0x00 0x00 0x00 0x00 0x00 0x00	0x000000000000	
info.host.peer_info.addr_type	0x00	Public address	
Message string			
0x05 0x0D 0x0D 0x0D 0x00 0x10 0x00 0x52 0x00 0x0C 0x00 0x00 0x00 0xC8 0x00 0xC8 0x00 0x07 0x01 0x00 0x1B 0x07 0x03 0x03 0x18 0x02 0x18 0x04 0x18 0x12 0x09 0x44 0x69 0x61 0X6C 0x6F 0x67 0x50 0x45 0x52 0x20 0x44 0x41 0x31 0x34 0x35 0x38 0x35 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			

5.3.2.5 Connection Sequence – Overall Example

Figure 8 shows an example of a connection sequence.

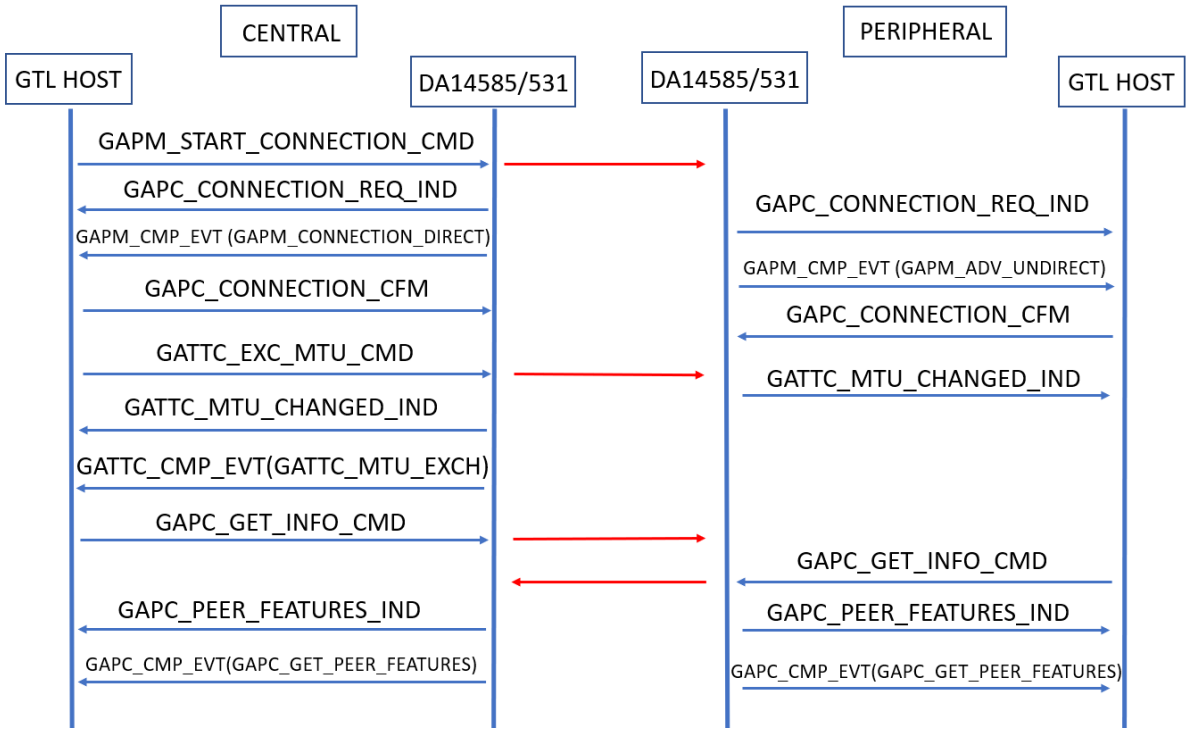


Figure 8. Connection sequence example

1. Step 1: Send a Connection Command.

   As shown in Figure 8, the Central sends **GAPM_START_CONNECTION_CMD** to the Bluetooth LE application. The Bluetooth LE application responds with **GAPC_CONNECTION_REQ_IND.**

   A **GAPC_CONNECTION_REQ_IND** also arrives on the peripheral side (as shown in Figure 6 and detailed in Section 5.3.2.2).

   The **GAPM_START_CONNECTION_CMD** in this example has its operation parameter set to **GAPM_CONNECTION_DIRECT.**

Table 22. GAPM_START_CONNECTION_CMD with operation GAPM_CONNECTION_DIRECT – example #1

Message stream - GAPM_START_CONNECTION_CMD with operation GAPM_CONNECTION_DIRECT		
Symbolic message structure reference can be found in Table 12.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x11 0x0D	GAPM_START_CONNECTION_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x1D 0x00	29 bytes
op.code	0x13	GAPM_CONNECTION_DIRECT
op.addr_src	0x00	GAPM_STATIC_ADDR
op.state	0x00 0x00	Dummy data
scan_interval	0x80 0x01	0x0180 = 384 (384 x 0.625 = 240 ms)
scan_window	0x60 0x01	0x0160 = 352 (352 x 0.625 = 220 ms)
con_intv_min	0x24 0x00	0x24 = 36 (36 × 1.25 ms = 45 ms)
con_intv_max	0x24 0x00	0x24 = 36 (36 × 1.25 ms = 45 ms)
con_latency	0x00 0x00	No Latency in this example
superv_to	0xF4 0x01	0x01F4 = 500 (500 x 10 = 5000 ms = 5 s)
ce_len_min	0x43 0x00	0x43 = 67 (67 x 0.625 = 41.875 ms)
ce_len_max	0x43 0x00	0x43 = 67 (67 x 0.625 = 41.875 ms)
nb_peers	0x01	GAPM_CONNECTION_DIRECT
peers.addr.addr	0x09 0xEE 0x70 0xCA 0xEA 0x80	0x80EACA70EE09
addr_type	0x00	Address type: Public
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x11 0x0D 0x0D 0x00 0x10 0x00 0x1D 0x00 0x13 0x00 0x00 0x00 0x80 0x01 0x60 0x01 0x24 0x00 0x24 0x00 0x00 0x00 0xF4 0x01 0x43 0x00 0x43 0x00 0x01 0x09 0xEE 0x70 0xCA 0xEA 0x80 0x00 0x00		

Table 23. GAPC_CONNECTION_REQ_IND example #2

Message stream: GAPC_CONNECTION_REQ_IND on the central peer's side		
Symbolic message structure reference can be found in Table 13.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x0E	GAPC_CONNECTION_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x10 0x00	16 bytes
conhdl	0x00 0x00	Handle 0
con_interval	0x24 0x00	0x24 = 36 (36 × 1.25 ms = 45 ms)
latency	0x00 0x00	No latency
sup_to	0xF4 0x01	0x01F4 = 500 (500 x 10 = 5000ms = 5s)
clk_accuracy	0x00	Clock accuracy of the peer device
peer_addr_type	0x00	0: PUBLIC
peer_addr	0x09 0xEE 0x70 0xCA 0xEA 0x80	Peer BD address: 0x80EACA70EE09
Message string		
0x05 0x01 0x0E 0x10 0x00 0x0E 0x00 0x10 0x00 0x00 0x00 0x24 0x00 0x00 0x00 0xF4 0x01 0x00 0x00 0x09 0xEE 0x70 0xCA 0xEA 0x80		

Table 24. GAPC_CONNECTION_REQ_IND example #3

Message stream GAPC_CONNECTION_REQ_IND on the peripheral peer's side		
Symbolic message structure reference can be found in Table 13.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x0E	GAPC_CONNECTION_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x10 0x00	16 bytes
conhdl	0x00 0x00	handle 0
con_interval	0x24 0x00	0x24 = 36 (36 × 1.25 ms = 45 ms)
latency	0x00 0x00	No latency
sup_to	0xF4 0x01	0x01F4 = 500 (500 x 10 = 5000ms = 5s)
clk_accuracy	0x00	Clock accuracy of the peer device
peer_addr_type	0x00	0: PUBLIC
peer_addr	0x02 0xEE 0x70 0xCA 0xEA 0x80	Peer BD address: 0x80EACA70EE02
Message string		
0x05 0x01 0x0E 0x10 0x00 0x0E 0x00 0x10 0x00 0x00 0x00 0x24 0x00 0x00 0x00 0xF4 0x01 0x00 0x00 0x02 0xEE 0x70 0xCA 0xEA 0x80		

With the reception of the **GAPC_CONNECTION_REQ_INDs**, the connection procedure begins and the advertising of the peripheral stops. Thus, **GAPM_CMP_EVT** with an operation of **GAPM_CONNECTION_DIRECT** (see Table 25) is sent from the Bluetooth LE application to the GTL Host on the

central's side and **GAPM_CMP_EVT** with an operation of **GAPM_ADV_UNDIRECT** (see Table 19) is sent from the Bluetooth LE application to the GTL Host on the peripheral's side.

Table 25. GAPM_CMP_EVT (GAPM_CONNECTION_DIRECT) – example

Message stream - GAPM_CMP_EVT of GAPM_CONNECTION_DIRECT operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x13	GAPM_CONNECTION_DIRECT
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x13 0x00		

2. Step 2: Send a Connection Confirmation Message.

   Subsequently, both the central and the peripheral send **GAPC_CONNECTION_CFM** to each other. In this example, both **GAPC_CONNECTION_CFM** messages are coincidentally identical to **GAPC_CONNECTION_CFM** of the previous example (Section 5.3.2.3), so you can see Table 16 for the example message stream.

3. Step 3: Initiate an MTU Exchange.

   After the exchange of the **GAPC_CONNECTION_CFM** messages, the central initiates an MTU Exchange by issuing **GATTC_EXC_MTU_CMD**. See Section 5.3.6.1 for **GATTC_EXC_MTU_CMD** symbolic message structure as well as an example. In Section 5.3.6.2 the symbolic message structure and an example of **GATTC_MTU_CHANGED_IND** (that is sent to both the peripheral's and the central's GTL Host by their respective Bluetooth LE Applications) is also outlined. An example of GAPM_CMP_EVT with an operation of GATTC_MTU_EXCH can be found in Table 56.

4. Step 4: Get Peer info.

   Following the MTU exchange, **GAPC_GET_INFO_CMD** is sent to the Bluetooth LE Application on the central's side. The peripheral also sends **GAPC_GET_INFO_CMD** to the Bluetooth LE APPLICATION on the peripheral's side.

   Both devices send the same **GAPC_GET_INFO** command, so the example shown in Table 26 holds for both devices.

Table 26. GAPC_GET_INFO_CMD with GAPC_GET_PEER_FEATURES operation example

Message stream - GAPC_GET_INFO_CMD with GAPC_GET_PEER_FEATURES operation		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x04	GAPC_GET_PEER_FEATURES
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x04		

Following **GAPC_GET_INFO_CMD** issued by both devices, **GAPC_PEER_FEATURES_IND** is received by both the central and the peripheral.

Table 27. GAPC_PEER_FEATURES_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_PEER_FEATURES_IND	0x0E08
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	8 bytes	0x08
Message parameters			
features [8]	8	8-byte array for LE features	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Both devices send the same **GAPC_GET_INFO** command, so the example shown in Table 26 holds for both devices.

Table 28. GAPC_PEER_FEATURES_IND example

Message stream - GAPC_PEER_FEATURES_IND on the central's side		
Symbolic message structure reference can be found in Table 27.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x08 0x0E	GAPC_PEER_FEATURES_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x08 0x00	8 bytes
features [8]	0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00	
Message string		
0x05 0x08 0x0E 0x10 0x00 0x0E 0x00 0x08 0x00 0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

In this example, both devices have the same features. The example shown in Table 28 is valid for both devices.

5. Step 5: Finally, **GAPC_CMP_EVT** with an operation of **GAPC_GET_PEER_FEATURES** is received on both sides to mark the completion of **GAPC_GET_INFO_CMD**.

Table 29. GAPC_CMP_EVT with GAPC_GET_PEER_FEATURES operation example

Message stream - GAPM_CMP_EVT of GAPM_CONNECTION_DIRECT operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x04	GAPC_GET_PEER_FEATURES
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x04 0x00		

5.3.3 Pairing Procedure

The initiator of the pairing procedure is always the GAP central role device. Peripheral devices can request for pairing by sending security request message and waiting for the initiator to initiate the pairing.

The example in this section describes the sequence of a successful pairing procedure for a peripheral device, supporting passkey method with display I/O capabilities. Finally, the peripheral distributes the LTK. The exact sequence of messages for a pairing procedure depends on the security settings of the peer devices.

The pairing sequence for the specific security settings is shown in Figure 9 and Figure 10.

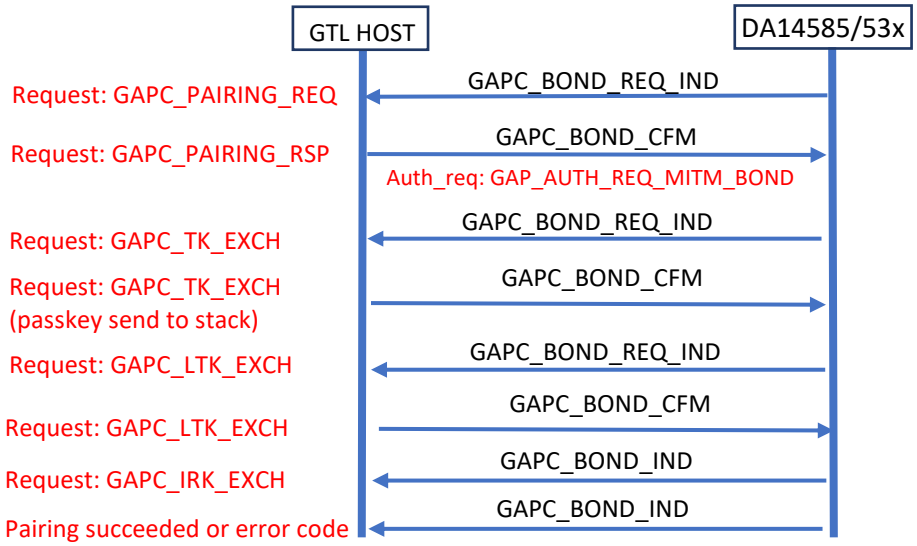


Figure 9. Pairing procedure (legacy pairing)

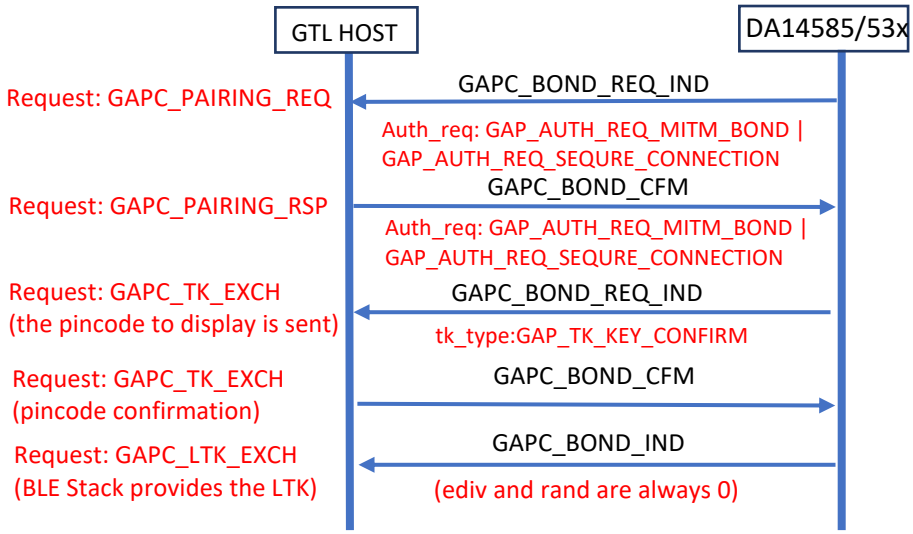


Figure 10. Pairing procedure (secure connections pairing)

IMPORTANT NOTE
For Static Random Address, GTL commands remain supported on the DA14535. However, if a Private Address is utilized with the Identity Resolving Key (IRK), the DA14535 cannot provide support due to limitations when establishing a connection to a private address. This functionality is not enabled on DA14535. See: https://lpccs-docs.renesas.com/UMB119_DA14585DA14531_SW_Platform_Reference/Introduction/Introduction.html

5.3.3.1 Pairing Procedure Step 1: Reception of a Pairing Request

On the pairing responder side (GAP peripheral role), the pairing sequence is always initiated by the reception of the GAPC_BOND_REQ_IND message with the GAPC_PAIRING_REQ value in the request field. This is an indication that the initiator has started the pairing procedure.

The GTL Host should respond to this message by sending GAPC_BOND_CFM with the GAPC_PAIRING_RSP value in the request field.

Table 30. GAPC_BOND_REQ_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_BOND_REQ_IND	0x0E13
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
PAR_LEN	2	18 bytes	0x0012
Message Parameters			
request	1	See Table 377	**0x00**: GAPC_PAIRING_REQ **0x04**: GAPC_TK_EXCH **0x06**: GAPC_CSRK_EXCH **0x07**: GAPC_LTK_EXCH
Request = GAPC_PAIRING_REQ (0x00)			
data.auth_req	1	Authentication level requested by initiator	**0x00**: GAP_AUTH_REQ_NO_MITM_NO_BOND, No MITM, No Bonding. **0x01**: GAP_AUTH_REQ_NO_MITM_BOND, No MITM, Bonding. **0x04**: GAP_AUTH_REQ_MITM_NO_BOND, MITM, No Bonding. **0x05**: GAP_AUTH_REQ_MITM_BOND, MITM and Bonding. The values can be ORed with the following: **0x08**: GAP_AUTH_REQ_SECURE_CONNECTION, supports secure connections. **0x10**: GAP_AUTH_REQ_KEYPRESS_NOTIFICATION, use keypress notifications.
Request = GAPC_LTK_EXCH (0x07)			
data.key_size	1	LTK Key Size	
Request = GAPC_TK_EXCH (0x04)			
data.tk_type	1	Device IO used to get TK	**GAP_TK_OOB**: TK get from out of band method. **GAP_TK_DISPLAY**:TK generated and is displayed by local device. **GAP_TK_KEY_ENTRY**: TK is entered by you using device keyboard. **GAP_TK_KEY_CONFIRM**: TK is displayed and confirmed.
Request = GAPC_CSRK_EXCH (0x06)			
{padding}	1	{padding}	Ignored. Any value is valid
tk	16	Temporary Key	Used for GAPC_TK_EXCH else ignored.

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 31. GAPC_BOND_REQ_IND with a GAPC_PAIRING_REQ example

Message stream (pairing step 1)			
Symbolic message structure reference can be found in Table 30.			
Hex data			
Data token		**Hex data (LSB)**	**Comments**
Initiator		0x05	GTL Initiator
MSG_ID		0x13 0x0E	GAPC_BOND_REQ_IND
DST_ID		0x10 0x00	TASK_ID_GTL
SRC_ID		0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN		0x12 0x00	18 bytes
request		0x00	GAPC_PAIRING_REQ
data.auth_req	**CASE Legacy Pairing (LP):** Central uses Legacy Pairing (Passkey Entry)	0x05	GAP_AUTH_REQ_MITM_BOND
	CASE Secure Connections Pairing (SC): Central supports/uses Secure connections Pairing	0x0D	GAP_AUTH_REQ_MITM_BOND | GAP_AUTH_REQ_SECURE_CONNECTION
tk		0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	No tk used in this example. Ignored
Message string			
CASE LP (Legacy pairing): 0x05 0x13 0x0E 0x10 0x00 0x0E 0x00 0x12 0x00 0x00 **0x05** 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			
CASE SC (Secure connections): 0x05 0x13 0x0E 0x10 0x00 0x0E 0x00 0x12 0x00 0x00 **0x0D** 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			

5.3.3.2 Pairing Procedure Step 2: Send Pairing Response

The GTL Host should respond to the GAPC_BOND_REQ_IND (GAPC_PAIRING_REQ) message by sending GAPC_BOND_CFM with the GAPC_PAIRING_RSP value in the request field of the message. In this message, the GTL Host should pass to DA14585/531 the configuration of the Bluetooth LE security feature.

Table 32. GAPC_BOND_CFM symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_BOND_CFM	0x0E14
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	30 bytes	0x001E
Message parameters			
request	1	See Table 377	**0x01**: GAPC_PAIRING_RSP **0x04**: GAPC_TK_EXCH **0x06**: GAPC_CSRK_EXCH **0x07**: GAPC_LTK_EXCH

Parameters	Bytes	Description	Data
accept	1	**0x01**: accept **0x00**: reject	
Request = GAPC_PAIRING_RSP (0x01)			
data.pairing_feat.iocap	1	Device has Display I/O capabilities	GAP_IO_CAP_DISPLAY_YES_NO (See Table 378)
data.pairing_feat.oob	1	No Out-Of-Band key supported	**0x00**: GAP_OOB_AUTH_DATA_NOT_PRESENT **0x01**: GAP_OOB_AUTH_DATA_PRESENT
data.pairing_feat.auth	1	Passkey and bonding are requested	GAP_AUTH_REQ_MITM_BOND (See Table 376)
data.pairing_feat.key_size	1	16 bytes	0x10
data.pairing_feat.ikey_dist	1	Initiator should distribute IRK for random address resolution	GAP_KDIST_IDKEY (See Table 379)
data.pairing_feat.rkey_dist	1	Responder distributes LTK	GAP_KDIST_ENCKEY (See Table 379)
data.pairing_feat.sec_req	1	Minimum security requirements: MITM is required. Peer cannot pair without MITM.	GAP_SEC1_AUTH_PAIR_ENC (See Table 380)
{padding}	21	{padding} Ignored.	Any value is valid.
Request = GAPC_LTK_EXCH (0x07)			
data.ltk.ltk.key	16	LTK (Long-term Key)	Randomly generated.
data.ltk.ediv	2	Encryption Diversifier	Randomly generated.
data.ltk.randnb.nb	8	Random Number	Randomly generated.
data.ltk.key_size	1	LTK size (7 to 16)	Must be equal to the key_size received in GAPC_BOND_REQ_IND (GAPC_LTK_EXCH) message (Section 5.3.3.5).
{padding}	1	{padding} Ignored.	Any value is valid.
Request = GAPC_CSRK_EXCH (0x06)			
data.csrk.key	16	CSRK	Connection Signature Resolving Key.
{padding}	12	{padding}	Ignored. Any value is valid.
Request = GAPC_TK_EXCH (0x04)			
data.tk.key	16	**CASE LP** (Legacy Pairing): Passkey: The passkey must be entered MSB to LSB order. The actual size of the key is the size of a six-digit number.	
		CASE SC (Secure Connections):	{padding} Ignored. Any value is valid.
{padding}	12	{padding}	Ignored. Any value is valid.

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 33. GAPC_BOND_CFM with GAPC_PAIRING_RSP example

Message stream (pairing step 2)		
Symbolic message structure reference can be found in Table 32.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x14 0x0E	GAPC_BOND_CFM
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x1E 0x00	30 bytes
request	0x01	GAPC_PAIRING_RSP
accept	0x01	0x01: accept
data.pairing_feat. iocap	0x01	GAP_IO_CAP_DISPLAY_YES_NO
data.pairing_feat.oob	0x00	GAP_OOB_AUTH_DATA_NOT_PRESENT
data.pairing_feat.auth	0x0D	GAP_AUTH_REQ_SECURE_CONNECTION | GAP_AUTH_REQ_MITM_BOND
data.pairing_feat.key_size	0x10	16 bytes
data.pairing_feat.ikey_dist	0x02	GAP_KDIST_IDKEY (Initiator should distribute IRK for random address resolution)
data.pairing_feat.rkey_dist	0x01	GAP_KDIST_ENCKEY (Responder distributes LTK)
data.pairing_feat.sec_req	0x02	GAP_SEC1_AUTH_PAIR_ENC (Minimum security requirements. MITM is required. Peer cannot pair without MITM).
tk	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	No TK used in this example so ignored
Message string		
0x05 0x14 0x0E 0x0E 0x00 0x10 0x00 0x1E 0x00 0x01 0x01 0x01 0x00 0x0D 0x10 0x02 0x01 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

5.3.3.3 Pairing Procedure Step 3: Request for a Supplied Passkey (LP)/Pincode (SC)

The MITM method is used in the specific pairing sequence example, so the GAPC_PAIRING_RSP from the GTL Host is followed by a GAPC_BOND_REQ_IND with GAPC_TK_EXCH value in the request field.

- **CASE LP (Legacy Pairing)**: the GTL Host informs that it sends the TK to the Bluetooth LE application in the following step
- **CASE SC (Secure Connections)**: the Bluetooth LE application sends the Pincode that the GTL Host must display

Table 34. GAPC_BOND_REQ_IND for GAPC_TK_EXCH example

Message stream (pairing step 3)			
Symbolic message structure reference can be found in Table 30.			
Hex data			
Data token		**Hex data (LSB)**	**Comments**
Initiator		0x05	GTL Initiator
MSG_ID		0x13 0x0E	GAPC_BOND_REQ_IND
DST_ID		0x10 0x00	TASK_ID_GTL
SRC_ID		0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN		0x12 0x00	18 bytes
request		0x04	GAPC_TK_EXCH
data.tk_type	**CASE LP** (Legacy Pairing)	0x01	Temporary Key Type: GAP_TK_DISPLAY (TK) is generated and displayed on screen. GTL Host sends the TK to the Bluetooth LE application in the following step.
	CASE SC (Secure Connections)	0x03	GAP_TK_KEY_CONFIRM Bluetooth LE application sends the Pincode which GTL Host must display in this message (see tk parameter in this table), so that users can confirm.
tk	**CASE LP** (Legacy Pairing)	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding}
	CASE SC (Secure Connections)	0x58 0xFD 0x88 0x18 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	Pincode: 0x00000000000000000000000001888FD58 So, the peripheral device must present to users the last six digits of the decimal representation of 0x1888FD58, that is 630936.
Message string			
CASE LP (Legacy pairing): 0x05 0x13 0x0E 0x10 0x00 0x0E 0x00 0x12 0x00 0x04 **0x01** 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			
CASE SC (Secure connections): 0x05 0x13 0x0E 0x10 0x00 0x0E 0x00 0x12 0x00 0x04 **0x03** 0x58 0xFD 0x88 0x18 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			

5.3.3.4 Pairing Procedure Step 4: Passkey Entry (LP)/Pincode Confirmation (SC)

Upon reception of the GAPC_BOND_REQ_IND (GAPC_TK_EXCH) message, the GTL Host responds with GAPC_BOND_CFM having GAPC_TK_EXCH in the request field.

- CASE LP (Legacy Pairing): The "accepted" and "passkey" parameters are filled in.
- CASE SC (Secure Connections): Only the "accept" parameter is filled in.

Table 35. GAPC_BOND_CFM for GAPC_TK_EXCH example

Message stream (pairing step 4)			
Symbolic message structure reference can be found in Table 32.			
Hex data			
Data token		**Hex data (LSB)**	**Comments**
Initiator		0x05	GTL Initiator
MSG_ID		0x14 0x0E	GAPC_BOND_CFM
DST_ID		0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID		0x10 0x00	TASK_ID_GTL
PAR_LEN		0x1E 0x00	30 bytes
request		0x04	GAPC_TK_EXCH
accept		0x01	accepted
data.tk.key	**CASE LP** (Legacy Pairing)	0x40 0xE2 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	Passkey (The passkey is in MSB to LSB order. The actual size of the key is the size of a six-digit number. Here: 0x40 0xE2 0x01 means 0x1E240 = 123456 (dec)
	CASE SC (Secure Connections)	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x000x00	{padding}
{padding}		0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
Message string			
CASE LP (Legacy pairing): 0x05 0x14 0x0E 0x0E 0x00 0x10 0x00 0x1E 0x00 0x04 0x01 0x40 0xE2 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			
CASE SC (Secure connections): 0x05 0x14 0x0E 0x0E 0x00 0x10 0x00 0x1E 0x00 0x04 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			

5.3.3.5 Pairing Procedure Step 5: Reception of Request for Long-term Key (LP/SC)

- **CASE LP (Legacy Pairing):**

  The DA14585/531 responder is the distributor of the Long-Term Key (LTK) requested in the pairing response message GAPC_BOND_CFM (GAPC_PAIRING_RSP) (Section 5.3.3.2). When the TK entry is completed successfully, the DA14585/531 GAPC task sends GAPC_BOND_REQ_IND with the GAPC_LTK_EXCH value in the request field to the GTL Host to request the LTK.

Table 36. GAPC_BOND_REQ_IND for GAPC_LTK_EXCH example

Message stream (pairing step 5 - legacy pairing)		
Symbolic message structure reference can be found in Table 30.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x13 0x0E	GAPC_BOND_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x12 0x00	18 bytes
request	0x07	GAPC_LTK_EXCH (Exchange LTK request)
data.key_size	0x10	16 bytes
tk	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	Not used in this example. Ignored
Message string		
0x05 0x13 0x0E 0x10 0x00 0x0E 0x00 0x12 0x00 0x07 0x10 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

- CASE SC (Secure Connections):

  The LTK is calculated by the Bluetooth LE stack, and the message GAPC_BOND_IND with the GAPC_LTK_EXCH value in the request field is sent to provide the LTK to the GTL Host.

Table 37. GAPC_BOND_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_BOND_IND	0x0E15
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
PAR_LEN	2	30 bytes	0x001E
Message parameters			
info	1	See Table 377	**0x02**: GAPC_PAIRING_SUCCEED **0x03**: GAPC_PAIRING_FAILED **0x05**: GAPC_IRK_EXCH **0x06**: GAPC_CSRK_EXCH **0x07**: GAPC_LTK_EXCH
{padding}	1	{padding}	Ignored. Any value is valid.
info = GAPC_PAIRING_SUCCEED (0x02)			
data.auth	1	Authentication level	See Table 376
{padding}	27	{padding}	{padding} Ignored. Any value is valid.
info = GAPC_PAIRING_FAILED (0x03)			
data.reason	1	Pairing failed reason	
{padding}	27	{padding}	{padding} Ignored. Any value is valid.
info = GAPC_LTK_EXCH (0x07)			
data.ltk.ltk.key	16	Long-term Key (LTK)	
data.ltk.ediv	2	ediv	0x0000 (ediv is always zero for Secure Connection Pairing)
data.ltk.randnb.nb	8	rand	0x0000000000000000 (rand is always zero for Secure Connection Pairing)
data.ltk.key_size	1	LTK key size	
{padding}	1	{padding}	Ignored. Any value is valid.
info = GAPC_IRK_EXCH (0x05)			
data.irk.irk.key	16	Identity Resolving Key	
data.irk.addr.addr	6	BD address of the device	
data.irk.addr.addr_type	1	Device address type	**0x00**: public **0x01**: random
{padding}	5	{padding}	Ignored. Any value is valid.
info = GAPC_CSRK_EXCH (0x06)			
data.csrk.key	16	Connection Signature Resolving Key information	
{padding}	12	{padding}	Ignored. Any value is valid.

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 38. GAPC_BOND_IND for GAPC_LTK_EXCH example

Message stream (pairing step 5 – secure connections)		
Symbolic message structure reference can be found in Table 37.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x15 0x0E	GAPC_BOND_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x1E 0x00	30 bytes
info	0x07	GAPC_LTK_EXCH
{padding}	0x00	{padding} Ignored. Any value is valid.
gapc_bond_data.ltk.ltk	0x9C 0xD3 0x32 0x93 0xFE 0x9B 0x3F 0x5A 0x9F 0x7A 0x73 0x91 0xD5 0x61 0x3A 0x4A	LTK: 0x4A3A61D591737A9F5A3F9BFE9332D39C
gapc_bond_data.ediv	0x00 0x00	ediv is always zero for Secure Connection Pairing
gapc_bond_data.ltk.rand	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	rand is always zero for Secure Connection Pairing
gapc_bond_data.ltk.key_size	0x10	16 bytes
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x15 0x0E 0x10 0x00 0x0E 0x00 0x1E 0x00 0x07 0x00 0x9C 0xD3 0x32 0x93 0xFE 0x9B 0x3F 0x5A 0x9F 0x7A 0x73 0x91 0xD5 0x61 0x3A 0x4A 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x10 0x00		

5.3.3.6 Pairing Procedure Step 6: Confirmation of Request for Long-Term Key (LP)/Reception of Encryption Request (SC)

- **CASE LP (Legacy Pairing):**

  The GTL Host should respond to the request for Long-Term Key by sending GAPC_BOND_CFM with the GAPC_LTK_EXCH value in the request field.

Table 39. GAPC_BOND_CFM for GAPC_LTK_EXCH example

Message stream (pairing step 6 – legacy pairing only)		
Symbolic message structure reference can be found in Table 32.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x14 0x0E	GAPC_BOND_CFM
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x1E 0x00	30 bytes
request	0x07	GAPC_LTK_EXCH (Exchange LTK)
accept	0x01	Accepted
data.ltk.ltk.key	0x91 0xC1 0x67 0x1D 0xD9 0xB0 0x4D 0x29 0xB8 0xA9 0xAF 0x43 0x74 0x6B 0xD1 0x53	Long-term key (Randomly generated)
data.ltk.ediv	0x4E 0x1B	Encryption Diversifier (Randomly generated)
data.ltk.randnb.nb	0xB7 0x57 0x83 0x2F 0x07 0x33 0x30 0x0E	Random Number (Randomly generated)
data.ltk.key_size	0x10	Long-term key size: 16 bytes
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x14 0x0E 0x0E 0x00 0x10 0x00 0x1E 0x00 0x07 0x01 0x91 0xC1 0x67 0x1D 0xD9 0xB0 0x4D 0x29 0xB8 0xA9 0xAF 0x43 0x74 0x6B 0xD1 0x53 0x4E 0x1B 0xB7 0x57 0x83 0x2F 0x07 0x33 0x30 0x0E 0x10 0x00		

The message transmission to the GAPC task distributes the encrypted Long-Term Key to the peer device.

- **CASE SC (Secure connections):**

  At this point, the GAPC_ENCRYPT_REQ_IND (Table 45) message may be received. This can be responded with GAPC_ENCRYPT_CFM (Table 47) using the LTK received in Section 5.3.3.5. GAPC_ENCRYPT_IND (Table 49) is then received.

5.3.3.7 Pairing Procedure Step 7: Exchange of Identity Resolving Key (IRK)

The DA14585/531 responder is the distributor of the Identity Resolving Key requested in the pairing response message, GAPC_BOND_CFM (GAPC_PAIRING_RSP) (Section 5.3.3.2). When the LTK entry is completed successfully, the DA14585/531 GAPC task sends GAPC_BOND_REQ_IND with the GAPC_IRK_EXCH value in the request field.

Table 40. GAPC_BOND_IND for GAPC_IRK_EXCH example

Message stream (pairing step 7)		
Symbolic message structure reference can be found in Table 37.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x15 0x0E	GAPC_BOND_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x1E 0x00	30 bytes
info	0x05	GAPC_IRK_EXCH
{padding}	0x00	{padding} Ignored. Any value is valid.
data.irk.irk.key	0x87 0x2F 0xF3 0xAC 0x0D 0x04 0x28 0xEB 0x37 0xB5 0xB6 0xCC 0x9E 0x5A 0xE8 0x67	Identity Resolving Key: 0x67E85A9ECCB6B537EB28040DACF32F87
data.irk.addr.addr	0xC2 0x30 0xA2 0x0B 0x22 0xAC	BD address of the device: 0xAC220BA230C2
data.irk.addr.addr_type	0x00	Device address type: public
{padding}	0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x15 0x0E 0x10 0x00 0x0E 0x00 0x1E 0x00 0x05 0x00 0x87 0x2F 0xF3 0xAC 0x0D 0x04 0x28 0xEB 0x37 0xB5 0xB6 0xCC 0x9E 0x5A 0xE8 0x67 0xC2 0x30 0xA2 0x0B 0x22 0xAC 0x00 0x00 0x00 0x00 0x00 0x00		

5.3.3.8 Pairing Procedure Step 8: Completion of Pairing Procedure

The reception of the GAPC_BOND_IND message signals the completion of the pairing procedure. The result of the procedure (success/failure) is determined by the value of the info field.

The GTL Host should store information of the device. The BD address, BD address type, LTK, ediv, rand, and LTK size must be stored and reused during the encryption establishment procedure upon every subsequent reconnection with the specific peer device.

Table 41. GAPC_BOND_IND with GAPC_PAIRING_SUCCEED example

Message stream (pairing step 8)			
Symbolic message structure reference can be found in Table 37.			
Hex data			
Data token		**Hex data (LSB)**	**Comments**
Initiator		0x05	GTL Initiator
MSG_ID		0x15 0x0E	GAPC_BOND_IND
DST_ID		0x10 0x00	TASK_ID_GTL
SRC_ID		0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN		0x1E 0x00	30 bytes
Info		0x02	GAPC_PAIRING_SUCCEED
{padding}		0x00	{padding} Ignored. Any value is valid.
data.auth	**CASE LP** (Legacy Pairing)	0x05	Authentication level: GAP_AUTH_REQ_MITM_BOND
	CASE SC (Secure Connections)	0x0D	Authentication level: GAP_AUTH_REQ_MITM_BOND | GAP_AUTH_REQ_SECURE_CONNECTION
{padding}		0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
Message string			
0x05 0x15 0x0E 0x10 0x00 0x0E 0x00 0x1E 0x00 0x02 0x00 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00			

5.3.3.9 Peer Device Identification: Bonding

When the pairing has completed successfully, the bonding data is stored in the security database. Also, the attempt to identify the peer is initiated at this point. The procedure is shown in Figure 11.

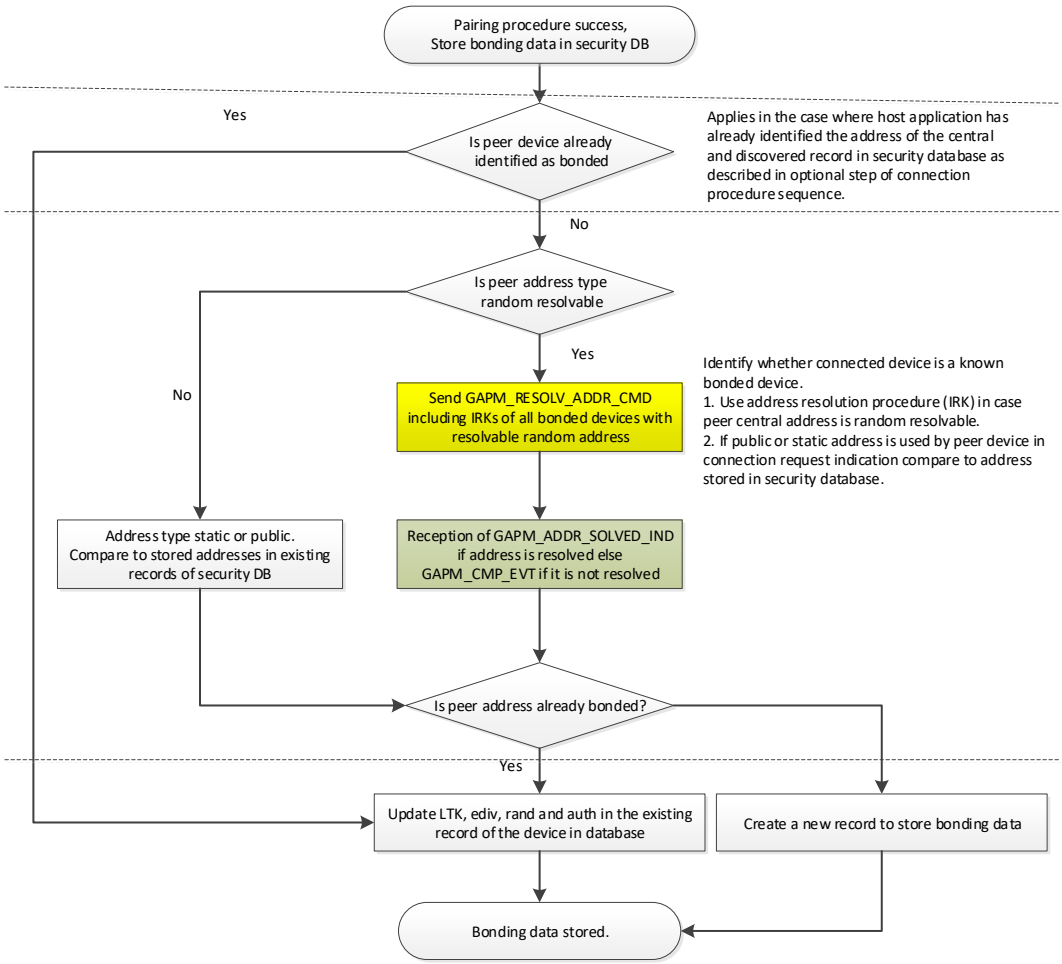


Figure 11. Peer identification and storage of the bonding data

5.3.3.9.1 Random Address Resolution

1. GTL Host requests the Bluetooth LE application for resolution of the central device's Bluetooth LE address.

   If the device is already bonded to at least one host with random address (at least one IRK in the bonded data is available) and the connection request command received in step 1 indicates that the central device uses random address, GTL Host responds with the GAPM_RESOLV_ADDR_CMD command to try to identify the central that requests the connection based on the existing bonding information.

Table 42. GAPM_RESOLV_ADDR_CMD example

Message stream (connection sequence – random address resolution request)		
Symbolic message structure reference can be found in Table 206.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x14 0x0D	GAPM_RESOLV_ADDR_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x18 0x00	24
operation	0x17	GAPM requested operation: GAPM_RESOLV_ADDR
nb_key	0x01	Only one IRK key is available
addr	0xBD 0x74 0xEB 0x75 0x7E 0x59	Central Device random address: 0x6A4903A21B59
IRK	0x87 0x2F 0xF3 0xAC 0x0D 0x04 0x28 0xEB 0x37 0xB5 0xB6 0xCC 0x9E 0x5A 0xE8 0x67	IRK: 0x67E85A9ECCB6B537EB28040DACF32F87
Message string		
0x05 0x14 0x0D 0x0D 0x00 0x10 0x00 0x18 0x00 0x17 0x01 0xBD 0x74 0xEB 0x75 0x7E 0x59 0x87 0x2F 0xF3 0xAC 0x0D 0x04 0x28 0xEB 0x37 0xB5 0xB6 0xCC 0x9E 0x5A 0xE8 0x67		

2. Reception of the GAPM_ADDR_SOLVED_IND message.

   Only when the address of the central is resolved successfully (the central has already bonded with the peripheral in the past, so the GTL Host has the respective IRK available in the bonding data), the GAPM_ADDR_SOLVED_IND message is sent by DA14585/531. Otherwise, this message is not generated by DA14585/531.

Table 43. GAPM_ADDR_SOLVED_IND example

Message stream (connection sequence – random address resolution request)		
Symbolic message structure reference can be found in Table 207.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x15 0x0D	GAPM_ADDR_SOLVED_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x16 0x00	22 bytes
bd_addr	0xBD 0x74 0xEB 0x75 0x7E 0x59	bd_addr: 0x597E75EB74BD
IRK	0x87 0x2F 0xF3 0xAC 0x0D 0x04 0x28 0xEB 0x37 0xB5 0xB6 0xCC 0x9E 0x5A 0xE8 0x67	IRK: 0X67E85A9ECCB6B537EB28040DACF32F87
Message string		
0x05 0x15 0x0D 0x10 0x00 0x0D 0x00 0x16 0x00 0xBD 0x74 0xEB 0x75 0x7E 0x59 0x87 0x2F 0xF3 0xAC 0x0D 0x04 0x28 0xEB 0x37 0xB5 0xB6 0xCC 0x9E 0x5A 0xE8 0x67		

3. Reception of the message GAPM_CMP_EVT.

Table 44. GAPM_CMP_EVT for GAPM_RESOLV_ADDR example

Message stream (connection sequence - random address resolution request)		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2
operation	0x17	GAPM requested operation: GAPM_RESOLV_ADDR
status	If the address has been resolved successfully: 0x00	GAP_ERR_NO_ERROR: no error; the address resolution is successful
	If the address has not been resolved: 0x47	GAP_ERR_NOT_FOUND: search algorithm has finished, but no result is found
Message string		
▪ Address resolved successfully: 0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x17 0x00 ▪ Address not resolved: 0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x17 0x47		

5.3.4 Security Establishment

The following example provides details of the message exchange sequence between the GTL Host and the DA14585/531 GAPC task for the encryption establishment procedure. The procedure is used between two bonded devices to establish a secure link without executing the pairing procedure again.

Like pairing, the encryption procedure is initiated by the central GAP role devices. During the encryption procedure, the devices use the LTK, ediv, and rand distributed during the pairing procedure.

The message exchange sequence is shown in Figure 13.

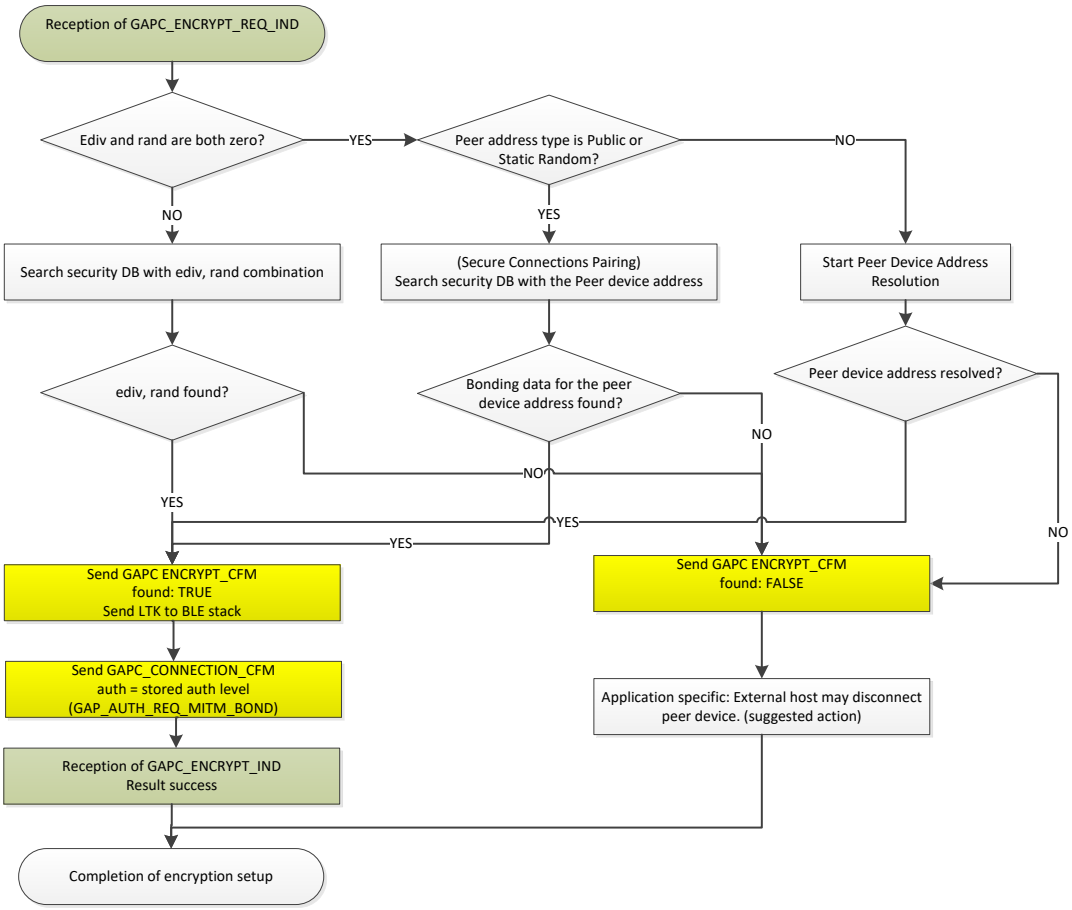


Figure 12. Encryption establishment procedure diagram

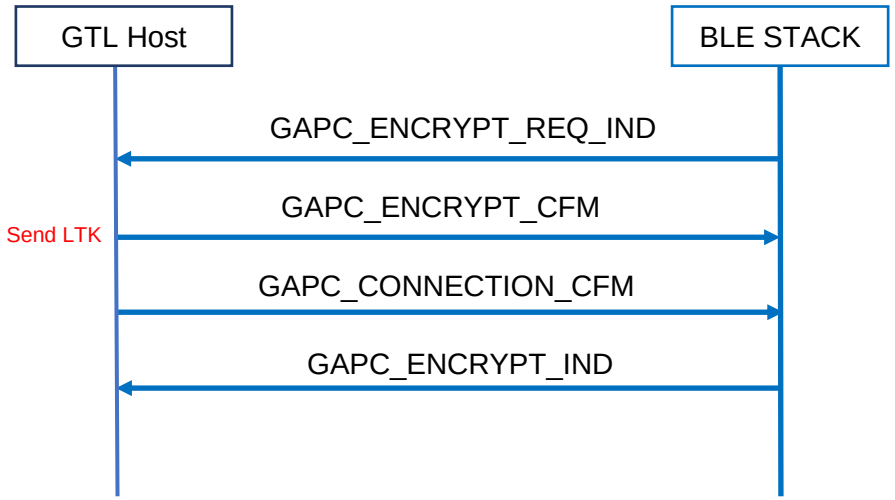


Figure 13. Encryption establishment procedure

5.3.4.1 Security Establishment Step 1: Reception of an Encryption Request

On the GAP peripheral role devices, the pairing sequence is always initiated when DA14585/531 receives a GAPC_ENCRYPT_REQ_IND message by the GTL Host. This message also includes the ediv and rand numbers that the peer device uses to initiate the encryption establishment procedure. For Legacy Pairing, these values can be used as a unique ID to identify the LTK that the GTL Host must use in the encryption procedure. For Secure Connections Pairing, these values are both "all zeros" and the retrieval of bonding information is based on the peer address.

The GTL Host should respond to this message by sending a GAPC_ENCRYPT_CFM with the GAPC_PAIRING_RSP value in the request field.

Table 45. GAPC_ENCRYPT_REQ_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_ENCRYPT_REQ_IND	0x0E17
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
PAR_LEN	2	10 bytes	0x000A
Message parameters			
ediv	2	Encryption Diversifier	For Secure Connection pairing ediv is zero.
rand_nb.nb	8	Random Number	For Secure Connection pairing rand is zero.

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 46. GAPC_ENCRYPT_REQ_IND example

Message stream (encryption step 1)		
Symbolic message structure reference can be found in Table 45.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x17 0x0E	GAPC_ENCRYPT_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x0A 0x00	10 bytes
ediv	0x39 0x62	ediv: 0x6239
rand_nb.nb	0xB0 0xCB 0x49 0x68 0x07 0x45 0xC8 0x7F	rand: 0x7FC845076849CBB0
Message string		
0x05 0x17 0x0E 0x10 0x00 0x0E 0x00 0x0A 0x00 0x39 0x62 0xB0 0xCB 0x49 0x68 0x07 0x45 0xC8 0x7F		

5.3.4.2 Security Establishment Step 2: Send Encryption Response

The GTL Host responds to the GAPC_ENCRYPT_REQ_IND (GAPC_ENCRYPT_REQ) message by sending the GAPC_ENCRYPT_CFM message. In this message the GTL Host sends the Long-Term Key (LTK) and the LTK size that are associated with the currently connected host.

Table 47. GAPC_ENCRYPT_CFM symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_ENCRYPT_CFM	0x0E18
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	18 bytes	0x0012
Message parameters			
found	1	**0x01**: found **0x00**: not found	
ltk.key	16	Long-term Key	
key_size	1	LTK size	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 48. GAPC_ENCRYPT_CFM example

Message stream (encryption – step 2)		
Symbolic message structure reference can be found in Table 47.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x18 0x0E	GAPC_ENCRYPT_CFM
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x12 0x00	18 bytes
found	0x01	Found
ltk.key	0x9C 0xD3 0x32 0x93 0xFE 0x9B 0x3F 0x5A 0x9F 0x7A 0x73 0x91 0xD5 0x61 0x3A 0x4A	Long-term Key (LTK): 0x4A3A61D591737A9F5A3F9BFE9332D39C
key_size	0x10	LTK size: 16 bytes
Message string		
0x05 0x18 0x0E 0x0E 0x00 0x10 0x00 0x12 0x00 0x01 0x9C 0xD3 0x32 0x93 0xFE 0x9B 0x3F 0x5A 0x9F 0x7A 0x73 0x91 0xD5 0x61 0x3A 0x4A 0x10		

5.3.4.3 Security Establishment Step 3: Encryption Completion

The encryption procedure is completed by the reception of the GAPC_ENCRYPT_IND message. The authentication level of the established secure link is included in this message.

Table 49. GAPC_ENCRYPT_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_ENCRYPT_IND	0x0E19
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
PAR_LEN	2	1 byte	0x0001
Message parameters			
auth	1	Authentication level	Any combination of the following GAP_AUTH_NONE (0x00) or {GAP_AUTH_BOND (0x01) and/or GAP_AUTH_MITM (0x04)} (See Table 376)

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 50. GAPC_ENCRYPT_IND example

Message stream (encryption – step 3)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x19 0x0E	GAPC_ENCRYPT_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x01 0x00	1 byte
auth	0x05	Authentication level: GAP_AUTH_BOND (**0x01**) and GAP_AUTH_MITM (**0x04**)
Message string		
0x05 0x19 0x38 0x3F 0x00 0x0E 0x00 0x01 0x00 0x05		

5.3.5 Completion Event Messages

5.3.5.1 GAPM Completion Event Message

The message GAPM_CMP_EVT is sent by GAPM task to the GTL Host to signal the completion of an operation.

Table 51. GAPM_CMP_EVT symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_CMP_EVT	0x0D00
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	2	0x0002
Message parameters			
operation	1	▪ GAPM_RESET: Reset Bluetooth LE subsystem. ▪ GAPM_SET_DEV_CONFIG: Set device configuration. ▪ GAPM_ADD_DEV_IN_WLIST: Add device in whitelist. ▪ (…) See Table 374	▪ 0x01 ▪ 0x03 ▪ 0x09 ▪ (…)
status	1	▪ GAP_ERR_NO_ERROR ▪ GAP_ERR_CANCELED ▪ GAP_ERR_NOT_FOUND ▪ (…) See Table 382	▪ 0x00 ▪ 0x44 ▪ 0x47 ▪ (…)

5.3.5.2 GAPC Completion Event Message

The message GAPC_CMP_EVT is sent by the GAPC task to the GTL Host to signal the completion of an operation.

Table 52. GAPC_CMP_EVT symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05

Parameters	Bytes	Description	Hex data
Header			
MSG_ID	2	GAPC_CMP_EVT	0x0E00
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	2	0x0002
Message parameters			
operation	1	GAPC_SECURITY_REQ	0x0C (See Table 372)
status	1	▪ GAP_ERR_NO_ERROR ▪ GAP_ERR_CANCELED ▪ GAP_ERR_REJECTED ▪ (…)	▪ 0x00 ▪ 0x44 ▪ 0x48 ▪ (…) See Table 382

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

5.3.5.3 GATTC Completion Event Message

The message GATTC_CMP_EVT is sent by the GATTC task to the GTL Host to signal the completion of an operation.

Table 53. GATTC_CMP_EVT symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_CMP_EVT	0x0C00
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC	0xnn0C (nn = conidx) (Note 1)
PAR_LEN	2	4 bytes	0x0004
Message parameters			
operation	1	▪ GATTC_MTU_EXCH ▪ GATTC_NOTIFY ▪ GATTC_INDICATE ▪ (…)	▪ 0x01 ▪ 0x12 ▪ 0x13 ▪ (…) See Table 373
status	1	▪ GAP_ERR_NO_ERROR ▪ (…)	▪ 0x00 ▪ (…) See Table 382
seq_num	2	Operation sequence number	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

5.3.6 MTU Exchange

5.3.6.1 Request to Change the MTU

1. Exchange MTU Request.

   The Exchange MTU Request is used by the client to inform the server of the client's maximum receive MTU size and request the server to respond with its maximum receive MTU size.

Table 54. GATTC_EXC_MTU_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_EXC_MTU_CMD	0x0C01
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	4 bytes	0x0004
Message parameters			
operation	1	GATT request type	GATTC_MTU_EXCH: 0x01
{padding}	1	{padding} Ignored. Any value is valid.	{padding}
seq_num	2	Operation sequence number	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 55. GATTC_EXC_MTU_CMD example

Message stream – exchange MTU request		
Symbolic message structure reference can be found in Table 54.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x0C	GATTC_EXC_MTU_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x04 0x00	4 bytes
operation	0x01	GATTC_MTU_EXCH
{padding}	0x00	{padding} Ignored. Any value is valid.
seq_num	0x34 0x12	Sequence number 0x1234
Message string		
0x05 0x01 0x0C 0x0C 0x00 0x10 0x00 0x04 0x00 0x01 0x00 0x34 0x12		

2. Completion event for the Exchange MTU Request.

   When the response of the Exchange MTU Request from the peer server is received, the GTL Host is notified with GATTC_CMP_EVT for the GATTC_MTU_EXCH operation.

Table 56. GATTC_CMP_EVT example for GATTC_MTU_EXCH operation

Message stream (GATTC_CMP_EVT for GATTC_MTU_EXCH operation)		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x01	GATTC_MTU_EXCH
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x34 0x12	Sequence number 0x1234
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x01 0x00 0x34 0x12		

5.3.6.2 Indication that MTU Has Changed

The DA14585/531 informs the GTL Host that the MTU size has changed by sending the GATTC_MTU_CHANGED_IND API message.

Table 57. GATTC_MTU_CHANGED_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_MTU_CHANGED_IND	0x0C02
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC	0xnn0C (nn = conidx) (Note 1)
PAR_LEN	2	4 bytes	0x0004
Message parameters			
mtu	2	The MTU size	
seq_num	2	Operation sequence number	

Table 58. GATTC_MTU_CHANGED_IND example

Message stream - GATTC_MTU_CHANGED_IND		
Symbolic message structure reference can be found in Table 57.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x02 0x0C	GATTC_MTU_CHANGED_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	0x0004 (4 bytes)
mtu	0x00 0x02	0x0200 (512 bytes)
seq_num	0x00 0x00	Sequence number 0x00000
Message string		
0x05 0x02 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x00 0x02 0x00 0x00		

5.3.7 Connection Parameter Update

The connection parameter update can be requested either by the central or the peripheral.

5.3.7.1 Connection Parameter Update Request Initiated by the Central

1. Step 1: Connection parameter update request.

   The central can request an update of connection parameters. This triggers the Bluetooth LE application to send GAPC_PARAM_UPDATE_REQ_IND to the GTL Host.

Table 59. GAPC_PARAM_UPDATE_REQ_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_PARAM_UPDATE_REQ_IND	0x0E0F
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	8 bytes	0x0008
Message parameters			
intv_min	2	Minimum of connection interval	Expressed in 1.25 ms slots
intv_max	2	Maximum of connection interval	Expressed in 1.25 ms slots
latency	2	Connection latency (number of events)	
time_out	2	Link supervision timeout	Multiply by 10 ms

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 60. GAPC_PARAM_UPDATE_REQ_IND example

Message stream – connection parameters update request initiated by the central		
Symbolic message structure reference can be found in Table 59.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0F 0x0E	GAPC_PARAM_UPDATE_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x08 0x00	8 bytes
intv_min	0x27 0x00	0x27 = 39 (39 × 1.25 ms = 48.75 ms)
intv_max	0x27 0x00	0x27 = 39 (39 × 1.25 ms = 48.75 ms)
latency	0x00 0x00	0x0000: No latency
time_out	0xD0 0x07	0x07D0 = 2000 (2000 x 10 = 20000 ms = 20 s)
Message string		
0x05 0x0F 0x0E 0x10 0x00 0x0E 0x00 0x08 0x00 0x27 0x00 0x27 0x00 0x00 0x00 0xD0 0x07		

2. Step 2: Connection parameter update request confirmation.

   The GTL Host sends the confirmation for the connection parameter update request to the Bluetooth LE application.

Table 61. GAPC_PARAM_UPDATE_CFM symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_PARAM_UPDATE_CFM	0x0E10
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	6 bytes	0x0006
Message parameters			
accept	1	True to accept slave connection parameters, False otherwise.	
{padding}	1	{padding} Ignored. Any value is valid.	{padding}
ce_len_min	2	Minimum CE length N Value Time = N × 0.625 ms (not used by a slave device)	0x0000
ce_len_max	2	Maximum CE length N Value Time = N × 0.625 ms (not used by a slave device)	0xFFFF

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 62. GAPC_PARAM_UPDATE_CFM example

Message stream - connection parameters update request confirmation		
Symbolic message structure reference can be found in Table 61.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x10 0x0E	GAPC_PARAM_UPDATE_CFM
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x06 0x00	6 bytes
Accept	0x01	accept
{padding}	0x00	{padding} Ignored. Any value is valid.
ce_len_min	0x00 0x00	0x0000
ce_len_max	0xFF 0xFF	0xFFFF
Message string		
0x05 0x10 0x0E 0x0E 0x00 0x10 0x00 0x06 0x00 0x01 0x00 0x00 0x00 0xFF 0xFF		

3. Step 3: Connection parameters update indication.

   The Bluetooth LE application informs the GTL Host of the completion of the connection parameter update procedure with **GAPC_PARAM_UPDATED_IND**. This message is detailed in Section 5.3.7.3.

5.3.7.2 Connection Parameters Update Request Initiated by the Peripheral

1. Step 1: GTL Host sends the request for connection parameters update.

   The GTL Host can send the command GAPC_PARAM_UPDATE_CMD to the Bluetooth LE application for a connection parameter update request to be sent to the central.

Table 63. GAPC_PARAM_UPDATE_CMD symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_PARAM_UPDATE_CMD	0x0E0E
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	14 bytes	0x000E
Message parameters			
operation	1	GAPC requested operation	0x09: GAPC_UPDATE_PARAMS (See Table 372)
pkt_id	1	Internal parameter used to manage internally L2CAP packet identifier for signaling	0x00
intv_min	2	Minimum of connection interval	Expressed in 1.25 ms slots
intv_max	2	Maximum of connection interval	Expressed in 1.25 ms slots
latency	2	Connection latency (number of events)	
time_out	2	Link supervision timeout	Multiply by 10

Parameters	Bytes	Description	Data
Header			
ce_len_min	2	Minimum CE length N Value Time = N × 0.625 ms (not used by a slave device)	0xFFFF
ce_len_max	2	Maximum CE length N Value Time = N × 0.625 ms (not used by a slave device)	0xFFFF

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 64. GAPC_PARAM_UPDATE_CMD example

Message stream – connection parameters update request command		
Symbolic message structure reference can be found in Table 63.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0E 0x0E	GAPC_PARAM_UPDATE_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0E 0x00	14 bytes
operation	0x09	GAPC_UPDATE_PARAMS
pkt_id	0x00	0x00
intv_min	0x0A 0x00	0x000A = 10 (10 × 1.25 ms = 12.5 ms)
intv_max	0x0A 0x00	0x000A = 10 (10 × 1.25 ms = 12.5 ms)
latency	0x00 0x00	0x0000: No latency
time_out	0xE8 0x03	0x03E8 = 1000 (1000 x 10 = 10000 ms = 10 s)
ce_len_min	0xFF 0xFF	0xFFFF
ce_len_max	0xFF 0xFF	0xFFFF
Message string		
0x05 0x0E 0x0E 0x0E 0x00 0x10 0x00 0x0E 0x00 0x09 0x00 0x0A 0x00 0x0A 0x00 0x00 0x00 0xE8 0x03 0xFF 0xFF 0xFF 0xFF		

2. Step 2: The Bluetooth LE application sends the connection parameters updated indication.

   The Bluetooth LE application sends the connection parameters updated indication when the central has responded to the connection parameter update request (see Section 5.3.7.3).

3. Step 3: Reception of completion event message for connection parameter update.

   The Bluetooth LE application informs the GTL Host of the completion of the connection parameter update with the completion event message (GAPC_CMP_EVT) of the GAPC_UPDATE_PARAMS operation.

Table 65. GAPC_CMP_EVT for GAPC_UPDATE_PARAMS example

Message stream - GAPC_CMP_EVT of GAPC_UPDATE_PARAMS operation		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x09	GAPC_UPDATE_PARAMS
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x09 0x00		

5.3.7.3 Connection Parameters Updated Indication (Central to Peripheral)

The Bluetooth LE application sends the connection parameters updated indication when the connection parameter update has completed.

Table 66. GAPC_PARAM_UPDATED_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_PARAM_UPDATED_IND	0x0E11
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	6	0x0006
Message parameters			
con_interval	2	Connection interval	Expressed in 1.25 ms slots
con_latency	2	Connection latency	
sup_to	2	Supervision timeout	Multiply by 10 ms

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 67. GAPC_PARAM_UPDATED_IND example

Message stream - connection parameters updated indication		
Symbolic message structure reference can be found in Table 66.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x11 0x0E	GAPC_PARAM_UPDATED_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x06 0x00	6 bytes
con_interval	0x27 0x00	0x0027 = 39 (39 × 1.25 ms = 48.75 ms)
con_latency	0x00 0x00	0x0000 (no latency)
sup_to	0xD0 0x07	0x07D0 = 2000 (2000 x 10 = 20000ms = 20s)
Message string		
0x05 0x11 0x0E 0x10 0x00 0x0E 0x00 0x06 0x00 0x27 0x00 0x00 0x00 0xD0 0x07		

5.3.8 Database Creation and Permissions Management

5.3.8.1 Description of the Example Database

The database of a service can be created by the GTL Host application. In this case, the database resides on the DA14585/531 side and the data of the attributes can either be maintained in the DA14585/531 side or in the GTL Host side. This is configurable on a per-attribute basis.

The database initialization must be done at system start-up before the device's first connection. When the device gets connected to the central, upon reception of any Bluetooth LE event related to this service, DA14585/531 sends the corresponding indication message. A service database can be created by the GTL Host, while the database and code of other services may reside in DA14585/531.

The attributes in the database of the example presented in this section are shown in Table 68.

Table 68. Example service database definition

Attribute type	**Service declaration**	**Description**
UUID	0x2800	Primary Service
Attribute value	0xEDFEC62E99100BAC5241D8BDA6932A2F	128-bit UUID for the Service
Start handle	0x0000	Assign automatically
Task id	0x0010	TASK_ID_GTL
permissions	0xCC	Primary Service, 128-bit UUID, AUTH
Number of attributes	0x07	7 attributes
Characteristic A		
Attribute type	**Characteristic declaration**	**Description**
UUID	0x2803	
permissions	0x00000001	Read Permission
Trigger Read Indication | maximum attribute length	0x8000	Trigger Read Indication: Yes Maximum attribute length: 0
Attribute type	**Characteristic value declaration**	**Description**
UUID	0x2D86686A53DC25B30C4AF0E10C8DEE20	

Attribute type	Service declaration	Description
permissions	0x000A0009	Write Permission, Write Request Accepted, Read Permission, 128-bit UUID
Trigger Read Indication | maximum attribute length	0x8064	Trigger Read Indication: Yes Maximum attribute length: 100 bytes
Attribute type	**Characteristic user description**	**Description**
UUID	0x2901	
permissions (characteristic properties)	0x00000001	Read Permission
Trigger Read Indication | maximum attribute length	0x80FA	Trigger Read Indication: Yes Maximum attribute length: 250 bytes
Characteristic B		
Attribute type	**Characteristic declaration**	**Description**
UUID	0x2803	0x2803: "Characteristic Declaration"
permissions	0x00000001	Read Permission
Trigger Read Indication | maximum attribute length	0x0000	Trigger Read Indication: No Maximum attribute length: 0
Attribute type	**Characteristic value declaration**	**Description**
UUID	0x5A87B4EF3BFA76A8E64292933C31434F	
permissions	0x00088249	128-bit UUID, write command accepted, notification permission enabled, indication permission enabled, write permission, read permission
Trigger Read Indication | maximum attribute length	0x0064	Trigger Read Indication: No Maximum attribute length: 100 bytes
Attribute type	**Characteristic user description**	**Description**
UUID	0x2901	
permissions	0x00000001	Read Permission
Trigger Read Indication | maximum attribute length	0x00FA	Trigger Read Indication: No Maximum attribute length: 250 bytes
Attribute type	**Client characteristic configuration**	**Description**
UUID	0x2902	
permissions	0x00020009	Write Permission, Write Request Accepted, Read Permission, 128-bit UUID
Trigger Read Indication | maximum attribute length	0x00020009	Trigger Read Indication: No Maximum attribute length: 2 bytes

5.3.8.2 Creation of Example Database

1. Step 1: Service creation.

   The GTL Host requests the creation of a service database from the DA14585/531 by sending the GATTM_ADD_SVC_REQ command. The DA14585/531 allocates the required memory space for the service database and returns the start handle of the service.

Table 69. GATTM_ADD_SVC_REQ symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ADD_SVC_REQ	0x0B00
DST_ID	2	TASK_ID_GATTM	0x000B
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Total count of bytes for the parameters	24 × ({nb_atts} + 1)
Message parameters			
start_handle	2	Set start handle of service attribute	Attribute Start Handle (0 = dynamically allocated)
task_id	2	Task identifier that manages service	TASK_ID_GTL(0x0010)
perm	1	Service Permission	Service Permission: ▪ Bit [0]: Task that manage service is multi-instantiated (Connection index is conveyed) ▪ Bit [1]: Encryption key size must be 16 bytes ▪ Bit [2-4]: Service Permission ($00_{(b)}$ = Disable, $01_{(b)}$ = Enable, $10_{(b)}$ = UNAUTH, $11_{(b)}$ = AUTH) ▪ Bit [5-6]: UUID Length ($00_{(b)}$ = 16 bits, $01_{(b)}$ = 32 bits, $10_{(b)}$ = 128 bits, $11_{(b)}$ = RFU) ▪ Bit [7] Primary Service (1 = Primary Service, 0 = Secondary Service)
nb_att	1	Number of attributes in the service	
uuid	16	The UUID of the service	
{padding}	2	{padding}	Can consist of any value
atts	24×nb_att	List of attribute description present in service.	

Table 70. GATTM_ADD_SVC_REQ for example database

Message stream		
Symbolic message structure reference can be found in Table 69.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0B	GATTM_ADD_SVC_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0C 0x00	192 bytes
start_handle	0x00 0x00	0x0000 instructs DA14585/531 to automatically set the handle to the first one available
task_id	0x10 0x00	TASK_ID_GTL
perm	0xCC	Primary Service, 128-bits UUID, AUTH, Encryption key size must be 16 bytes. See Table 71.

Message stream		
nb_att	0x07	The database includes seven attributes (those declared with the atts parameter in this table)
uuid	0x2F 0x2A 0x93 0xA6 0xBD 0xD8 0x41 0x52 0xAC 0x0B 0x10 0x99 0x2E 0xC6 0xFE 0xED	0xEDFEC62E99100BAC5241D8BDA6932A2F
{padding}	0x00 0x00	Can consist of any value
atts		
[Attribute 1] Characteristic A: characteristic declaration		
uuid	0x03 0x28 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	0x2803: "Characteristic Declaration"
perm	0x01 0x00 0x00 0x00	0x00000001: Read Permission
max_len	0x00 0x80	0x8000: Trigger Read Indication: Yes Maximum attribute length: 0
{padding}	0x00 0x00	Can consist of any value
[Attribute 2] Characteristic A: characteristic value declaration		
uuid	0x20 0xEE 0x8D 0x0C 0xE1 0xF0 0x4A 0x0C 0xB3 0x25 0xDC 0x53 0x6A 0x68 0x86 0x2D	0x2D86686A53DC25B30C4AF0E10C8DEE20
perm	0x09 0x00 0x0A 0x00	0x000A0009: Write Permission, Write Request Accepted, Read Permission, 128-bit UUID
max_len	0x64 0x80	0x8064: Trigger Read Indication: Yes Maximum attribute length: 100 bytes
{padding}	0x00 0x00	Can consist of any value
[Attribute 3] Characteristic A: characteristic user description		
uuid	0x01 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	0x2901: "Characteristic User Description"
perm	0x01 0x00 0x00 0x00	0x00000001: Read Permission
max_len	0xFA 0x80	0x80FA: Trigger Read Indication: Yes Maximum attribute length: 250 bytes
{padding}	0x00 0x00	Can consist of any value
[Attribute 4] Characteristic B: characteristic declaration		
uuid	0x03 0x28 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	0x2803: "Characteristic Declaration"
perm	0x01 0x00 0x00 0x00	0x00000001: Read Permission
max_len	0x00 0x00	0x0000: Trigger Read Indication: No Maximum attribute length: 0
{padding}	0x00 0x00	Can consist of any value
[Attribute 5] Characteristic B: characteristic value declaration		
uuid	0x4F 0x43 0x31 0x3C 0x93 0x92 0x42 0xE6 0xA8 0x76 0xFA 0x3B 0xEF 0xB4 0x87 0x5A	0x5A87B4EF3BFA76A8E64292933C31434F
perm	0x49 0x82 0x08 0x00	0x00088249: Write Permission, Write Request Accepted, Read Permission, 128-bit UUID
max_len	0x64 0x00	0x0064:

Message stream		
		Trigger Read Indication: No Maximum attribute length: 100 bytes
[Attribute 6] Characteristic B: characteristic user description		
uuid	0x01 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	0x2901: "Characteristic User Description"
perm	0x01 0x00 0x00 0x00	0x00000001: Read Permission
max_len	0xFA 0x00	0x00FA: Trigger Read Indication: No Maximum attribute length: 250 bytes
{padding}	0x00 0x00	Can consist of any value
[Attribute 7] Characteristic B: client characteristic configuration declaration		
uuid	0x02 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	0x2902: "Client Characteristic Configuration"
perm	0x09 0x00 0x02 0x00	0x00020009: Write Permission, Write Request Accepted, Read Permission, 128-bit UUID
max_len	0x02 0x00	0x0002: Trigger Read Indication: No Maximum attribute length: 2 bytes
{padding}	0x00 0x00	Can consist of any value
Message string		
0x05 0x00 0x0B 0x0B 0x00 0x10 0x00 0xC0 0x00 0x00 0x00 0x04 0x00 0xCC 0x07 0x2F 0x2A 0x93 0xA6 0xBD 0xD8 0x41 0x52 0xAC 0x0B 0x10 0x99 0x2E 0xC6 0xFE 0xED 0x00 0x00 0x03 0x28 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x80 0x00 0x00 0x20 0xEE 0x8D 0x0C 0xE1 0xF0 0x4A 0x0C 0xB3 0x25 0xDC 0x53 0x6A 0x68 0x86 0x2D 0x09 0x00 0x0A 0x00 0x64 0x80 0x00 0x00 0x01 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0xFA 0x80 0x00 0x00 0x03 0x28 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x4F 0x43 0x31 0x3C 0x93 0x92 0x42 0xE6 0xA8 0x76 0xFA 0x3B 0xEF 0xB4 0x87 0x5A 0x49 0x82 0x08 0x00 0x64 0x00 0x00 0x00 0x01 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0xFA 0x00 0x00 0x00 0x02 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x09 0x00 0x02 0x00 0x02 0x00 0x00 0x00		

Table 71. Service permissions

Bit	Service permissions	0xCC = 11001100(b)
7	Primary Service	1 = Primary Service, 0 = Secondary Service
6..5	UUID Length	0 = 16 bits, 1 = 32 bits, 2 = 128 bits, 3 = RFU
4..2	Service Permission	0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH
1	Encryption key size must be 16 bytes	0: No, 1: Yes
0	Task that manages service is multi-instantiated (Connection index is conveyed)	0: No, 1: Yes

Table 72. Attribute permissions

Bit	Attribute permissions	Description
31..20	{reserved}	
19..18	UUID length	0 = 16 bits, 1 = 32 bits, 2 = 128 bits, 3 = RFU
17	Write Request accepted	
16	Write Signed accepted	
15	Write Command accepted	

Bit	Attribute permissions	Description
14	Encryption key Size must be 16 bytes	
13	Broadcast permission	Only relevant for a characteristic value
12	Extended properties present	Only relevant for a characteristic value
11..9	Notification Permission	0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH
8..6	Indication Permission	0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH
5..3	Write Permission	0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH
2..0	Read Permission	0 = Disable, 1 = Enable, 2 = UNAUTH, 3 = AUTH

Table 73. Attribute value max length and trigger read indication option

Bit	Property	Description
15	Trigger Read Indication	0 = Value present in Database 1 = Value not present in Database
14..0	Maximum Attribute Length	

Note 1 [bit 15]: For Included Services and Characteristic Declarations, the "Trigger Read Indication" bit contains the targeted handle.

Note 2 For Characteristic Extended Properties, the "Trigger Read Indication" bit contains 2-byte value.

Note 3 For Client Characteristic Configuration and Server Characteristic Configuration, the "Trigger Read Indication" bit is not used.

2. Confirmation of the service allocation.

   DA14585/531 responds to GATTM_ADD_SVC_REQ by sending the GATTM_ADD_SVC_RSP message. The start handle of the service is returned in this message. The GTL Host must store the value of the start handle to use it for the handling of the GATT events while the device is connected.

Table 74. GATTM_ADD_SVC_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ADD_SVC_RSP	0x0B01
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTM	0x000B
PAR_LEN	2	4	0x0004
Message parameters			
start_hdl	2	Start Handle of the service	0x0017
status	1	Operation status	**0x00**: ATT_ERR_NO_ERROR **0x01**: ATT_ERR_INVALID_HANDLE **0x40**: GAP_ERR_INVALID_PARAM **0x56**: GAP_ERR_INVALID_PERM
{padding}	1	{padding}	{padding} Ignored. Any value is valid.

Table 75. GATTM_ADD_SVC_RSP example

Message stream for GATTM_ADD_SVC_RSP example		
Symbolic message structure reference can be found in Table 74.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x0B	GATTM_ADD_SVC_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0B 0x00	TASK_ID_GATTM
PAR_LEN	0x04 0x00	4 bytes
start_hdl	0x17 0x00	The start handle of the service, assigned by DA14585/531 (0x0017 here)
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	Can consist of any value
Message string		
0x05 0x01 0x0B 0x10 0x00 0x0B 0x00 0x04 0x00 0x17 0x00 0x00 0x00		

5.3.8.3 GATT Service Permissions Management

Each service has a set of permissions defined along with the service declaration. These are listed in Table 71. To read or alter these permissions, the commands and procedures outlined in the following paragraphs can be used.

- GATTM_SVC_GET_PERMISSION_REQ and GATTM_SVC_GET_PERMISSION_RSP commands (Table 76, Table 77, and Table 78)

  The GATTM_SVC_GET_PERMISSION_REQ command (Table 76) is sent to DA14585/531 to get the GATT service permissions. The DA14585/531 responds to the request with the GATTM_SVC_GET_PERMISSION_RSP (Table 77) message to provide the permissions of the service.

Table 76. GATTM_SVC_GET_PERMISSION_REQ symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_SVC_GET_PERMISSION_REQ	0x0B02
DST_ID	2	TASK_ID_GATTM	0x000B
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	2 bytes	0x0002
Message parameters			
start_hdl	2	Service start attribute handle	The start handle of the service returned in GATTM_ADD_SVC_RSP (Table 75)

Table 77. GATTM_SVC_GET_PERMISSION_RSP symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_SVC_GET_PERMISSION_RSP	0x0B03
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTM	0x000B

Parameters	Bytes	Description	Data
Header			
PAR_LEN	2	4 bytes	0x0004
Message parameters			
start_hdl	2	Service start attribute handle	The start handle of the service returned in GATTM_ADD_SVC_RSP (Table 75)
perm	1	The service permissions	As detailed in Table 71, but masked with 10011111(b) (the UUID length field is masked-out)
status	1	Return status	**0x00:** ATT_ERR_NO_ERROR **0x01:** ATT_ERR_INVALID_HANDLE

Table 78. Example: querying service status

Request: GATTM_SVC_GET_PERMISSION_REQ		
Symbolic message structure reference can be found in Table 76.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x02 0x0B	GATTM_SVC_GET_PERMISSION_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0x00	0x0002: 2 bytes
start_hdl	0x17 0x00	0x0017 (23)
Message string		
0x05 0x02 0x0B 0x0B 0x00 0x10 0x00 0x02 0x00 0x17 0x00		
Response: GATTM_SVC_GET_PERMISSION_RSP		
Symbolic message structure reference can be found in Table 77.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x0B	GATTM_SVC_GET_PERMISSION_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0B 0x00	TASK_ID_GATTM
PAR_LEN	0x04 0x00	4 bytes
start_hdl	0x17 0x00	The attribute with handle 0x0017 (23) has been written
perm	0x8C	0x8C = 10001100(b) ▪ Bit 7: Primary Service: Yes ▪ Bits 6...5: {masked-out} ▪ Bits 4...2: 11(b)→AUTH ▪ Bit 1: Encryption key must be 16 bytes: no ▪ Bit 0: Task that manages service is multi-instantiated: No
status	0x00	ATT_ERR_NO_ERROR
Message string		
0x05 0x03 0x0B 0x10 0x00 0x0B 0x00 0x04 0x00 0x17 0x00 0x8C 0x00		

- GATTM_SVC_SET_PERMISSION_REQ and GATTM_SVC_SET_PERMISSION_RSP commands (Table 79, Table 80, and Table 81)

The GATTM_SVC_SET_PERMISSION_REQ command (Table 79) is sent to DA14585/531 to set the GATT service permissions. The DA14585/531 responds to the request with the GATTM_SVC_SET_PERMISSION_RSP message (Table 80).

Table 79. GATTM_SVC_SET_PERMISSION_REQ symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_SVC_SET_PERMISSION_REQ	0x0B04
DST_ID	2	TASK_ID_GATTM	0x000B
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	4 bytes	0x0004
Message parameters			
start_hdl	2	Service start attribute handle	The start handle of the service returned in GATTM_ADD_SVC_RSP (Table 75)
perm	1	The service permissions	See Table 71
{padding}	1	{padding}	Can consist of any value

Table 80. GATTM_SVC_SET_PERMISSION_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_SVC_SET_PERMISSION_RSP	0x0B05
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTM	0x000B
PAR_LEN	2	4 bytes	0x0004
Message parameters			
start_hdl	2	Service start attribute handle	The start handle of the service returned in GATTM_ADD_SVC_RSP (Table 75)
status	1	Return status	**0x00**: ATT_ERR_NO_ERROR **0x01**: ATT_ERR_INVALID_HANDLE
{padding}	1	{padding}	Can consist of any value

Table 81. Example: disabling a service

Request: GATTM_SVC_SET_PERMISSION_REQ		
Symbolic message structure reference can be found in Table 79.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x04 0x0B	GATTM_SVC_SET_PERMISSION_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x04 0x00	0x0004: 4 bytes
start_hdl	0x17 0x00	0x0017 (23)
perm	0xC0	0xC0 = 11000000(b)

Request: GATTM_SVC_SET_PERMISSION_REQ		
		▪ Bit 7: Primary Service ▪ Bits 6...5: UUID size: 128 bits ▪ Bits 4...2: 0→Disable ▪ Bit 1: Encryption key must be 16 bytes: no ▪ Bit 0: Task that manage service is multi-instantiated: no
{padding}	0x00	Can consist of any value
Message string		
0x05 0x04 0x0B 0x0B 0x00 0x10 0x00 0x04 0x00 0x17 0x00 0xC0		
Response: GATTM_SVC_SET_PERMISSION_RSP		
Symbolic message structure reference can be found in Table 80.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0B	GATTM_SVC_SET_PERMISSION_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0B 0x00	TASK_ID_GATTM
PAR_LEN	0x04 0x00	4 bytes
start_hdl	0x17 0x00	The attribute with handle 0x0017 (23) has been written
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	Can consist of any value
Message string		
0x05 0x05 0x0B 0x10 0x00 0x0B 0x00 0x04 0x00 0x17 0x00 0x00 0x00		

5.3.9 GATT Database Transactions

The GATT database attributes feature permissions configuration. The GTL Host can initiate the GATT database and you can read and write the values using the procedures in the following subsections.

5.3.9.1 Data Storage

Attribute data can be stored either in the DA14585/531 database or in the GTL Host. In database configuration, set the field "Trigger Field Indication - RI (Read Indication)" for an attribute to instruct the DA14585/531 to forward the read request to the GTL Host for a specific attribute. The attributes listed in Table 82 are always forwarded to the GTL Host (forced as RI), irrespective of the configuration options in the database declaration.

Table 82. Attributes for which read indication (RI) is enforced by DA14585/531

Type	ID	Mnemonic	Description
Characteristic	0x2902	ATT_DESC_CLIENT_CHAR_CFG	Client characteristic configuration
Characteristic	0x2903	ATT_DESC_SERVER_CHAR_CFG	Server characteristic configuration

Similarly, the characteristics listed in Table 83 are read-only and are forced not to get forwarded to the GTL Host (RI flag is discarded) and their value is read from the DA14585/531 database directly.

Table 83. Attributes which are read-only and enforced as non-RI

Type	ID	Mnemonic	Description
Declaration	0x2800	ATT_DECL_PRIMARY_SERVICE	Primary service Declaration
Declaration	0x2801	ATT_DECL_SECONDARY_SERVICE	Secondary service Declaration
Declaration	0x2802	ATT_DECL_INCLUDE	Include Declaration
Characteristic	0x2803	ATT_DECL_CHARACTERISTIC	Characteristic Declaration

Type	ID	Mnemonic	Description
Characteristic	0x2900	ATT_DESC_CHAR_EXT_PROPERTIES	Characteristic extended properties

The GTL Host cannot write in the DA14585/531 GATT database the value of an attribute that is declared as RI or is a type referred in Table 83. If this is attempted, the DA14585/531 response (GATTM_ATT_SET_VALUE_RSP) has a status of ATT_ERR_REQUEST_NOT_SUPPORTED.

5.3.9.2 Handles of Attributes

To access a value in the DA14585/531 database, the handle of the associated attribute is needed. Due to all attributes in the service being stored in the order in which they are placed in the database declaration, the index of the attribute can be used as the offset from the Start Handle of the service. Because the service creation is done with the start_handle parameter set to zero, the Service Start handle is assigned automatically and it is returned in the parameter start_hdl of the GATTM_ADD_SVC_RSP API message (see step "Confirmation of the service allocation" in Section 5.3.8.2).

Table 84. Handles of attributes in example custom service

Index	Attribute type	Read indicated (RI)	Handle (Note 1)	
			Hex	Dec
0	Service Declaration	{N.A.}	0x0017 (Note 2)	23
1	CHAR_A.{Characteristic Declaration}	{N.A.}	0x0018	24
2	CHAR_A.{Characteristic Value}	YES	0x0019	25
3	CHAR_A.{Characteristic User Description}	YES	0x001A	26
4	CHAR_B.{Characteristic Declaration}	{N.A.}	0x001B	27
5	CHAR_B.{Characteristic Value}	NO	0x001C	28
6	CHAR_B.{Characteristic User Description}	NO	0x001D	29
7	CHAR_B.{Client Characteristic Configuration}	{N.A.} (Always RI)	0x001E	30

Note 1 Values of a handle can change.

Note 2 0x0017 is the value assigned automatically in the example.

5.3.9.3 Writing Values to Database

Writing to the DA14585/531GATT database is triggered **only by the GTL Host** with commands in Table 85 and Table 86. The command GATTM_ATT_SET_VALUE_REQ writes data in an attribute of the database using the handle of the attribute. DA14585/531responds to that command with the GATTM_ATT_SET_VALUE_RSP message.

It is allowed to write the values of attributes that are not Read Indicate (see Section 5.3.9.1) to the DA14585/531 database with the GATTM_ATT_SET_VALUE_REQ command. This command cannot be used for RI attributes.

Table 85. GATTM_ATT_SET_VALUE_REQ symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ATT_SET_VALUE_REQ	0x0B0C
DST_ID	2	TASK_ID_GATTM	0x000B
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	{length} + 4	
Message parameters			
handle	2	The handle of the attribute to write	Start Handle of the Service + index of the attribute
length	2	Length of {value} in bytes	Max. length is 512 bytes

Parameters	Bytes	Description	Data
Header			
Value	{length}	Attribute Value - the data to write	

Table 86. GATTM_ATT_SET_VALUE_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ATT_SET_VALUE_RSP	0x0B0D
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTM	0x000B
PAR_LEN	2	4 bytes	0x0004
Message parameters			
Handle	2	The handle of the attribute of which the value has been written	Must be the same as the handle parameter of Table 85
status	1	Operation status	**0x00**: ATT_ERR_NO_ERROR **0x06**: ATT_ERR_REQUEST_NOT_SUPPORTED **0x07**: ATT_ERR_INVALID_OFFSET **0x0D**: ATT_ERR_INVALID_ATTRIBUTE_VAL_LEN
{padding}	1	{padding} Ignored. Any value is valid.	{padding}

5.3.9.3.1 Example Database Attribute Values Initialization

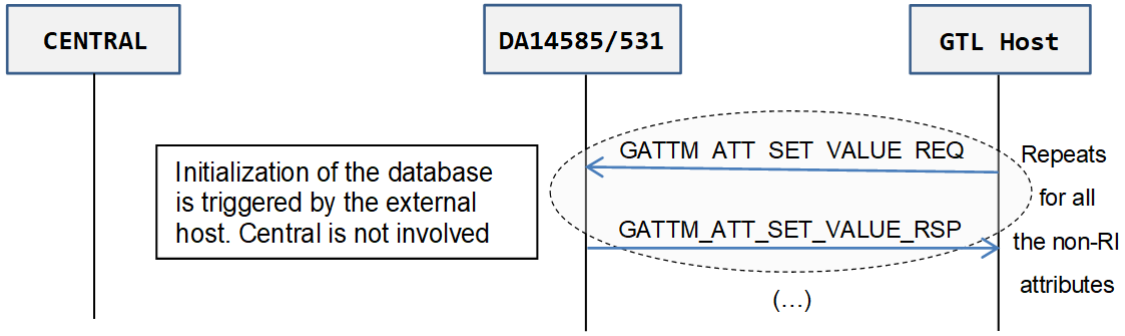


Figure 14. Database value initialization

Because in the example database the attributes of the second characteristic (characteristic B) are not RI, they get initialized by the following two steps:

1. Step1: CHAR_B.{Characteristic Value}

Table 87. Setting the characteristic value of characteristic B in the example database

Request: GATTM_ATT_SET_VALUE_REQ - attribute write		
Symbolic message structure reference can be found in Table 85.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0C 0B	GATTM_ATT_SET_VALUE_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x3A 0x00	0x003A: 58 bytes

Request: GATTM_ATT_SET_VALUE_REQ - attribute write		
handle	0x1C 0x00	0x001C (28)
length	0x36 0x00	0x0036: The value is 54 bytes long
value	0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x42 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x76 0x61 0x6C 0x75 0x65 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D	{characteristic B}.{characteristic value} -demo value-
Message string		
0x05 0x0C 0x0B 0x0B 0x00 0x10 0x00 0x3A 0x00 0x1C 0x00 0x36 0x00 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x42 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x76 0x61 0x6C 0x75 0x65 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D		
Response: GATTM_ATT_SET_VALUE_RSP - confirmation of attribute write		
Symbolic message structure reference can be found in Table 86.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0D 0x0B	GATTM_ATT_SET_VALUE_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0B 0x00	TASK_ID_GATTM
PAR_LEN	0x04 0x00	4 bytes
handle	0x1C 0x00	The attribute with handle 0x001C (28) has been written
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	Can consist of any value
Message string		
0x05 0x0D 0x0B 0x10 0x00 0x0B 0x00 0x04 0x00 0x1C 0x00 0x00 0x00		

2. Step 2: CHAR_B. {Characteristic User Description}

Table 88. Setting characteristic user description of characteristic B in example database

Request: GATTM_ATT_SET_VALUE_REQ - attribute write		
Symbolic message structure reference can be found in Table 85.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0C 0B	GATTM_ATT_SET_VALUE_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x45 0x00	0x0045: 69 bytes
handle	0x1D 0x00	0x001D (29)
length	0x41 0x00	0x0041: The value is 65 bytes long
value	0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x42 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x75 0x73 0x65 0x72 0x20 0x64 0x65 0x73 0x63 0x72 0x69 0x70 0x74 0x69 0x6F 0x6E 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D	{characteristic B}.{characteristic user description} -demo value-
Message string		
0x05 0x0C 0x0B 0x0B 0x00 0x10 0x00 0x45 0x00 0x1D 0x00 0x41 0x00 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x42 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x75 0x73 0x65 0x72 0x20 0x64 0x65 0x73 0x63 0x72 0x69 0x70 0x74 0x69 0x6F 0x6E 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D		
Response: GATTM_ATT_SET_VALUE_RSP - confirmation of attribute write		
Symbolic message structure reference can be found in Table 86.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0D 0x0B	GATTM_ATT_SET_VALUE_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0B 0x00	TASK_ID_GATTM
PAR_LEN	0x04 0x00	4 bytes
handle	0x1D 0x00	The attribute with handle 0x001D (29) has been written
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	Can consist of any value
Message string		
0x05 0x0D 0x0B 0x10 0x00 0x0B 0x00 0x04 0x00 0x1D 0x00 0x00 0x00		

5.3.9.4 Attribute Value Write Triggered by Peer Central

Peer central can write attribute values using:

- Write Command (Write without Response)
  - When a peer writes an attribute with Write Command (Write without Response), no acknowledgment or error indication is sent to the central by DA14585/531
  - Only for attribute sizes from 0 to ATT_MTU-3
  - When a Write Command (Write without Response) is received, the Bluetooth LE application sends the GATTC_WRITE_REQ_IND indication to the GTL Host
- Write Request (Figure 15)

- A response is sent to the central by DA14585/531
- Only for attribute sizes from 0 to ATT_MTU-3

- Prepare Write Request/Execute Write Request (Figure 16)

  For attribute sizes longer than ATT_MTU-3, the value is sent in chunks of "0...ATT_MTU-3" using two or more "Prepare Write Request" commands, and an "Execute Write Request" is sent at the end. The assembly of the values transferred by the Prepare Write Requests is done by DA14585/531. This way, only one GATTC_WRITE_REQ_IND indication (Table 90) is sent to the GTL Host, so that the GTL Host provides the complete value when the Execute Write Request is received

- For a write operation to be valid for an attribute, certain permissions must be enabled in the database declaration for the specific attribute (Table 89). After each "Prepare Write Request" sent by the peer central, DA14585/531 sends the command GATTC_ATT_INFO_REQ_IND (Table 92) to the GTL Host. Then, GTL Host must respond with the command GATTC_ATT_INFO_CFM (Table 93)

When a "Write Request" or a "Prepare Write Request/Execute Write Request" is received, the Bluetooth LE application sends the GATTC_WRITE_REQ_IND indication (Table 90) to the GTL Host. GTL Host must respond with the GATTC_WRITE_CFM indication (Table 91).

Table 89. Permissions mandatory for an attribute to support peer write operations

Command	Mandatory permissions
Write Command	WR, PERM_MASK_WRITE_COMMAND
Write Request	WR, WRITE_REQ
Prepare Write Request/Execute Write Request	WR, WRITE_REQ

NOTE
It is allowed to write the values of attributes that are not Read Indicate (see Section 5.3.9.1) to the DA14585/531 database with the GATTM_ATT_SET_VALUE_REQ command. This command cannot be used for RI attributes.

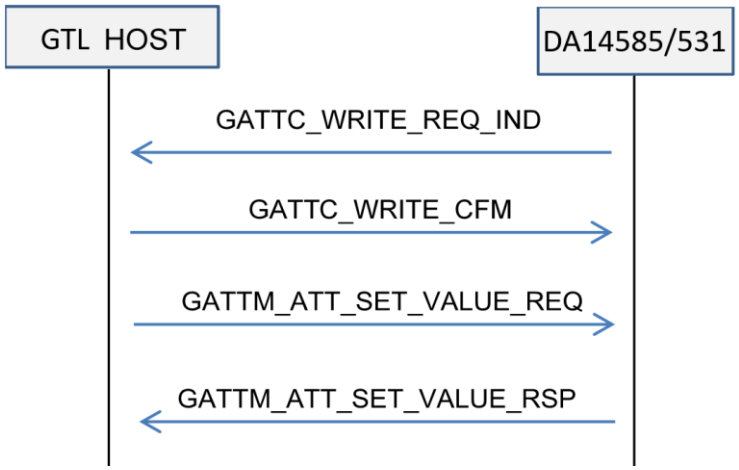


Figure 15. Peer writing attribute values using a write command or a write request

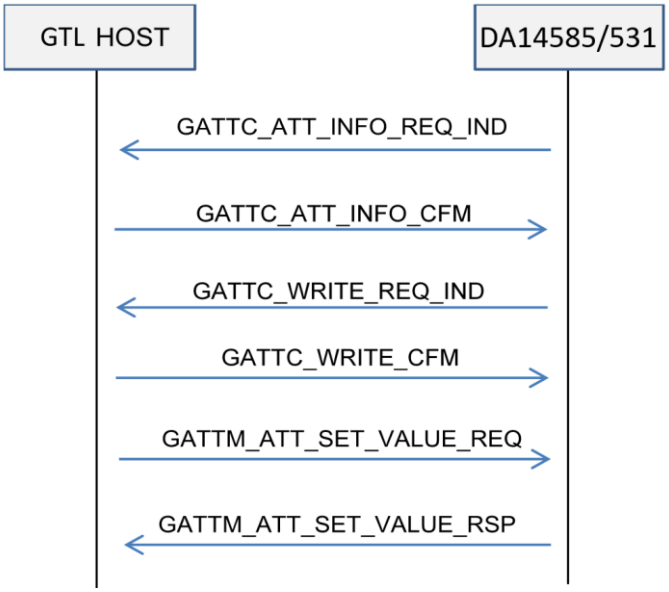


Figure 16. Peer writing attribute values using a prepare write request

GATTC_WRITE_REQ_IND Indication and GATTC_WRITE_CFM Response

When a peer device sends "Write Command (Write without Response)", "Write Request" or "Prepare Write Request/Execute Write Request", the Bluetooth LE application sends the GATTC_WRITE_REQ_IND indication (Table 90) to the GTL Host. GTL Host responds to this indication by sending the GATTC_WRITE_CFM response (Table 91) to the Bluetooth LE application.

Table 90. GATTC_WRITE_REQ_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_WRITE_REQ_IND	0x0C15
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
PAR_LEN	2	The length of the parameters in bytes	{6 + length}
Message parameters			
handle	2	Handle of the attribute that must be written	
offset	2	Offset at which the data must be written	
length	2	Length of the data to be written	Max. Length is ATT_MTU-3
value	{length}	Data to be written	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 91. GATTC_WRITE_CFM symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_WRITE_CFM	0x0C16
DST_ID	2	nn, TASK_ID_GATTC	0xnn0C (nn = conidx) (Note 1)
SRC_ID	2	TASK_ID_GTL	0x0010

Parameters	Bytes	Description	Data
PAR_LEN	2	4 bytes	0x0004
Message parameters			
handle	2	Handle of the attribute written (LSB)	
status	1	Status of the write command execution	**0x00**: ATT_ERR_NO_ERROR **0x05**: ATT_ERR_INSUFF_AUTHEN **0x06**: ATT_ERR_REQUEST_NOT_SUPPORTED **0x0C**: ATT_ERR_INSUFF_ENC_KEY_SIZE **0x0F**: ATT_ERR_INSUFF_ENC
{padding}	1	Ignored. Any value is valid.	{padding}

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

GATTC_ATT_INFO_REQ_IND Indication and GATTC_ATT_INFO_CFM Response

This message is sent by the Bluetooth LE application to the GTL Host to request attribute info. It only applies to "Prepare Write Request/Execute Write Request". It can be triggered during prepare write to check whether attribute modification is authorized by profile/application or not and to get current attribute length. The GTL Host responds to GATTC_ATT_INFO_REQ_IND by sending GATTC_ATT_INFO_CFM to the Bluetooth LE application.

Table 92. GATTC_ATT_INFO_REQ_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_ATT_INFO_REQ_IND	0x0C17
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
PAR_LEN	2	2 bytes	0x0002
Message parameters			
handle	2	Handle of the attribute for which data is requested (LSB)	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 93. GATTC_ATT_INFO_CFM symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_ATT_INFO_CFM	0x0C18
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	The length of the parameters in bytes.	0x0006
Message parameters			
handle	2	Handle of the attribute to which the request refers (LSB).	
length	2	The current length of the attribute (LSB).	

Parameters	Bytes	Description	Data
Header			
status	1	Status of the write command execution.	**0x00**: ATT_ERR_NO_ERROR, write proceeds **0x03**: ATT_ERR_WRITE_NOT_PERMITTED, write does not proceed {other error codes}: write does not proceed
{padding}	1	{padding} Ignored. Any value is valid.	{padding}

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

5.3.9.4.1 Example: Peer Central Writes Attribute Values

1. Step 1: Peer central retrieves data from the GTL Host.

   Only when attribute values with a length of more than MTU-3 bytes are written (Prepare Write Requests), the peer central retrieves data from the GTL Host to check whether attribute modification is authorized or not and to get the current attribute length.

Table 94. Checking attribute modification authorization and getting current attribute length

Request: GATTC_ATT_INFO_REQ_IND		
Symbolic message structure reference can be found in Table 92.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x17 0x0C	GATTC_ATT_INFO_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x02 0x00	0x0002: 2 bytes
handle	0x1C 0x00	0x001C (28)
Message string		
0x05 0x17 0x0C 0x10 0x00 0x0C 0x00 0x02 0x00 0x1C 0x00		
Response: GATTC_ATT_INFO_CFM		
Symbolic message structure reference can be found in Table 93.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x18 0x0C	GATTC_ATT_INFO_CFM
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x06 0x00	6 bytes
handle	0x1C 0x00	Data refers to the attribute with handle 0x001C (28)
length	0x00 0x00	The current length of the attribute is zero
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	Can consist of any value
Message string		
0x05 0x16 0x0C 0x0C 0x00 0x10 0x00 0x04 0x00 0x1C 0x00 0x00 0x00		

2. Step 2: Peer central writes a value:
   a. Using "Write Command" or "Write Request" or "Prepare Write Request/Execute Write Request".

Table 95. Peer central writes a value using "write request" or "prepare write request/execute write request"

Request: GATTC_WRITE_REQ_IND		
Symbolic message structure reference can be found in Table 90.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x15 0x0C	GATTC_WRITE_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x19 0x00	0x0019: 25 bytes
handle	0x1C 0x00	0x001C (28)
offset	0x00 0x00	Write starts from the first byte
length	0x13 0x00	0x0013: The value is 19 bytes long
value	0x57 0x72 0x69 0x74 0x74 0x65 0x6E 0x20 0x62 0x79 0x20 0x63 0x65 0x6E 0x74 0x72 0x61 0x6C 0x21	"Written by central!"
Message string		
0x05 0x15 0x0C 0x10 0x00 0x0C 0x00 0x19 0x00 0x1C 0x00 0x00 0x00 0x13 0x00 0x57 0x72 0x69 0x74 0x74 0x65 0x6E 0x20 0x62 0x79 0x20 0x63 0x65 0x6E 0x74 0x72 0x61 0x6C 0x21		
Response: GATTC_WRITE_CFM		
Symbolic message structure reference can be found in Table 91.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x16 0x0C	GATTC_WRITE_CFM
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x04 0x00	4 bytes
handle	0x1C 0x00	Attribute with handle 28 has been written
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	Can consist of any value
Message string		
0x05 0x16 0x0C 0x0C 0x00 0x10 0x00 0x04 0x00 0x1C 0x00 0x00 0x00		

3. Step 3 (optional): Data is written to the database.

Table 96. Setting a value of an attribute in the database from GTL host - example

Request: GATTM_ATT_SET_VALUE_REQ - attribute write		
Symbolic message structure reference can be found in Table 84.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0C 0B	GATTM_ATT_SET_VALUE_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x17 0x00	0x0017: 19 bytes

Request: GATTM_ATT_SET_VALUE_REQ - attribute write		
handle	0x1C 0x00	0x001C (28)
length	0x36 0x00	0x0036: The value is 54 bytes long
value	0x57 0x72 0x69 0x74 0x74 0x65 0x6E 0x20 0x62 0x79 0x20 0x63 0x65 0x6E 0x74 0x72 0x61 0x6C 0x21	"Written by central!"
Message string		
0x05 0x0C 0x0B 0x0B 0x00 0x10 0x00 0x17 0x00 0x1C 0x00 0x13 0x00 0x57 0x72 0x69 0x74 0x74 0x65 0x6E 0x20 0x62 0x79 0x20 0x63 0x65 0x6E 0x74 0x72 0x61 0x6C 0x21		
Response: GATTM_ATT_SET_VALUE_RSP - confirmation of attribute write		
Symbolic message structure reference can be found in Table 85.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0D 0x0B	GATTM_ATT_SET_VALUE_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0B 0x00	TASK_ID_GATTM
PAR_LEN	0x04 0x00	4 bytes
handle	0x1C 0x00	The attribute with handle 0x001C (28) has been written
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	Can consist of any value
Message string		
0x05 0x0D 0x0B 0x10 0x00 0x0B 0x00 0x04 0x00 0x1C 0x00 0x00 0x00		

5.3.9.5 Reading Values from DA14585/531GATT Database

Reading attribute values from the database by the GTL Host can be performed using:

- The GATTM_ATT_GET_VALUE_REQ command (Table 97): it reads an attribute value from the database using the handle of the attribute.
- The GATTM_ATT_GET_VALUE_RSP command (Table 98): the DA14585/531 responds to the GATTM_ATT_GET_VALUE_REQ command with this command.

Table 97. GATTM_ATT_GET_VALUE_REQ symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ATT_GET_VALUE_REQ	0x0B0A
DST_ID	2	TASK_ID_GATTM	0x000B
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Length of the parameters in bytes.	0x0002
Message parameters			
handle	2	The handle of the attribute to be read from the database.	Start Handle of the Service + index of the attribute.

Table 98. GATTM_ATT_GET_VALUE_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05

Parameters	Bytes	Description	Data
Header			
MSG_ID	2	GATTM_ATT_GET_VALUE_RSP	0x0B0B
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTM	0x000B
PAR_LEN	2	Length of the parameters in bytes.	{length} + 6 bytes
Message parameters			
handle	2	The handle of the attribute which has been written.	Must be the same as the handle parameter of the associated GATTM_ATT_GET_VALUE_REQ command.
length	2	The length of the value in bytes.	Max. length is 512 bytes.
status	1	Operation status.	**0x00:** ATT_ERR_NO_ERROR **0x01:** ATT_ERR_INVALID_HANDLE **0x06:** ATT_ERR_REQUEST_NOT_SUPPORTED **0x80:** ATT_ERR_APP_ERROR
value	{length}	The current value of the attribute.	
{padding}	1	Ignored. Any value is valid.	{padding}

5.3.9.5.1 Example: Reading an Attribute Value from DA14585/531 GATT Database

In the example database, the characteristic value attribute of the second characteristic (characteristic B) is not RI, so it is stored in the database and can be read.

Table 99. Example: reading an attribute value from the DA14585/531 database

Request: GATTM_ATT_GET_VALUE_REQ - attribute read		
Symbolic message structure reference can be found in Table 97.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0A 0B	GATTM_ATT_GET_VALUE_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0x00	0x0002: 2 bytes
handle	0x1C 0x00	0x001C (28)
Message string		
0x05 0x0A 0x0B 0x0B 0x00 0x10 0x00 0x02 0x00 0x1C 0x00		
Response: GATTM_ATT_GET_VALUE_RSP - response to attribute read		
Symbolic message structure reference can be found in Table 98.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0B 0x0B	GATTM_ATT_SET_VALUE_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0B 0x00	TASK_ID_GATTM
PAR_LEN	0x3C 0x00	0x003C: 60 bytes
handle	0x1C 0x00	The attribute with handle 0x001C (28) has been written
length	0x36 0x00	0x0036: 54 bytes
status	0x00	ATT_ERR_NO_ERROR
	0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x42 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x76 0x61 0x6C 0x75 0x65 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D	{characteristic B}.{characteristic value} -demo value-
{padding}	0x00	Can consist of any value
Message string		
0x05 0x0B 0x0B 0x10 0x00 0x0B 0x00 0x3C 0x00 0x1C 0x00 0x36 0x00 0x00 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x42 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x76 0x61 0x6C 0x75 0x65 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D 0x00		

5.3.9.6 Attribute Value Read by Peer Central

5.3.9.6.1 Introduction

When the peer central requests to read an attribute:

- If the attribute is stored in the DA14585/531 GATT database (not RI), no message is sent to the GTL Host.
- If the attribute is not stored in the database (RI, Figure 17), the value has to be requested by the GTL Host (Table 100). A response is sent to DA14585/531, containing the current value of the attribute.

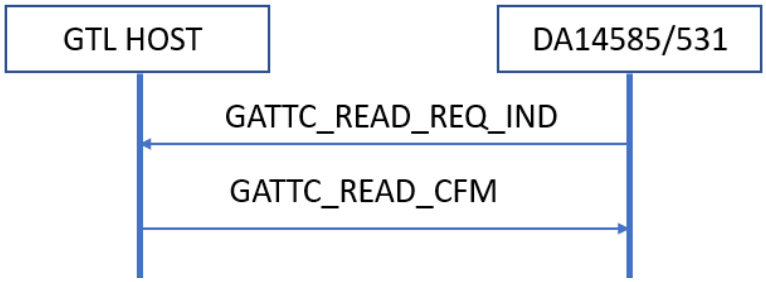


Figure 17. Peer reading RI values

The GATTC_READ_REQ_IND message is sent to the GTL Host when the peer central reads the value of an attribute which is Read Indicated (RI). The GTL Host responds to GATTC_READ_REQ_IND by sending GATTC_READ_CFM to the Bluetooth LE application.

Table 100. GATTC_READ_REQ_IND (only for RI attributes) symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_READ_REQ_IND	0x0C13
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC (91Note 1)	0xnn0C (nn = conidx)
PAR_LEN	2	2 bytes	0x0002
Message parameters			
handle	2	Handle of the attribute for which data is requested (LSB)	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 101. GATTC_READ_CFM command (only for RI attributes) symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_READ_CFM	0x0C14
DST_ID	2	nn, TASK_ID_GATTC	0xnn0C (nn = conidx) (Note 1)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	The length of the parameters in bytes	{length of value} + 6
Message parameters			
handle	2	Handle of the attribute to which the request refers (LSB)	
length	2	The current length of the attribute (LSB)	Max. length is ATT_MTU-1 (about ATT_MTU, see Ref. [1])

Parameters	Bytes	Description	Data
status	1	Status of the write command execution	**0x00**: ATT_ERR_NO_ERROR **0x06**: ATT_ERR_REQUEST_NOT_SUPPORTED
value	{length}	The current value of the attribute	
{padding}	1	Ignored. Any value is valid.	{padding}

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

5.3.9.6.2 Example: Attribute Value Read by Peer Central

1. Step 1: DA14585/531 sends a read request to the GTL Host.

Table 102. GATTC_READ_REQ_IND example

Request: GATTC_READ_REQ_IND		
Symbolic message structure reference can be found in Table 100.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x13 0x0C	GATTC_READ_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x02 0x00	0x0002: 2 bytes
handle	0x19 0x00	0x0019 (25)
Message string		
0x0C 0x10 0x00 0x0C 0x00 0x02 0x00 0x19 0x00		

2. Step 2: GTL Host responds to provide the value.

Table 103. GATTC_READ_CFM value

Response: GATTC_READ_CFM		
Symbolic message structure reference can be found in Table 101.		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x14 0x0C	GATTC_READ_CFM
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x3C 0x00	60 bytes
handle	0x19 0x00	The value of the attribute with handle 0x0019 (25) is requested
length	0x36 0x00	0x0036: 54 bytes
status	0x00	ATT_ERR_NO_ERROR
value	0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x41 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x76 0x61 0x6C 0x75 0x65 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D	{characteristic A}.{characteristic value} -demo value-
{padding}	0x00	Can consist of any value
Message string		
0x05 0x14 0x0C 0x0C 0x00 0x10 0x00 0x3C 0x00 0x19 0x00 0x36 0x00 0x00 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x41 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x76 0x61 0x6C 0x75 0x65 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D 0x00		

5.3.9.7 Notifications and Indications

The peripheral (GATT server) can notify the central of a characteristic value.

Two options are available:

- Sending a notification: no confirmation from the central.
- Sending an indication: a confirmation is sent by the central.

In both cases, the command GATTC_SEND_EVT_CMD is used.

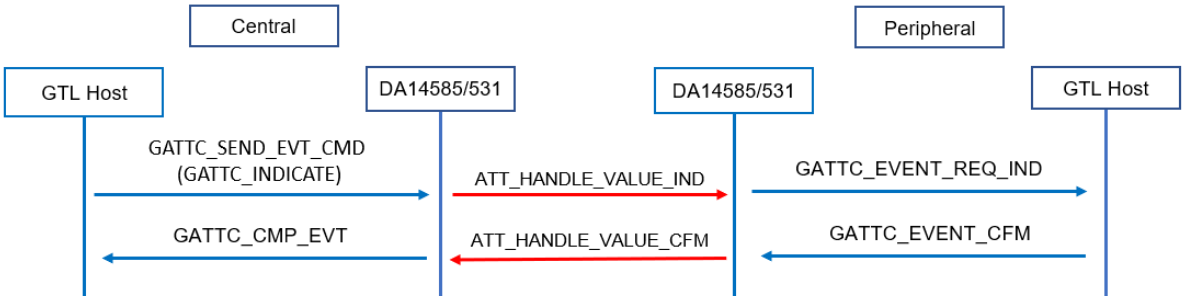


Figure 18. GATTC_SENT_EVT_CMD with an indication

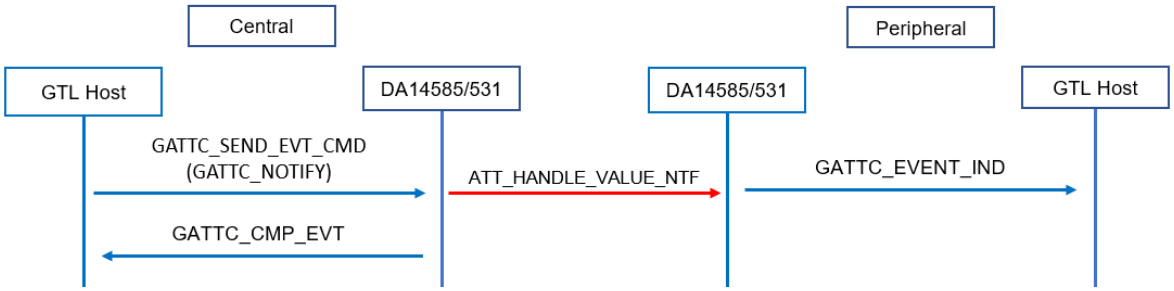


Figure 19. GATTC_SENT_EVT_CMD with a notification

NOTE
If the value of an attribute for which the notification is sent is stored in the DA14585/531 GATT database (attribute is not RI), it can be updated before the notification is sent (see Section 5.3.9.3), but this is application specific.

1. Step 1: The notification (or indication) is sent to the central.

   To have the notification (or indication) sent to the central, the GTL Host sends the command GATTC_SEND_EVT_CMD of the proper GATTC request type to the Bluetooth LE application.

Table 104. GATTC_SEND_EVT_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_SEND_EVT_CMD	0x0C10
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Length of the parameters	{length} + 8
Message parameters			
operation	1	GATTC request type	**0x12**: GATTC_NOTIFY: To send a notification: **0x13**: GATTC_INDICATE: To send an indication:
{padding}	1	Ignored. Any value is valid.	{padding}
seq_num	2	Operation sequence number	
handle	2	The handle of the attribute	
length	2	The length of the *value* parameter	Max. length is ATT_MTU-3 (about ATT_MTU, see Ref. [1])
value	{length}	The value to send to the GTL Host	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 105. GATTC_SEND_EVT_CMD example - notification

Message stream - sending a notification – GATTC_SEND_EVT_CMD		
Symbolic message structure reference can be found in Table 104.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x10 0x0C	GATTC_SEND_EVT_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x3E 0x00	0x003E: 62 bytes
operation	0x12	GATTC_NOTIFY
{padding}	0x00	{padding} Ignored. Any value is valid.
seq_num	0x01 0x00	operation sequence number: 0x0001
handle	0x1C 0x00	0x001C
length	0x36 0x00	0x0036: 54 bytes
value	0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x42 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x76 0x61 0x6C 0x75 0x65 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D	The value to send to the GTL Host
Message string		
0x05 0x10 0x0C 0x0C 0x00 0x10 0x00 0x3E 0x00 0x12 0x00 0x01 0x00 0x1C 0x00 0x36 0x00 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x42 0x7D 0x2E 0x7B 0x63 0x68 0x61 0x72 0x61 0x63 0x74 0x65 0x72 0x69 0x73 0x74 0x69 0x63 0x20 0x76 0x61 0x6C 0x75 0x65 0x7D 0x20 0x2D 0x64 0x65 0x6D 0x6F 0x20 0x76 0x61 0x6C 0x75 0x65 0x2D		

Table 106. GATTC_SEND_EVT_CMD example - indication

Example message stream - sending an indication – GATTC_SEND_EVT_CMD		
Symbolic message structure reference can be found in Table 104.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x10 0x0C	GATTC_SEND_EVT_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0C 0x00	12 bytes
operation	0x13	GATTC_INDICATE
{padding}	0x00	{padding} Ignored. Any value is valid.
seq_num	0x01 0x00	operation sequence number: 0x0001
handle	0x12 0x00	Handle 0x0012
length	0x04 0x00	4 bytes
value	0x01 0x00 0xFF 0xFF	The value to send to the GTL Host
Message string		
0x05 0x10 0x0C 0x0C 0x00 0x10 0x00 0x0C 0x00 0x13 0x00 0x01 0x00 0x12 0x00 0x04 0x00 0x01 0x00 0xFF 0xFF		

2. Step 2: Completion event for the notification or indication.

- When the operation parameter is set to GATTC_NOTIFY and the notification of step 1 has been sent over air, the Bluetooth LE application sends the GATTC_CMP_EVT message
- When the operation parameter is set to GATTC_INDICATE and a confirmation of an indication being correctly received by a peer device arrives, the Bluetooth LE application sends the GATTC_CMP_EVT message

Table 107. GATTC_CMP_EVT for GATTC_NOTIFY/GATTC_INDICATE example

Message stream (GATTC_CMP_EVT) on completion of GATTC_SEND_EVT_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x12/0x13	GATTC_NOTIFY/GATTC_INDICATE
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x01 0x00	Sequence number 0x0001
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 **0x12** 0x00 0x01 0x00, in case of **GATTC_NOTIFY** 0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 **0x13** 0x00 0x01 0x00, in case of **GATTC_INDICATE**		

5.3.9.8 GATTC_EVENT_IND

When a peripheral sends a **notification** to a central, the Bluetooth LE application running on DA14585/531 of the central sends **GATTC_EVENT_IND** to its GTL Host.

Table 108. GATTC_EVENT_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_EVENT_IND	0x0C0C
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
PAR_LEN	2	Length of the parameters	0xXXXX (depending on the value)
Message parameters			
type	1	Event Type	0xXX
{padding}	1	Ignored. Any value is valid.	0xXX
length	2	Data Length	0xXXXX
handle	2	Attribute Handle	0xXXXX
value	{length}	The value of the notification	0xXXXX…

Table 109. GATTC_EVENT_IND – example

Message stream (GATTC_EVENT_IND)		
Symbolic message structure reference can be found in Table 108.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0C 0x0C	GATTC_EVENT_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x0E 0x00	0x000E = 14 bytes
type	0x12	GATTC_NOTIFY: Send an attribute notification.
{padding}	0x00	{padding} Ignored. Any value is valid.
length	0x08 0x00	8 bytes
handle	0x28 0x00	Handle is 0x0028 = 40
value	0x00 0x10 0x00 0x01 0x00 0x00 0x00 0x00	
Message string		
0x05 0x0C 0x0C 0x10 0x00 0x0C 0x00 0x0E 0x00 0x12 0x00 0x08 0x00 0x28 0x00 0x00 0x10 0x00 0x01 0x00 0x00 0x00 0x00		

5.3.9.9 GATTC_EVENT_REQ_IND Indication and GATTC_EVENT_CFM Response

5.3.9.9.1 GATTC_EVENT_REQ_IND

This message is sent by the Bluetooth LE application of DA14585/531 to the GTL Host to send an indication. External host responds to GATTC_EVENT_REQ_IND by sending the GATTC_EVENT_CFM to the Bluetooth LE application. This message is triggered to a registered task and contains a new value of a peer attribute handle. This event must be acknowledged by the GATTC_EVENT_CFM message (see Section 5.3.9.9.2).

Table 110. GATTC_EVENT_REQ_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_EVENT_REQ_IND	0x0C0D
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTC	0x000C
PAR_LEN	2	Depending on value length.	0xXX 0XX
Message parameters			
type	1	GATTC_INDICATE	0x13
{padding}	1	{padding} Ignored. Any value is valid.	0xXX
length	2	Length of the notification.	0xXX 0xXX
handle	2	Attribute Handle	0xXX 0xXX
value	Depending on value length	Event Value	0xXXxxxxx

Table 111. GATT_EVENT_REQ_IND - example

Message stream (GATTC_EVENT_REQ_IND)		
Symbolic message structure reference can be found in Table 110.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0D 0x0C	GATTC_EVENT_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC
PAR_LEN	0x0A 000	Depending on value length
type	0x13	GATTC_INDICATE
{padding}	0x00	{padding} Ignored. Any value is valid
length	0x04 0x00	4 bytes
handle	0x12 x00	Handle = 0x0012
value	0x01 0x00 0xFF 0xFF	Value
Message string		
0x05 0x0D 0x0C 0x10 0x00 0x0C 0x00 0x0A 000 0x13 0x00 0x04 0x00 0x12 x00 0x01 0x00 0xFF 0xFF		

5.3.9.9.2 GATTC_EVENT_CFM

After the GTL Host receives GATTC_EVENT_REQ_IND, the answer it sends to the DA14585/531 is GATTC_EVENT_CFM.

Table 112. GATTC_EVENT_CFM symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_EVENT_CFM	0x0C0E
DST_ID	2	TASK_ID_GATTC	0x000C
SRC_ID	2	TASK_ID_GTL	0x0010

Parameters	Bytes	Description	Data
Header			
PAR_LEN	2	2 bytes	0x0002
Message parameters			
handle	2	Handle of the attribute	

Table 113. GATTC_EVENT_CFM - example

Message stream (GATTC_EVENT_REQ_IND)		
Symbolic message structure reference can be found in Table 112.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0E 0x0C	GATT_EVENT_CFM
DST_ID	0x0C 0x00	TASK_ID_GATTC
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0x00	2 bytes
type	0x13 0x00	GATTC_INDICATE
Message string		
0x05 0x0E 0x0C 0x0C 0x00 0x10 0x00 0x02 0x00 0x13 0x00		

5.3.9.10 GATT Client Commands

5.3.9.10.1 GATTC_DISC_CMD

The GTL Host can trigger GATTC_DISC_CMD for various Discovery Operations.

Table 114. GATTC_DISC_CMD symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_DISC_CMD	0x0C03
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	**0x000A**: 16-bit UUID. (Also used by operations not requiring uuid)	10 bytes
		0x0018:128-bit UUID	24 bytes
Message parameters			
operation	1	GATTC_DISC_ALL_SVC	0x02
		GATTC_DISC_BY_UUID_SVC	0x03
		GATTC_DISC_INCLUDED_SVC	0x04
		GATTC_DISC_ALL_CHAR	0x05
		GATTC_DISC_BY_UUID_CHAR	0x06
		GATTC_DISC_DESC_CHAR	0x07
uuid_len	1	2 bytes for 16-bit UUID (Also used by operations not requiring uuid)	0x02
		16 bytes for 128-bit UUID	0x10

Parameters	Bytes	Description	Hex data
Header			
seq_num	2	The sequence number	0xXXXX (arbitrary value)
start_hdl	2	The start handle	0xXXXX
end_hdl	2	The end handle	0xXXXX
uuid (ignored by operations not requiring uuid)	2	The 16-bit UUID	0xXXXX
	16	The 128-bit UUID	0xXXXXXXXXXXXXXXXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

GATTC_DISC_CMD with operation GATTC_DISC_DESC_CHAR

The GTL Host can send GATTC_DISC_CMD with operation GATTC_DISC_DESC_CHAR to trigger discovery for attributes in a specific range of handles.

1. Step 1: GTL Host sends GATTC_DISC_CMD to retrieve the handle-UUID pairs for attributes that correspond to a certain range of handles.

Table 115. GATTC_DISC_CMD with operation GATTC_DISC_DESC_CHAR - example

Message stream - GATTC_DISC_CMD with operation GATTC_DISC_DESC_CHAR		
Symbolic message structure reference can be found in Table 114.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x0C	GATTC_DISC_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	0x000A = 10 bytes
operation	0x07	GATTC_DISC_DESC_CHAR
uuid_len	0x02	UUID not used in this command
seq_num	0x20 0x00	0x0020: Sequence Number 20 (example value - arbitrary)
start_hdl	0x1A 0x00	0x001A: Handle 26
end_hdl	0x1C 0x00	0x001C: Handle 28
uuid	0x00 0x00	Ignored by this operation. Any value is valid.
Message string		
0x05 0x03 0x0C 0x0C 0x00 0x10 0x00 0x08 0x00 0x07 0x00 0x20 0x00 0x1A 0x00 0x1C 0x00 0x00 0x00		

2. Step 2: Bluetooth LE application sends a GATTC_DISC_CHAR_DESC_IND for the first attribute found.

Table 116. GATTC_DISC_CHAR_DESC_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_DISC_CHAR_DESC_IND	0x0C07
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
PAR_LEN	2	Length of the parameters	

Parameters	Bytes	Description	Data
Message parameters			
attr_hdl	2	The handle of the attribute	0xXXXX
uuid_len	1	**0x02:** (16-bit UUID or operations not requiring uuid)	2 bytes
		0x10: (128-bit UUID)	16 bytes
uuid	2 (16-bit UUID)	0xXXXX (16-bit UUID)	
	16 (128-bit UUID)	0xXXXXXXXXXXXXXXXX (128-bit UUID)	
{padding}	1	{padding} Ignored. Any value is valid.	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 117. GATTC_DISC_CHAR_DESC_IND – example (response #1)

Message stream (GATTC_DISC_CHAR_DESC_IND)		
Symbolic message structure reference can be found in Table 116.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x07 0x0C	GATTC_DISC_CHAR_DESC_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x06 0x00	0x0006 = 6 bytes
attr_hdl	0x1A 0x00	0x001A: Handle 26
uuid_len	0x02	(16-bit UUID)
uuid	0x03 0x28	UUID for "Characteristic Declaration" - 0x2803
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x07 0x0C 0x10 0x00 0x0C 0x00 0x06 0x00 0x1A 0x00 0x02 0x03 0x28 0x00		

3. Step 3: The Bluetooth LE application sends GATTC_DISC_CHAR_DESC_IND for the second attribute found.

Table 118. GATTC_DISC_CHAR_DESC_IND – example (response #2)

Message stream (GATTC_DISC_CHAR_DESC_IND)		
Symbolic message structure reference can be found in Table 116.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x07 0x0C	GATTC_DISC_CHAR_DESC_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x14 0x00	0x0014 = 20 bytes
attr_hdl	0x1B 0x00	0x001B: Handle 27
uuid_len	0x10	(128-bit UUID)
uuid	0xD9 0xD9 0xAA 0xFD 0xBD 0x9B 0x21 0x98 0xA8 0x49 0xE1 0x45 0xF3 0xDB 0xD1 0x69	ANCS Control Point: 0x69D1D8F345E149A898219BBDFDAAD9D9
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x07 0x0C 0x10 0x00 0x0C 0x00 0x14 0x00 0x1B 0x00 0x10 0xD9 0xD9 0xAA 0xFD 0xBD 0x9B 0x21 0x98 0xA8 0x49 0xE1 0x45 0xF3 0xDB 0xD1 0x69 0x00		

4. Step 4: The Bluetooth LE application sends GATTC_DISC_CHAR_DESC_IND for the third attribute found.

Table 119. GATTC_DISC_CHAR_DESC_IND – example (response #3)

Message stream (GATTC_DISC_CHAR_DESC_IND)		
Symbolic message structure reference can be found in Table 116.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x07 0x0C	GATTC_DISC_CHAR_DESC_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x06 0x00	0x0006 = 6 bytes
attr_hdl	0x1C 0x00	0x001C: Handle 28
uuid_len	0x02	(16-bit UUID)
uuid	0x00 0x29	UUID for "Characteristic Extended Properties" - 0x2900
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x07 0x0C 0x10 0x00 0x0C 0x00 0x06 0x00 0x1C 0x00 0x02 0x00 0x29 0x00		

5. Step 5: The Bluetooth LE application sends GATTC_CMP_EVT for GATTC_DISC_CMD.

   As there are no other attributes in the requested range, the Bluetooth LE application sends the GATTC_CMP_EVT indication for the GATTC_DISC_DESC_CHAR operation, with status ATT_ERR_ATTRIBUTE_NOT_FOUND.

Table 120. GATTC_CMP_EVT for GATTC_DISC_CMD example

Message stream (GATTC_CMP_EVT) on completion of GATTC_DISC_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x07	GATTC_DISC_DESC_CHAR
status	0x0A	ATT_ERR_ATTRIBUTE_NOT_FOUND
seq_num	0x20 0x00	Sequence number 0x0020
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x07 0x0A 0x20 0x00		

GATTC_DISC_CMD with Operation GATTC_DISC_ALL_SVC or GATTC_DISC_BY_UUID_SVC

The GTL Host can send GATTC_DISC_CMD with operation GATTC_DISC_ALL_SVC to trigger discovery for all available services or GATTC_DISC_BY_UUID_SVC to trigger the discovery of all services corresponding to a UUID in a handle range that is passed as a parameter. Figure 20 shows the transactions between the GTL host and the Bluetooth LE application for a GATTC_DISC_CMD with operation GATTC_DISC_ALL_SVC.

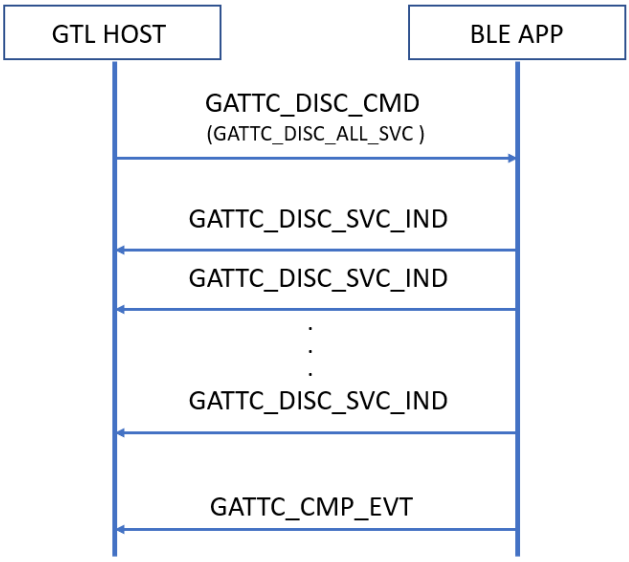


Figure 20. Transactions for GATTC_DISC_CMD with operation GATTC_DISC_ALL_SVC

1. Step 1: The GTL Host sends GATTC_DISC_CMD with an operation of GATTC_DISC_ALL_SVC to discover all services.

Table 121. GATTC_DISC_CMD with operation GATTC_DISC_ALL_SVC - example

Message stream: GATTC_DISC_CMD with operation GATTC_DISC_ALL_SVC		
Symbolic message structure reference can be found in Table 114.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x0C	GATTC_DISC_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	0x000A = 10 bytes
operation	0x02	GATTC_DISC_ALL_SVC
uuid_len	0x02	2 bytes
seq_num	0x06 0x00	0x0006: Sequence Number 6 (example value - arbitrary)
start_hdl	0x01 0x00	0x0001: Start Handle = 1
end_hdl	0xFF 0xFF	0xFFFF: End Handle = 65535 (search for all possible handles)
uuid	0x00 0x00	Ignored by this operation. Any value is valid.
Message string		
0x05 0x03 0x0C 0x0C 0x00 0x10 0x00 0x0A 0x00 0x02 0x02 0x06 0x00 0x01 0x00 0xFF 0xFF 0x00 0x00		

2. Step 2: The Bluetooth LE application sends **GATTC_DISC_SVC_IND** indications for all the attributes found.

Table 122. GATTC_DISC_SVC_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_DISC_SVC_IND	0x0C04
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
PAR_LEN	2	8 when parameter uuid is 16-bit UUID	0x0008
		22 when parameter uuid is 128-bit UUID	0x0016
Message parameter			
start_hdl	2	The start handle	0xXXXX
end_hdl	2	The end handle	0xXXXX
uuid_len	1	Length of 16-bit UUID	0x02
		Length of 128-bit UUID	0x10
uuid	2	The 16-bit UUID	0xXXXX
	16	The 128-bit UUID	0xXXXXXXXXXXXXXXXX
{padding}	1	{padding} Ignored. Any value is valid.	0xXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 123. GATTC_DISC_SVC_IND example

Message stream (GATTC_DISC_SVC_IND)		
Symbolic message structure reference can be found in Table 122.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x04 0x0C	GATTC_DISC_SVC_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x08 0x00	8 bytes
start_hdl	0x01 0x00	0x0001: Handle 1
end_hdl	0x05 0x00	0x0005: Handle 5
uuid_len	0x02	16-bit UUID
uuid	0x01 0x18	UUID: 0x1801
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x04 0x0C 0x10 0x00 0x0C 0x00 0x08 0x00 0x01 0x00 0x05 0x00 0x02 0x18 0x00		

3. Step 3: The Bluetooth LE application sends **GATTC_CMP_EVT** for **GATTC_DISC_CMD** when no other attributes are found in the requested range. The GATTC_CMP_EVT indication for the GATTC_DISC_ALL_SVC or GATTC_DISC_BY_UUID_SVC operation has a status of **ATT_ERR_ATTRIBUTE_NOT_FOUND**.

Table 124. GATTC_CMP_EVT for GATTC_DISC_CMD example

Message stream (GATTC_CMP_EVT) on completion of GATTC_DISC_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x02	GATTC_DISC_ALL_SVC
status	0x0A	ATT_ERR_ATTRIBUTE_NOT_FOUND
seq_num	0x06 0x00	Sequence number 0x0006
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x02 0x0A 0x06 0x00		

GATTC_DISC_CMD with Operation GATTC_DISC_INCLUDED_SVC

The GTL Host can send GATTC_DISC_CMD with operation GATTC_DISC_INCLUDED_SVC to trigger discovery for all available included services, shows the transactions between the GTL host and the Bluetooth LE application for GATTC_DISC_CMD with operation GATTC_DISC_INCLUDED_SVC.

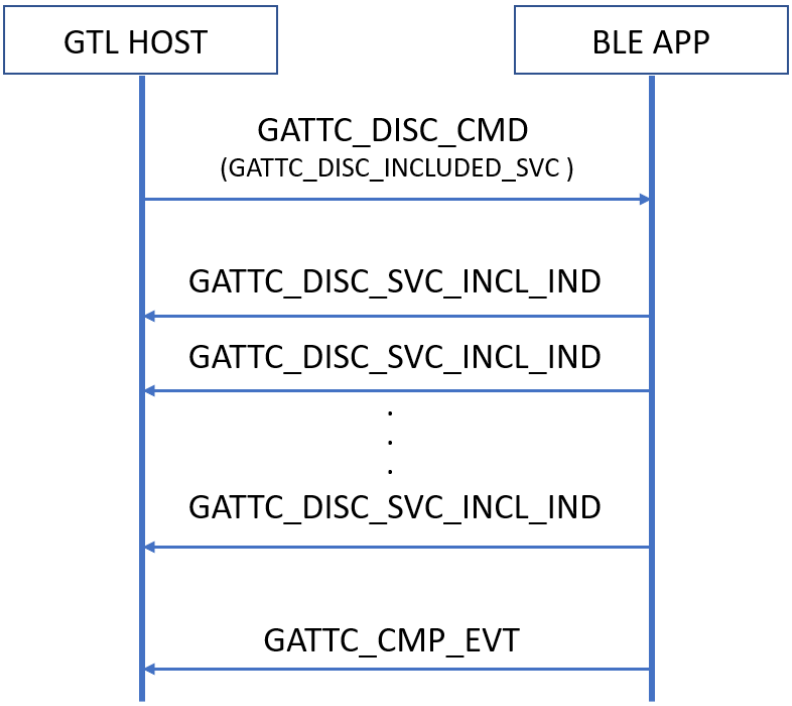


Figure 21. Transactions for GATTC_DISC_CMD with operation GATTC_DISC_INCLUDED_SVC

1. Step 1: External Host sends the GATTC_DISC_CMD with an operation of GATTC_DISC_INCLUDED_SVC to discover all included services in the given range.

Table 125. GATTC_DISC_CMD with operation GATTC_DISC_INCLUDED_SVC - example

Message stream GATTC_DISC_CMD with operation GATTC_DISC_INCLUDED_SVC		
Symbolic message structure reference can be found in Table 114.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x0C	GATTC_DISC_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	0x000A = 10 bytes
operation	0x04	GATTC_DISC_INCLUDED_SVC
uuid_len	0x02	2 bytes
seq_num	0x00 0x00	0x0000: Sequence Number 0x00 (example value - arbitrary)
start_hdl	0x01 0x00	0x0001: Start Handle = 1
end_hdl	0xFF 0xFF	0xFFFF: End Handle = 65535 (search for all possible handles)
uuid	0x00 0x00	Ignored by this operation. Any value is valid.
Message string		
0x05 0x03 0x0C 0x0C 0x00 0x10 0x00 0x0A 0x00 0x04 0x02 0x00 0x00 0x01 0x00 0xFF 0xFF 0x00 0x00		

2. Step 2: The Bluetooth LE application sends the GATTC_DISC_SVC_INCL_IND indications for the included services found.

Table 126. G ATTC_DISC_SVC_INCL_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_DISC_SVC_INCL_IND	0x0C05
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
PAR_LEN	2	10 or 24 bytes	0x000A or 0x0018
Message parameter			
attr_hdl	2	Attribute handle	0xXXXX
start_hdl	2	The start handle	0xXXXX
end_hdl	2	The end handle	0xXXXX
uuid_len	1	16-bit UUID	0xXX
		128-bit UUID	0xXX
uuid	2	The 16-bit UUID	0xXXXX
	16	The 128-bit UUID	0xXXXXXXXXXXXXXXXX
{padding}	1	{padding} Ignored. Any value is valid.	0xXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 127. GATTC_DISC_SVC_INCL_IND example #1

Message stream GATTC_DISC_SVC_INCL_IND		
Symbolic message structure reference can be found in Table 126.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0C	GATTC_DISC_SVC_INCL_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x0A 0x00	10 Bytes
attr_hdl	0x15 0x00	Attribute Handle = 21
start_hdl	0x09 0x00	Start Handle = 9
end_hdl	0x0C 0x00	End Handle = 12
uuid_len	0x02	16-bit UUID
uuid	0x0F 0x18	UUID is 0x180F
{padding}	0x00	0x00
Message string		
0x05 0x05 0x0C 0x0C 0x00 0x10 0x00 0x0A 0x00 0x15 0x00 0x09 0x00 0x0C 0x00 0x02 0x0F 0x18 0x00		

Table 128. GATTC_DISC_SVC_INCL_IND example #2

Message stream GATTC_DISC_SVC_INCL_IND		
Symbolic message structure reference can be found in Table 126.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0C	GATTC_DISC_SVC_INCL_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x0A 0x00	10 Bytes
attr_hdl	0x16 0x00	Attribute Handle = 22
start_hdl	0x0D 0x00	Start Handle = 13
end_hdl	0x13 0x00	End Handle = 19
uuid_len	0x02	16-bit UUID
uuid	0x0A 0x18	0x180A
{padding}	0x00	{padding} Ignored. Any value is valid
Message string		
0x05 0x05 0x0C 0x0C 0x00 0x10 0x00 0x0A 0x00 0x16 0x00 0x0D 0x00 0x13 0x00 0x02 0x0A 0x18 0x00		

3. Step 3: The Bluetooth LE application sends **GATTC_CMP_EVT** for **GATTC_DISC_CMD** when no other attributes are found in the requested range. The GATTC_CMP_EVT indication for the GATTC_DISC_INCLUDED_SVC operation has a status of **ATT_ERR_ATTRIBUTE_NOT_FOUND**.

Table 129. GATTC_CMP_EVT for GATTC_DISC_CMD example

Message stream (GATTC_CMP_EVT) on completion of GATTC_DISC_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC
PAR_LEN	0x04 0x00	4 bytes
operation	0x04	GATTC_DISC_INCLUDED_SVC
status	0x0A	ATT_ERR_ATTRIBUTE_NOT_FOUND
seq_num	0x00 0x00	Sequence number 0x0000
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x02 0x0A 0x00 0x00		

GATTC_DISC_CMD with Operation GATTC_DISC_ALL_CHAR or GATTC_DISC_BY_UUID_CHAR

The GTL Host can send GATTC_DISC_CMD with operation GATTC_DISC_ALL_CHAR to discover all characteristics in a given handle range and with operation GATTC_DISC_BY_UUID_CHAR to discover specific searched UUID in a provided handle range.

1. Step 1: The GTL Host sends GATTC_DISC_CMD with an operation of GATTC_DISC_ALL_CHAR to discover all characteristics.

Table 130. GATTC_DISC_CMD with operation GATTC_DISC_ALL_CHAR - example

Message stream GATTC_DISC_CMD with operation GATTC_DISC_ALL_SVC		
Symbolic message structure reference can be found in **Table 114**.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x0C	GATTC_DISC_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	0x000A = 10 bytes
operation	0x05	GATTC_DISC_ALL_CHAR
uuid_len	0x02	16-bit UUID
seq_num	0x8 0x00	0x0008: Sequence Number 8 (example value - arbitrary)
start_hdl	0x01 0x00	0x0001: Handle 1
end_hdl	0xFF 0xFF	0xFFFF: Handle 65535 (search for all possible handles)
uuid	0x00 0x00	Ignored by this operation. Any value is valid.
Message string		
0x05 0x03 0x0C 0x0C 0x00 0x10 0x00 0x0A 0x00 0x05 0x02 0x8 0x00 0x01 0x00 0xFF 0xFF 0x00 0x00		

2. Step 2: The Bluetooth LE application sends the GATTC_DISC_CHAR_IND indication for each attribute found.

Table 131. GATTC_DISC_CHAR_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_DISC_CHAR_IND	0x0C06
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTC	0x000C
PAR_LEN	2	8 when parameter uuid is 16-bit UUID 22 when parameter uuid is 128-bit UUID	0x0008
Message parameter			
attr_hdl	2	Attribute Handle	0xXXXX
pointer_hdl	2	Pointer attribute handle to UUID	0xXXXX
prop	1	Properties	0xXXXX
uuid_len	1	0x02 (16-bit UUID)	0xXXXX
		0x10 (128-bit UUID)	0xXXXX…
uuid	2 (16-bit UUID)	The 16-bit UUID	0xXXXX
	16 (128-bit UUID)	The 128-bit UUID	0xXXXX….

Table 132. GATTC_DISC_CHAR_IND example

Message stream (GATTC_DISC_CHAR_IND)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0C	GATTC_DISC_CHAR_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GAPC
PAR_LEN	0x08 0x00	8 bytes
attr_hdl	0x02 0x00	Attribute Handle: 0x0002
pointer_hdl	0x03 0x00	Pointer Handle: 0x0003
prop	0x02	Properties: 0x02 = ATT_CHAR_PROP_RD
uuid_len	0x02	16-bit UUID
uuid	0x00 0x2A	0x2A00
Message string		
0x05 0x06 0x0C 0x10 0x00 0x0C 0x00 0x08 0x00 0x02 0x00 0x03 0x00 0x02 0x02 0x00 0x2A		

3. Step 3: The Bluetooth LE application sends **GATTC_CMP_EVT** for **GATTC_DISC_CMD** when no other attributes are found in the requested range. The **GATTC_CMP_EVT** indication for the GATTC_DISC_ALL_CHAR or GATTC_DISC_BY_UUID_CHAR operation has the status parameter set to **ATT_ERR_ATTRIBUTE_NOT_FOUND.**

Table 133. GATTC_CMP_EVT for GATTC_DISC_CMD example

Message stream (GATTC_CMP_EVT) on completion of GATTC_DISC_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC
PAR_LEN	0x04 0x00	4 bytes
operation	0x05	GATTC_DISC_ALL_CHAR
status	0x0A	ATT_ERR_ATTRIBUTE_NOT_FOUND
seq_num	0x08 0x00	Sequence number 0x0008
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x05 0x0A 0x08 0x00		

5.3.9.10.2 GATTC_SDP_SVC_DISC_CMD

This command can be used to perform a service discovery using GATTC discovery procedures. This discovery can search all (**GATTC_SDP_DISC_SVC_ALL**) or a specific (**GATTC_SDP_DISC_SVC**) service. It automatically searches for included services, characteristics, and descriptors. This procedure can be canceled using the GATTC_SDP_DISC_CANCEL operation code.

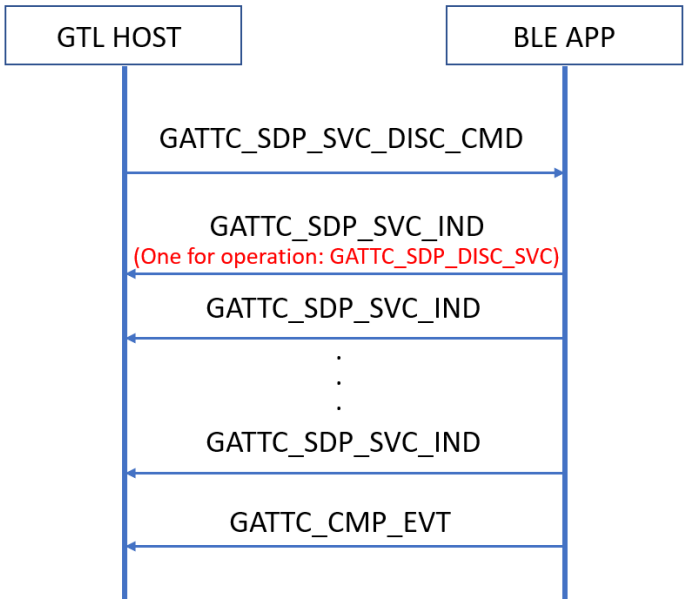


Figure 22. Discovery procedure with GATTC_SDP_SVC_CMD for operations GATTC_SDP_DISC_SVC and GATTC_SDP_DISC_SVC_ALL

Table 134. GATTC_SDP_SVC_DISC_CMD symbolic message structure

Parameters	Bytes	Data	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_SDP_SVC_DISC_CMD	0x0C19
DST_ID	2	TASK_ID_GATTC	0x000C
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Depends on the uuid	0xXXXX
Message parameters			
operation	1	GATTC_SDP_DISC_SVC	0x15
		GATTC_SDP_DISC_SVC_ALL	0x16
		GATTC_SDP_DISC_CANCEL	0x17
uuid_len	1	16-bit UUID	0x02
		32-bit UUID	0x04
		128-bit UUID	0x10
seq_num	2	Operation sequence number	0xXX
start_hdl	2	Start Handle	0xXX 0xXX
end_hdl	2	End Handle	0xXX 0xXX
uuid	2	16-bit UUID	0xXXXX
	4	32-bit UUID	0xXXXXXXXX
	16	128-bit UUID	0xXXXXXXXXXXXXXXXX

Table 135. GATTC_SDP_SVC_IND symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_SDP_SVC_IND	0x0C1A
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTC	0x000C
PAR_LEN	2	Depending on the operation	0xXXXX
Message parameters			
uuid_len	1	Service UUID Length	0xXX
uuid	2, 4 or 16	Service UUID	
{padding}	15 if uuid_len = 2 13 if uuid_len = 4 1 if uuid_len = 16	Padding due to fixed allocation for max 128-bits UUIDs +1 one byte for alignment	{padding} Ignored
start_hdl	2	Service Start Handle	0xXX 0xXX
end_hdl	2	Service End Handle	0xXX 0xXX
info.att_char.att_type	1	GATTC_SDP_ATT_CHAR	0x02
info.att_char.prop	1	Value Property	0xXX
info.att_char.handle	2	Value Handle	0xXX
{padding}	18	Padding to the maximum allocated value of 22 bytes	{padding} Ignored
info.inc_svc.att_type	1	GATTC_SDP_INC_SVC	0x01
info.inc_svc.uuid_len	1	Included Service UUID length	0xXX
info.inc_svc.uuid	2, 4 or 16	Included Service UUID	
{padding}	18 if uuid_len = 2 16 if uuid_len = 4 4 if uuid_len = 16	Padding to the maximum allocated value of 22 bytes.	{padding}
info.inc_svc.start_hdl	2	Included Service UUID Start Handle	0xXX
info.inc_svc.end_hdl	2	Included Service UUID End Handle	0xXX
info.att.type	1	GATTC_SDP_ATT_VAL	0x03
info.att.uuid_len	1	Attribute UUID length	0xXX
info.att.uuid	2, 4 or 16	Attribute UUID	
{padding}	18 if uuid_len = 2 16 if uuid_len = 4 4 if uuid_len = 16	Padding to the maximum allocated value of 22 bytes.	{padding}
info.att.type	1	GATTC_SDP_ATT_DESC	0x04
info.att.uuid_len	1	Attribute UUID length	0xXX
info.att.uuid	2, 4 or 16	Attribute UUID	
{padding}	18 if uuid_len = 2 16 if uuid_len = 44 if uuid_len = 16	Padding to the maximum allocated value of 22 bytes	{padding}

GATTC_SDP_SVC_DISC_CMD with Operation GATTC_SDP_DISC_SVC

1. Step 1: The GTL host sends **GATTC_SDP_SVC_DISC_CMD** to DA145858/531.

Table 136. GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC example #1

Message stream GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC		
Symbolic message structure reference can be found in Table 134.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x19 0x0C	GATTC_SDP_SVC_DISC_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x1A 0x00	26 Bytes
operation	0x15	GATTC_SDP_DISC_SVC
uuid_len	0x02	UUID len
seq_num	0x20 0x00	Example value - arbitrary
start_hdl	0x01 0x00	0x0001: Handle 1
end_hdl	0xFF 0xFF	0xFFFF: Handle 65535 (search for all possible handles)
uuid	0x01 0x18	UUID (ATT_SVC_GENERIC_ATTRIBUTE)
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	Padding to the maximum allocated value of 26 bytes
Message string		
0x05 0x19 0x0C 0x0C 0x00 0x10 0x00 0x1A 0x00 0x15 0x02 0x20 0x00 0x01 0x00 0xFF 0xFF 0x01 0x18 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

2. Step 2: DA145858/531 responds with the **GATTC_SDP_SVC_IND** indication.

Table 137. GATTC_SDP_SVC_IND example #1

Message stream GATTC_SDP_SVC_IND		
Symbolic message structure reference can be found in Table 135.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1A 0x0C	GATTC_SDP_SVC_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC
PAR_LEN	0x58 0x00	88 bytes
uuid_len	0x02	ATT_UUID_16_LEN = 2 bytes
uuid	0x01 0x18	UUID
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
start_hdl	0x06 0x00	
end_hdl	0x09 0x00	
info.att_char.att_type	0x02	GATTC_SDP_ATT_CHAR
info.att_char.prop	0x20	
info.att_char.handle	0x08 0x00	
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att.type	0x03	GATTC_SDP_ATT_VAL
info.att.uuid_len	0x02	ATT_UUID_16_LEN = 2 bytes
info.att.uuid	0x05 0x2A	0x2A05 = Service Changed
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att.type	0x04	GATTC_SDP_ATT_DESCL
info.att.uuid_len	0x02	ATT_UUID_16_LEN = 2 bytes
info.att.uuid	0x02 0x29	Client Characteristic Configuration
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x1A 0x0C 0x10 0x00 0x0C 0x00 0x58 0x00 0x02 0x01 0x18 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x06 0x00 0x09 0x00 0x02 0x20 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x03 0x02 0x05 0x2A 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x04 0x02 0x02 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

3. Step 3: The Bluetooth LE application sends GATTC_CMP_EVT to the GTL Host when the operation has been completed.

Table 138. GATTC_CMP_EVT after a GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC

Message stream (GATTC_CMP_EVT) on completion of GATTC_SDP_SVC_DISC_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC
PAR_LEN	0x04 0x00	4 bytes
operation	0x15	GATTC_SDP_DISC_SVC
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x00 0x00	Sequence number 0x0000
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x15 0x00 0x00 0x00		

Table 139. GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC example #2

Message stream GATTC_SDP_SVC_DISC_CMD (GATTC_SDP_SVC_DISC_CMD)		
Symbolic message structure reference can be found in Table 134.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x19 0x0C	GATTC_SDP_SVC_DISC_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x18 0x00	26 Bytes
operation	0x15	GATTC_SDP_DISC_SVC
uuid_len	0x10	16 Bytes
seq_num	0x20 0x00	Sequence Number custom-used by the application
start_hdl	0x01 0x00	Start handle
end_hdl	0xFF 0xFF	End handle
uuid	0xD0 0x00 0x2D 0x12 0x1E 0x4B 0x0F 0xA4 0x99 0x4E 0xCE 0xB5 0x31 0xF4 0x05 0x79	UUID
Message string		
0x05 0x19 0x0C 0x0C 0x00 0x10 0x00 0x18 0x00 0x15 0x10 0x20 0x01 0x00 0xFF 0xFF 0xD0 0x00 0x2D 0x12 0x1E 0x4B 0x0F 0xA4 0x99 0x4E 0xCE 0xB5 0x31 0xF4 0x05 0x79		

Table 140. GATTC_SDP_SVC_IND example #2

Message stream GATTC_SDP_SVC_IND		
Symbolic message structure reference can be found in Table 135.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1A 0x0C	GATTC_SDP_SVC_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC
PAR_LEN	0xDC 0x00	220 bytes
uuid_len	0x10	ATT_UUID_128_LEN = 16Bytes
uuid	0xD0 0x00 0x2D 0x12 0x1E 0x4B 0x0F 0xA4 0x99 0x4E 0xCE 0xB5 0x31 0xF4 0x05 0x79	UUID
{padding}	0x00	{padding} Ignored. Any value is valid.
start_hdl	0x23 0x00	
end_hdl	0x2C 0x00	
info.att_char.att_type	0x02	GATTC_SDP_ATT_CHAR
info.att_char.prop	0x88	ATT_CHAR_PROP_EXT_PROP(0x80) | ATT_CHAR_PROP_WR(0x08)
info.att_char.handle	0x25 0x00	Handle = 37
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att.type	0x03	GATTC_SDP_ATT_VAL
info.att.uuid_len	0x10	ATT_UUID_128_LEN = 16 Bytes
info.att.uuid	0xD9 0xD9 0xAA 0xFD 0xBD 0x9B 0x21 0x98 0xA8 0x49 0xE1 0x45 0xF3 0xD8 0xD1 0x69	
{padding}	0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att.type	0x04	GATTC_SDP_ATT_DESCL
info.att.uuid_len	0x02	ATT_UUID_16_LEN = 2Bytes
info.att.uuid	0x00 0x29	UUID
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att_char.att_type	0x02	GATTC_SDP_ATT_CHAR
info.att_char.prop	0x10	ATT_CHAR_PROP_NTF
info.att_char.handle	0x28 0x00	Handle = 40
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att.type	0x03	GATTC_SDP_ATT_VAL
info.att.uuid_len	0x10	ATT_UUID_128_LEN = 16 Bytes

Message stream GATTC_SDP_SVC_IND		
info.att.uuid	0xBD 0x1D 0xA2 0x99 0xE6 0x25 0x58 0x8C 0xD9 0x42 0x01 0x63 0x0D 0x12 0xBF 0x9F	UUID
{padding}	0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att.type	0x04	GATTC_SDP_ATT_DESCL
info.att.uuid_len	0x02	ATT_UUID_16_LEN = 2Bytes
info.att.uuid	0x02 0x29	UUID
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att_char.att_type	0x02	GATTC_SDP_ATT_CHAR
info.att_char.prop	0x10	ATT_CHAR_PROP_NTF
info.att_char.handle	0x2B 0x00	Handle = 43
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att.type	0x03	GATTC_SDP_ATT_VAL
info.att.uuid_len	0x10	ATT_UUID_128_LEN = 16 Bytes
info.att.uuid	0xFB 0x7B 0x7C 0xCE 0x6A 0xB3 0x44 0xBE 0xB5 0x4B 0xD6 0x24 0xE9 0xC6 0xEA 0x22	UUID
{padding}	0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
info.att.type	0x04	GATTC_SDP_ATT_DESCL
info.att.uuid_len	0x02	ATT_UUID_16_LEN = 2Bytes
info.att.uuid	0x02 0x29	UUID
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x1A 0x0C 0x10 0x00 0x0C 0x00 0xDC 0x00 0x10 0xD0 0x00 0x2D 0x12 0x1E 0x4B 0x0F 0xA4 0x99 0x4E 0xCE 0xB5 0x31 0xF4 0x05 0x79 0x00 0x23 0x00 0x2C 0x00 0x02 0x88 0x25 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x03 0x10 0xD9 0xD9 0xAA 0xFD 0xBD 0x9B 0x21 0x98 0xA8 0x49 0xE1 0x45 0xF3 0xD8 0xD1 0x69 0x00 0x00 0x00 0x00 0x04 0x02 0x00 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02 0x10 0x28 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x03 0x10 0xBD 0x1D 0xA2 0x99 0xE6 0x25 0x58 0x8C 0xD9 0x42 0x01 0x63 0x0D 0x12 0xBF 0x9F 0x00 0x00 0x00 0x00 0x04 0x02 0x02 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02 0x10 0x2B 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x03 0x10 0xFB 0x7B 0x7C 0xCE 0x6A 0xB3 0x44 0xBE 0xB5 0x4B 0xD6 0x24 0xE9 0xC6 0xEA 0x22 0x00 0x00 0x00 0x00 0x04 0x02 0x02 0x29 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	

The GTL Host sends GATTC_CMP_EVT with an operation of GATTC_SDP_DISC_SVC when the procedure is finished (see Table 138).

GATTC_SDP_SVC_DISC_CMD with Operation GATTC_SDP_DISC_SVC_ALL

1. Step 1: The GTL Host sends **GATTC_SDP_SVC_DISC_CMD** to DA145858/531.

Table 141. GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC_ALL example

Message stream GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC_ALL		
Symbolic message structure reference can be found in Table 134.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x19 0x0C	GATTC_SDP_SVC_DISC_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x1A 0x00	26 Bytes
operation	0x16	GATTC_SDP_DISC_SVC_ALL
uuid_len	0x00	Not used in SVC_ALL GATTC_SDP_DISC_SVC_ALL operation
seq_num	0x24 0x00	Sequence Number custom-used by the application
start_hdl	0x01 0x00	0x0001: Handle 1
end_hdl	0xFF 0xFF	0xFFFF: Handle 65535 (search for all possible handles)
uuid	0x00 0x00	Not used in SVC_ALL GATTC_SDP_DISC_SVC_ALL operation
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	
Message string		
0x05 0x19 0x0C 0x0C 0x00 0x10 0x00 0x18 0x00 0x16 0x00 0x24 0x00 0x01 0x00 0xFF 0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

2. Step 2: DA145858/531 responds with the **GATTC_SDP_SVC_IND** indications. If **GATTC_SDP_SVC_DISC_CMD** has the **GATTC_SDP_DISC_SVC** operation, DA14585/531 sends only one GATTC_SDP_SVC_IND. With an operation of **GATTC_SDP_DISC_SVC_ALL** (if there are more services) more than one GATTC_SDP_SVC_IND arrives to the GTL Host sent by DA14585/531.

   Examples of GATTC_SDP_SVC_IND can be seen in Table 137 and Table 140.

3. Step 3: The Bluetooth LE application sends GATTC_CMP_EVT to the GTL Host when the operation has been completed.

Table 142. GATTC_CMP_EVT after a GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC_ALL

Message stream (GATTC_CMP_EVT) on completion of GATTC_SDP_SVC_DISC_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC
PAR_LEN	0x04 0x00	4 bytes
operation	0x16	GATTC_SDP_DISC_SVC_ALL
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x24 0x00	Sequence number 0x0024
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x16 0x00 0x24 0x00		

GATTC_SDP_SVC_DISC_CMD with Operation GATTC_SDP_DISC_SVC_CANCEL

When the GTL Host is in the process of discovering, it can also send GATTC_SDP_SC_DISC_CMD with an operation of GATTC_SDP_DISC_SVC_CANCEL to cancel the operation.

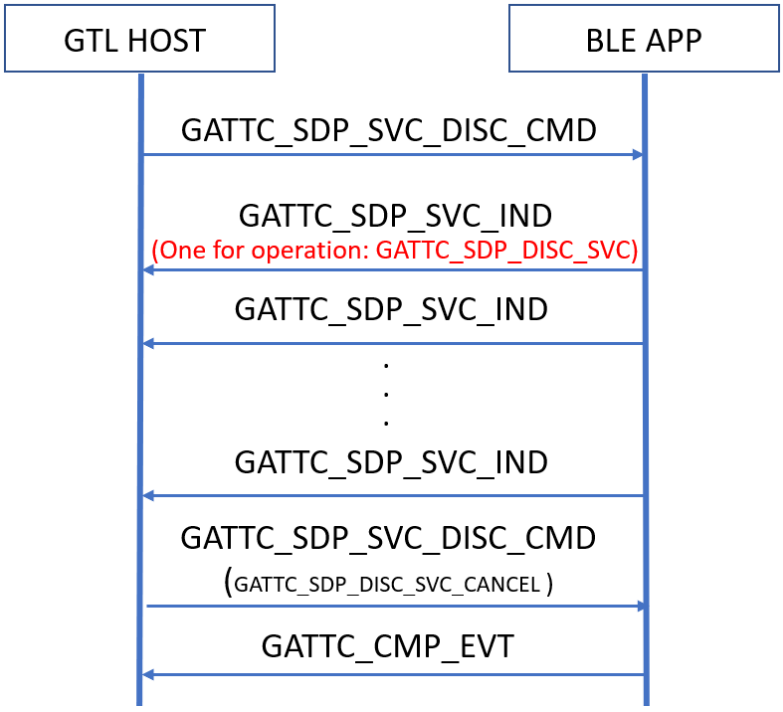


Figure 23. Issuing a GATTC_SDP_SVC_DISC_CMD with an operation of GATTC_SDP_DISC_SVC_CANCEL while discovering

Table 143. GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC_CANCEL

Message stream GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC_CANCEL		
Symbolic message structure reference can be found in Table 134.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x19 0x0C	GATTC_SDP_SVC_DISC_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x18 0x00	24 Bytes
operation	0x17	GATTC_SDP_DISC_SVC_CANCEL
uuid_len	0x00	Not used in SVC_ALL GATTC_SDP_DISC_SVC_CANCEL operation
seq_num	0x25 0x00	Sequence Number custom-used by the application
start_hdl	0x01 0x00	0x0001: Handle 1
end_hdl	0xFF 0xFF	0xFFFF: Handle 65535 (search for all possible handles)
uuid	0x00 0x00	Not used in SVC_ALL GATTC_SDP_DISC_SVC_CANCEL operation
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	
Message string		
0x05 0x19 0x0C 0x0C 0x00 0x10 0x00 0x18 0x00 0x17 0x00 0x25 0x00 0x01 0x00 0xFF 0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

DA14585/531 responds with GATTC_CMP_EVT. In this example, GATTC_CMP_EVT has an operation of GATTC_SDP_DISC_SVC_ALL as the canceling of the procedure GATTC_SDP_DISC_CMD with an operation of GATTC_SDP_DISC_SVC_CANCEL sent during GATTC_SDP_DISC_CMD with an operation of GATTC_SDP_DISC_SVC_ALL. The status of GATTC_CMP_EVT is 0x44 (GAP_ERR_CANCELED).

Table 144. GATTC_CMP_EVT after a GATTC_SDP_SVC_DISC_CMD with operation GATTC_SDP_DISC_SVC_CANCEL

Message stream (GATTC_CMP_EVT) on completion of GATTC_SDP_SVC_DISC_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC
PAR_LEN	0x04 0x00	4 bytes
operation	0x17	GATTC_SDP_DISC_SVC_CANCEL
status	0x44	GAP_ERR_CANCELED
seq_num	0x25 0x00	Sequence number 0x0025
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x16 0x00 0x25 0x00		

5.3.9.10.3 GATTC_READ_CMD

The GTL Host sends GATTC_READ_CMD to read a characteristic value.

There are four different operations that can be executed with GATTC_READ_CMD: GATTC_READ, GATTC_READ_LONG, GATTC_READ_BY_UUID, GATTC_READ_MULTIPLE.

All operations of GATTC_READ_CMD follow the procedure shown in Figure 24.

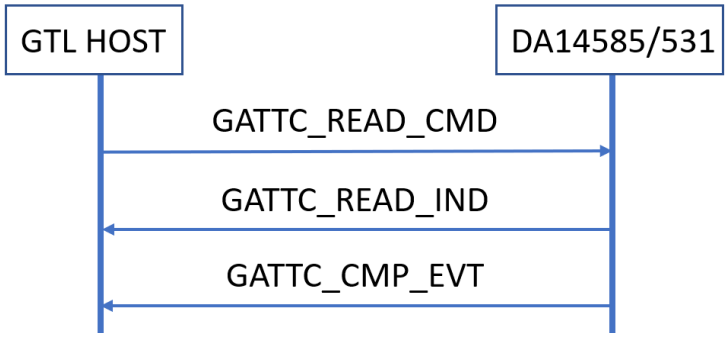


Figure 24. Read characteristic value

Table 145. GATTC_READ_CMD symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_READ_CMD	0x0C08
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Depends on the operation	0xXXXX
Message parameters			
operation	1	GATTC_READ	0x08
		GATTC_READ_LONG	0x09
		GATTC_READ_BY_UUID	0x0A
		GATTC_READ_MULTIPLE	0x0B
nb	1	Number of read (only used for multiple read)	0xXX
seq_num	2	Operation sequence number	0xXXXX
GATTC_READ or GATTC_READ_LONG			
simple.handle	2	Attribute handle	0xXXXX
simple.offset	2	Start offset in data payload	0xXXXX
simple.length	2	Length of data to read (0 = read all)	0xXXXX
GATTC_READ_BY_UUID			
by_uuid.start_hdl	2	Start handle	0xXXXX
by_uuid.end_hdl	2	End handle	0xXXXX
by_uuid.uuid_len	1	**0x02** (16-bit UUID or operations not requiring uuid)	2 bytes
		0x10 (128-bit UUID)	16 bytes
by_uuid.uuid	2 (16-bit UUID)	0xXXXX (16-bit UUID)	
	16 (128-bit UUID)	0xXXXXXXXXXXXXXXXX (128-bit UUID)	

Parameters	Bytes	Description	Hex data
Header			
GATTC_READ_MULTIPLE			
multiple.handle	2	Attribute handle	
multiple.len	2	Known Handle length (must not be 0)	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

GATTC_READ_CMD with Operation GATTC_READ

This subprocedure is used to read a Characteristic Value from a server when the client knows the Characteristic Value Handle. The Attribute Protocol Read Request is used with the Attribute Handle parameter set to the Characteristic Value Handle. The Read Response returns the Characteristic Value in the Attribute Value parameter (see 4.8.1, Part G, Vol 3 in Ref. [1]).

1. Step 1: The GTL Host sends GATTC_READ_CMD (with the GATTC_READ operation) to the Bluetooth LE application to read a simple characteristic from a peer device.

Table 146. GATTC_READ_CMD with operation GATTC_READ – example

Message stream GATTC_READ_CMD with operation GATTC_READ		
Symbolic message structure reference can be found in Table 145.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x08 0x0C	GATTC_READ_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	0x000A = 10 bytes
operation	0x08	GATTC_READ
nb	0x00	Not used - Only applicable for GATTC_READ_MULTIPLE operation
seq_num	0x0B 0x00	Operation sequence number
simple.handle	0x04 0x00	Attribute handle
simple.offset	0x00 0x00	Start offset in data payload
simple.length	0x00 0x00	Length of data to read (0 = read all)
Message string		
0x05 0x08 0x0C 0x0C 0x00 0x10 0x00 0x0A 0x00 0x08 0x00 0x0B 0x00 0x04 0x00 0x00 0x00 0x00 0x00		

2. Step 2: The Bluetooth LE application sends **GATTC_READ_IND** for the attribute that has been read.

Table 147. GATTC_READ_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_READ_IND	0x0C09
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC	0xnn0C (nn = conidx) (Note 1)
PAR_LEN	2	Length of the parameters	0xXXXX (depending on the value)

Parameters	Bytes	Description	Data
Message parameter			
handle	2	Attribute Handle	0xXXXX
offset	2	Read Offset	0xXXXX
length	2	Read Length	0xXXXX
value	{length}	The value read	0xXXXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 148. GATTC_READ_IND after GATTC_READ_CMD (GATTC_READ)

Message stream (GATTC_READ_IND)		
Symbolic message structure reference can be found in Table 147.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x09 0x0C	GATTC_READ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x0B 0x00	11 bytes
handle	0x04 0x00	Read Handle: 0x0004
offset	0x00 0x00	Offset = 0
length	0x05 0x00	Length = 5 bytes
value	0x02 0x05 0x00 0x01 0x2A	
Message string		
0x05 0x09 0x0C 0x10 0x00 0x0C 0x00 0x0B 0x00 0x04 0x00 0x00 0x00 0x05 0x00 0x02 0x05 0x00 0x01 0x2A		

3. Step 3: After **GATTC_READ_IND**, the Bluetooth LE application then sends **GATTC_CMP_EVT** to the **GTL Host**.

Table 149. GATTC_CMP_EVT after GATTC_READ_CMD with operation GATTC_READ

Message stream (GATTC_CMP_EVT) on completion of GATTC_READ_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x08	GATTC_READ
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x0B 0x00	Sequence number 0x000B
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x08 0x00 0x0B 0x00		

GATTC_READ_CMD with Operation GATTC_READ_LONG

This subprocedure is used to read a Characteristic Value from a server when the client knows the Characteristic Value Handle and the length of the Characteristic Value is longer than can be sent in a single Read Response Attribute Protocol message (see Section 4.8.3 in Ref. [1]).

1. Step 1: The GTL Host sends GATTC_READ_CMD (with the GATTC_READ_LONG operation) to the Bluetooth LE application to read a long characteristic from a peer device.

Table 150. GATTC_READ_CMD with operation GATTC_READ_LONG – example

Message stream GATTC_READ_CMD with operation GATTC_READ_LONG		
Symbolic message structure reference can be found in Table 145.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x08 0x0C	GATTC_READ_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	0x000A = 10 bytes
operation	0x09	GATTC_READ_LONG
nb	0x00	Only applicable for GATTC_READ_MULTIPLE operation
seq_num	0x21 0x00	Operation sequence number
simple.handle	0x03 0x00	Attribute handle
simple.offset	0x00 0x00	Start offset in data payload
simple.length	0x00 0x00	Length of data to read
Message string		
0x05 0x08 0x0C 0x0C 0x00 0x10 0x00 0x0A 0x00 0x09 0x00 0x21 0x00 0x03 0x00 0x00 0x00 0x00 0x00		

2. Step 2: Bluetooth LE application sends GATTC_READ_IND for the attribute that has been read.

Table 151. GATTC_READ_IND after GATTC_READ_CMD (GATTC_READ_LONG)

Message stream (GATTC_READ_IND)		
Symbolic message structure reference can be found in Table 147.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x09 0x0C	GATTC_READ_IND
DST_ID	0x10 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x0C 0x00	TASK_ID_GTL
PAR_LEN	0x18 0x01	280 bytes
handle	0x03 0x00	Read Handle: 0x0003
offset	0x00 0x00	Offset = 0
length	0x12 0x01	274 bytes
value	0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x44 0x69 0x61 0x6C 0x6F 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x20 0x48 0x6F 0x47 0x50 0x20 0x44 0x65 0x76 0x69 0x63 0x65	"VeryLongVeryLongVeryLongVeryLongVeryLongVeryLongVeryLongDialogVeryLong HoGP Device"
Message string		
0x05 0x09 0x0C 0x10 0x00 0x0C 0x00 0x18 0x01 0x03 0x00 0x00 0x00 0x12 0x01 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x44 0x69 0x61 0x6C 0x6F 0x67 0x56 0x65 0x72 0x79 0x4C 0x6F 0x6E 0x67 0x20 0x48 0x6F 0x47 0x50 0x20 0x44 0x65 0x76 0x69 0x63 0x65		

3. Step 3: After **GATTC_READ_IND,** the Bluetooth LE application then sends **GATTC_CMP_EVT** to the GTL Host.

Table 152. GATTC_CMP_EVT after a GATTC_READ_CMD with operation GATTC_READ

Message stream (GATTC_CMP_EVT) on completion of GATTC_READ_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x09	GATTC_READ_LONG
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x21 0x00	Sequence number 0x0021
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x09 0x00 0x21 0x00		

GATTC_READ_CMD with Operation GATTC_READ_BY_UUID – Example

This subprocedure is used to read a Characteristic Value from a server when the client only knows the characteristic UUID and does not know the handle of the characteristic (see Section 4.8.2 in Ref. [1]).

1. Step 1: GTL Host sends GATTC_READ_CMD (with the GATTC_READ_BY_UUID operation) to the Bluetooth LE application to read a characteristic from a peer device with a specific UUID.

Table 153. GATTC_READ_CMD with operation GATTC_READ_BY_UUID – example

Message stream - GATTC_READ_CMD with operation GATTC_READ_BY_UUID		
Symbolic message structure reference can be found in Table 145.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x08 0x0C	GATTC_READ_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0C 0x00	0x000C = 12 bytes
operation	0x0A	GATTC_READ_BY_UUID
nb	0x00	Only applicable for GATTC_READ_MULTIPLE operation
seq_num	0x1B 0x00	Operation sequence number – Used by the application
by_uuid.start_hdl	0x01 0x00	Start handle
by_uuid.end_hdl	0xFF 0xFF	End handle
by_uuid.uuid_len	0x02 0x00	Size of UUID
by_uuid.uuid	0x2A 0x00	Specific UUID
Message string		
0x0A 0x00 0x1B 0x00 0x01 0x00 0xFF 0xFF 0x02 0x00 0x2A 0x00		

2. Step 2: The Bluetooth LE application sends GATTC_READ_IND for the attribute that has been read.

Table 154. GATTC_READ_IND after GATTC_READ_CMD (GATTC_READ_BY_UUID)

Message stream (GATTC_READ_IND)		
Symbolic message structure reference can be found in Table 147		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x09 0x0C	GATTC_READ_IND
DST_ID	0x10 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x0C 0x00	TASK_ID_GTL
PAR_LEN	0x10 0x00	16 bytes
handle	0x03 0x00	Read Handle: 0x0003
offset	0x00 0x00	Offset = 0
length	0x0A 0x00	10 bytes
value	0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x42 0x4C 0x45	"Renesas Bluetooth LE"
Message string		
0x05 0x09 0x0C 0x10 0x00 0x0C 0x00 0x10 0x00 0x03 0x00 0x00 0x00 0x0A 0x00 0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x42 0x4C 0x45		

3. Step 3: After **GATTC_READ_IND**, the Bluetooth LE application then sends **GATTC_CMP_EVT** to the GTL Host.

Table 155. GATTC_CMP_EVT after GATTC_READ_CMD with operation GATTC_READ

Message stream (GATTC_CMP_EVT) on completion of GATTC_READ_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x0A	GATTC_READ_BY_UUID
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x1B 0x00	Sequence number 0x001B
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x0A 0x00 0x1B 0x00		

GATTC_READ_CMD with Operation GATTC_READ_MULTIPLE – Example

This subprocedure is used to read multiple Characteristic Values from a server when the client knows the Characteristic Value Handles. The Attribute Protocol Read Multiple Requests is used with the Set of Handles parameter set to the Characteristic Value Handles. The Read Multiple Response returns the Characteristic Values in the Set of Values parameter. (see Section 4.8.4 in Ref. [1]).

1. Step 1: The GTL Host sends GATTC_READ_CMD (with the GATTC_READ_MULTIPLE operation) to the Bluetooth LE application to read multiple characteristics' values from the peer device.

Table 156. GATTC_READ_CMD with operation GATTC_READ_MULTIPLE – example

Message stream GATTC_READ_CMD with operation GATTC_READ_BY_UUID		
Symbolic message structure reference can be found in Table 145.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x08 0x0C	GATTC_READ_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0C 0x00	0x000C = 12 bytes
operation	0x0B	GATTC_READ_MULTIPLE
nb	0x02	No of Handles = 2
seq_num	0x16 0x00	Operation sequence number
multiple.handle	0x04 0x00 0x05 0x00	Attribute handles: 0x0004 and 0x0005
multiple.len	0x05 0x00 0x02 0x00	Attribute Handle lengths: 5 bytes and 2 bytes respectively
Message string		
0x05 0x08 0x0C 0x0C 0x00 0x10 0x00 0x0C 0x00 0x0B 0x02 0x16 0x00 0x04 0x00 0x05 0x00 0x05 0x00 0x02 0x00		

2. Step 2: The Bluetooth LE application sends GATTC_READ_INDs for the attributes that have been read.

Table 157. GATTC_READ_IND #1 after a GATTC_READ_CMD (GATTC_READ_MULTIPLE)

Message stream (GATTC_READ_IND)		
Symbolic message structure reference can be found in Table 147		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x09 0x0C	GATTC_READ_IND
DST_ID	0x10 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x0C 0x0	TASK_ID_GTL
PAR_LEN	0x0B 0x00	0x000B 11 bytes
handle	0x04 0x00	Handle = 4
offset	0x00 0x00	Offset = 0
length	0x05 0x00	Length = 5 bytes
value	0x02 0x05 0x00 0x01 0x2A	Value
Message string		
0x05 0x09 0x0C 0x10 0x00 0x0C 0x00 0x0B 0x00 0x04 0x00 0x00 0x00 0x05 0x00 0x02 0x05 0x00 0x01 0x2A		

Table 158. GATTC_READ_IND #2 after a GATTC_READ_CMD (GATTC_READ_MULTIPLE)

Message stream (GATTC_READ_IND)		
Symbolic message structure reference can be found in Table 147		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x09 0x0C	GATTC_READ_IND
DST_ID	0x10 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x0C 0x0	TASK_ID_GTL
PAR_LEN	0x08 0x00	8 bytes
handle	0x05 0x00	Handle = 5
offset	0x00 0x00	Offset = 0
length	0x02 0x00	Length = 2 bytes
value	0xC0 0x03	Value
Message string		
0x05 0x09 0x0C 0x10 0x00 0x0C 0x00 0x08 0x00 0x05 0x00 0x00 0x00 0x02 0x00 0xC0 0x03		

3. Step 3: After **GATTC_READ_IND**, the Bluetooth LE application then sends **GATTC_CMP_EVT** to the GTL Host.

Table 159. GATTC_CMP_EVT after GATTC_READ_CMD with operation GATTC_READ

Message stream (GATTC_CMP_EVT) on completion of GATTC_READ_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x0B	GATTC_READ_MULTIPLE
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x16 0x00	Sequence number 0x0016
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x0B 0x00 0x16 0x00		

5.3.9.10.4 GATTC_WRITE_CMD

This procedure is used to write a Characteristic Value to a server. The GTL Host sends GATTC_WRITE_CMD to the Bluetooth LE application.

Table 160. GATTC_WRITE_CMD symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_WRITE_CMD	0x0C0A

Parameters	Bytes	Description	Hex data
Header			
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Depends on the operation	0xXXXX
Message parameters			
operation	1	GATTC_WRITE	0x0C
		GATTC_WRITE_NO_RESPONSE	0x0D
		GATTC_WRITE_SIGNED	0x0E
auto_execute	1	Perform automatic execution (if false, an ATT Prepare Write is used, this must be used for reliable write)	0x00 or 0x01
seq_num	2	Operation sequence number	
handle	2	Attribute handle	
offset	2	Write offset	
length	2	Length of data to write	
cursor	2	Internal write cursor must be initialized to 0	
value	XXXX	Value to write	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

GATTC_WRITE_CMD with Operation GATTC_WRITE

This subprocedure is used to write a Characteristic Value to a server when the client knows the Characteristic Value Handle. A Write Response is sent by the server if the write of the Characteristic Value succeeded, whereas an Error Response is sent by the server in response to the Write Request if something goes wrong (see Section 4.9.3 in Ref. [1]).

1. Step 1: The GTL Host sends GATTC_WRITE_CMD (with the GATTC_WRITE operation) to DA14585/531 to write to the characteristic's value handle.

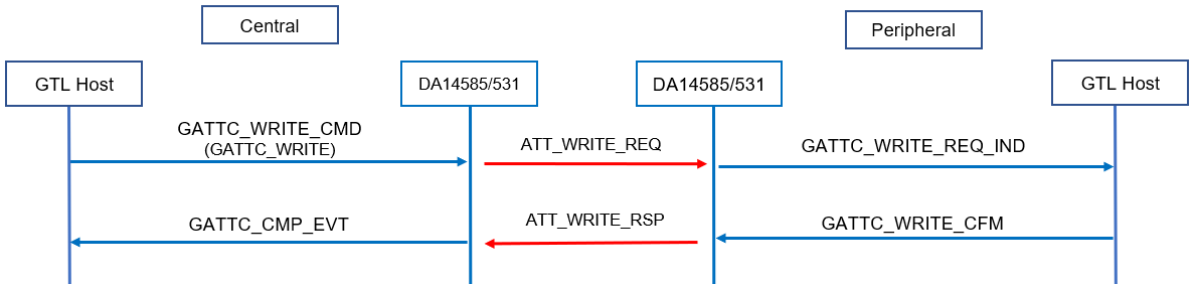


Figure 25. Write value to an attribute (GATTC_WRITE_CMD with operation GATTC_WRITE)

Table 161. GATTC_WRITE_CMD with operation GATTC_WRITE – example

Message stream GATTC_WRITE_CMD with operation GATTC_WRITE		
Symbolic message structure reference can be found in Table 160.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0A 0x0C	GATTC_WRITE_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0E 0x00	14 bytes
operation	0x0C	GATTC_WRITE
auto_execute	0x01	If false, an ATT Prepare Write is used
seq_num	0x09 0x00	Sequence number 0x0009
handle	0x03 0x00	Handle of the attribute
offset	0x00 0x00	Write offset
length	0x01 0x00	1 byte
cursor	0x00 0x00	Internal write cursor (must be initialized to 0)
value	0x01	Value
{padding}	0x00	Ignored, can consist of any value
Message string		
0x05 0x0A 0x0C 0x0C 0x00 0x10 0x00 0x0E 0x00 0x0C 0x01 0x09 0x00 0x03 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x01 0x00		

2. Step 2: The Bluetooth LE application sends GATTC_CMP_EVT when the procedure is completed.

Table 162. GATTC_CMP_EVT after GATTC_WRITE_CMD with operation GATTC_WRITE

Message stream (GATTC_CMP_EVT) on completion of GATTC_READ_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x0C	GATTC_WRITE
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x09 0x00	Sequence number 0x0009
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x0C 0x00 0x09 0x00		

GATTC_WRITE_CMD with Operation GATTC_WRITE_NO_RESPONSE

This subprocedure is used to write a Characteristic Value to a server when the client knows the Characteristic Value Handle and the client does not need an acknowledgment that the write was successfully performed. This subprocedure only writes the first (ATT_MTU – 3) octets of a Characteristic Value (see Section 4.9.1 in Ref. [1]).

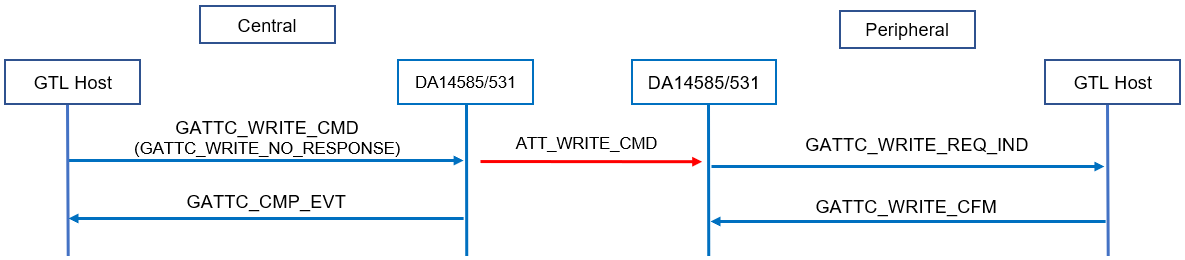


Figure 26. Write no response

1. Step 1: The GTL Host sends GATTC_WRITE_CMD (with the GATTC_WRITE_NO_RESPONSE operation) to the Bluetooth LE application to write to the characteristic's value handle without any confirmation from the peer.

Table 163. GATTC_WRITE_CMD with operation GATTC_WRITE_NO_RESPONSE – example

Message stream GATTC_WRITE_CMD with operation GATTC_WRITE_NO_RESPONSE		
Symbolic message structure reference can be found in Table 160.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0A 0x0C	GATTC_WRITE_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x10 0x00	16 bytes
operation	0x0D	GATTC_WRITE_NO_RESPONSE
auto_execute	0x01	If false, an ATT Prepare Write is used
seq_num	0x0D 0x00	Sequence number 0x000D
handle	0x1D 0x00	Handle of the attribute
offset	0x00 0x00	Write offset
length	0x04 0x00	4 bytes
cursor	0x00 0x00	Internal write cursor (must be initialized to 0)
value	0x74 0x65 0x73 0x74	Value
Message string		
0x05 0x0A 0x0C 0x0C 0x00 0x10 0x00 0x10 0x00 0x0D 0x01 0x0D 0x00 0x1D 0x00 0x00 0x00 0x04 0x00 0x00 0x00 0x74 0x65 0x73 0x74		

2. Step 2: The Bluetooth LE application sends GATTC_CMP_EVT to the GTL Host when the procedure is completed.

Table 164. GATTC_CMP_EVT after GATTC_WRITE_CMD with operation GATTC_WRITE_NO_RESPONSE

Message stream (GATTC_CMP_EVT) on completion of GATTC_READ_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x0D	GATTC_WRITE_NO_RESPONSE
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x0D 0x00	Sequence number 0x000D
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x0D 0x00 0x0D 0x00		

GATTC_WRITE_CMD with Operation GATTC_WRITE_SIGNED

This subprocedure is used to write a Characteristic Value to a server when the client knows the Characteristic Value Handle and the ATT Bearer is not encrypted. This subprocedure must only be used if the Characteristic Properties authenticated bit is enabled and the client and server device share a bond as defined in Vol 3, Part C, Section 4.9.2 of Ref. [1].

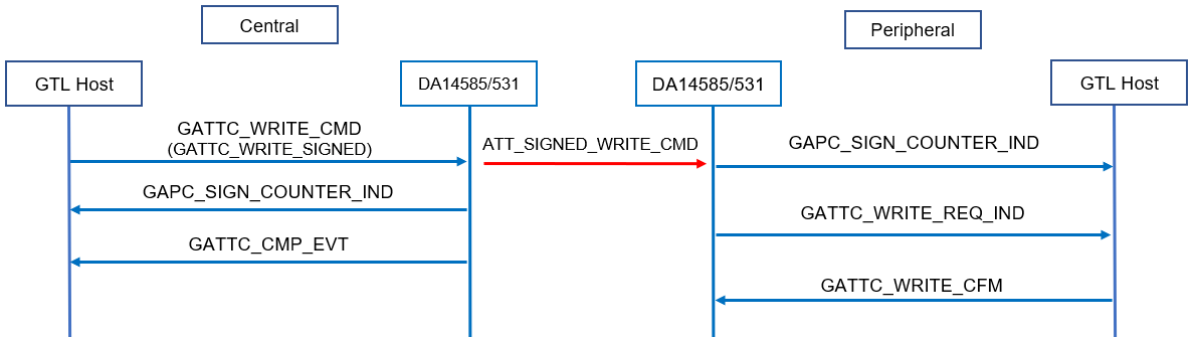


Figure 27. Write signed procedure

1. Step 1: The GTL Host sends GATTC_WRITE_CMD (with the GATTC_WRITE_SIGNED operation) to the Bluetooth LE application to write to the characteristic's value.

Table 165. GATTC_WRITE_CMD with operation GATTC_WRITE_SIGNED – example

Message stream GATTC_WRITE_CMD with operation GATTC_WRITE_SIGNED		
Symbolic message structure reference can be found in Table 160.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0A 0x0C	GATTC_WRITE_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0E 0x00	14 bytes
operation	0x0E	GATTC_WRITE_SIGNED
auto_execute	0x01	If false, an ATT Prepare Write is used
seq_num	0x10 0x00	Sequence number 0x0010
handle	0xAB 0x00	Handle of the attribute
offset	0x00 0x00	Write offset
length	0x01 0x00	2 bytes
cursor	0x00 0x00	Internal write cursor (must be initialized to 0)
value	0x0B 0x00	Value
Message string		
0x05 0x0A 0x0C 0x0C 0x00 0x10 0x00 0x0E 0x00 0x0E 0x01 0x10 0x00 0xAB 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x0B 0x00		

2. Step 2: DA14585/531 responds with **GAPC_SIGN_COUNTER_IND** to the GTL Host.

Table 166. GAPC_SIGN_COUNTER_IND #1 example – on the central's side

Message stream GAPC_SIGN_COUNTER_IND		
Symbolic message structure reference can be found in Table 216		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1C 0x0E	GAPC_SIGN_COUNTER_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x08 0x00	8 bytes
local_sign_counter	0x02 0x00 0x00 0x00	gapc_get_sign_counter (SMPC_INFO_LOCAL)
peer_sign_counter	0x00 0x00 0x00 0x00	gapc_get_sign_counter (SMPC_INFO_PEER)
Message string		
0x05 0x1C 0x0E 0x10 0x00 0x0E 0x00 0x08 0x00 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

Table 167. GAPC_SIGN_COUNTER_IND #2 example – on the peripheral's side

Message stream GAPC_SIGN_COUNTER_IND		
Symbolic message structure reference can be found in Table 216		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1C 0x0E	GAPC_SIGN_COUNTER_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x08 0x00	8 bytes
local_sign_counter	0x00 0x00 0x00 0x00	gapc_get_sign_counter (SMPC_INFO_LOCAL)
peer_sign_counter	0x02 0x00 0x00 0x00	gapc_get_sign_counter (SMPC_INFO_PEER)
Message string		
0x05 0x1C 0x0E 0x10 0x00 0x0E 0x00 0x08 0x00 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

The value of the local_sign_counter field in Table 166 and peer_sign_counter field in Table 167 are the same. The local value and the peer value of the sign counter are checked by the Bluetooth LE stack to prevent replay attacks. (See Section 2.4.5, in Part H, Vol 3 in Ref. [1])

3. Step 3: The Bluetooth LE application sends GATTC_CMP_EVT to the GTL Host when the procedure is completed.

Table 168. GATTC_CMP_EVT after GATTC_WRITE_CMD with operation GATTC_WRITE_SIGNED

Message stream (GATTC_CMP_EVT) on completion of GATTC_READ_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x0E	GATTC_WRITE_SIGNED
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x10 0x00	Sequence number 0x0010
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x0E 0x00 0x10 0x00		

5.3.9.10.5 GATTC_EXECUTE_WRITE_CMD

The execute write command is used to request the server to write or cancel the write of all the prepared values held in the prepare queue from a client. This request must be handled by the server as an atomic operation (Vol 3, Part F, Section 3.4.6 in Ref. [1]).

Table 169. GATTC_WRITE_EXECUTE_CMD symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05

Parameters	Bytes	Description	Hex data
Header			
MSG_ID	2	GATTC_WRITE_EXECUTE_CMD	0x0C0B
DST_ID	2	TASK_ID_GATTC	0x000C
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	4 bytes	0x0004
Message parameters			
operation	1	GATTC_EXEC_WRITE	0x0F
execute	1	**0x00**: Cancel pending write operations **0x01**: Perform write operations	0x00 or 0x01
seq_num	2	Operation sequence number	0xXXXX

Table 170. GATTC_WRITE_EXECUTE_CMD example

Message stream GATTC_WRITE_EXECUTE_CMD		
Symbolic message structure reference can be found in Table 169.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0B 0x0C	GATTC_WRITE_EXECUTE_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x04 0x00	0x0004
operation	0x0F	GATTC_EXEC_WRITE
execute	0x01	TRUE = Perform queued operations
seq_num	0x20 0x00	Sequence number 0x0020. Used by the application
Message string		
0x05 0x0B 0x0C 0x0C 0x00 0x10 0x00 0x04 0x00 0x0F 0x01 0x20 0x00		

The GATTC_WRITE_EXECUTE_CMD Example

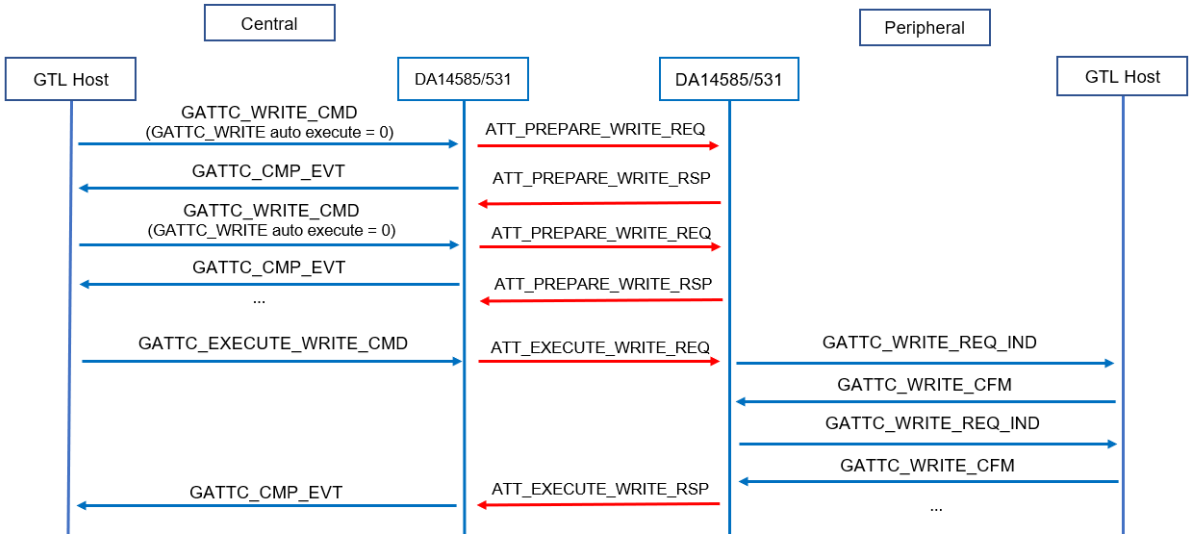
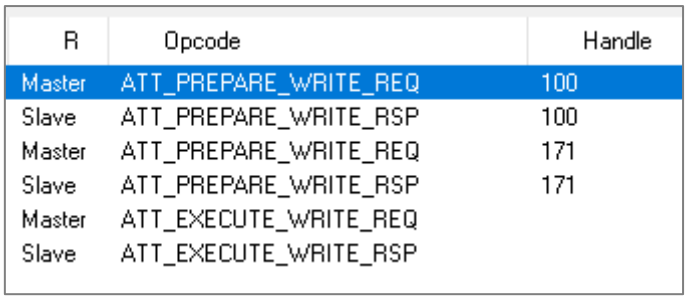


Figure 28. The GATTC_WRITE_EXECUTE_CMD example



R	Opcode	Handle
Master	ATT_PREPARE_WRITE_REQ	100
Slave	ATT_PREPARE_WRITE_RSP	100
Master	ATT_PREPARE_WRITE_REQ	171
Slave	ATT_PREPARE_WRITE_RSP	171
Master	ATT_EXECUTE_WRITE_REQ	
Slave	ATT_EXECUTE_WRITE_RSP	

Figure 29. The GATTC_WRITE_EXECUTE_CMD example - what happens on the air

The GTL Host on the central's side sends two GATTC_WRITE_CMDs with the auto_execute field set to 0x00 (see Table 160) to DA14581/531 on the central's side. This creates ATT_PREPARE_WRITE_REQs that are queued on the peripheral's side. When the GTL Host on the central's side performs GATTC_EXECUTE_WRITE_CMD, DA145845/531 on the peripheral's side sends an equal number of GATTC_WRITE_REQ_INDs (see Section GATTC_WRITE_REQ_IND Indication and GATTC_WRITE_CFM Response) to the GTL Host on the peripheral's side as ATT_PREPARE_WRITE_REQs queued. When the final GATTC_WRITE_CFM arrives to DA14585/531 from the GTL Host on the peripheral's side, ATT_EXECUTE_WRITE_RSP is sent to DA14585/531 on the central's side that in turn sends GATTC_CMP_EVT to the corresponding GTL Host.

5.3.10 Controller's Resolving List

The DA14585/531 maintains a Link Layer Resolving List where a set of records of local and peer IRK value pairs is maintained.

5.3.10.1 Adding a Device to Controller's Resolving List

1. Step 1: The GTL Host sends the request to add a device to the controller's resolving list.

   The GTL Host sends GAPM_RAL_MGT_CMD (Table 171) to add a peer device information to the controller's resolving list.

Table 171. GAPM_RAL_MGT_CMD symbolic message structure

Parameters		Bytes	Description	Data
Header				
Initiator		1	GTL Initiator	0x05
MSG_ID		2	GAPM_RAL_MGT_CMD	0x0D20
DST_ID		2	TASK_ID_GAPM	0x000D
SRC_ID		2	TASK_ID_GTL	0x0010
PAR_LEN		2	nb × 39 + 2, where nb is listed in Message Parameters in this table	0x0029 (41 bytes for adding 1 device)
Message parameters				
operation		1	GAPM_ADD_DEV_IN_RAL	0x24
nb		1	Number of device information present in the command	0x01 (for one device)
devices [0]	addr_type	1	The Identity Address type of the first device to add	**0x00**: ADDR_PUBLIC, Public Identity Address **0x01**: ADDR_PRIVATE, Random (static) Identity Address
	addr	6	The Identity Address of the first device to add	
	peer_irk	16	The IRK of the peer device for the first device to add	

Parameters		Bytes	Description	Data
	local_irk	16	The IRK of the local device for the first device to add	
devices [1]		1 + 6 + 16 + 16 = 39	See devices [0] in this table	
(…)	See devices [0] in this table			
devices[nb-1]	See devices [0] in this table.			

Table 172. GAPM_RAL_MGT_CMD for GAPM_ADD_DEV_IN_RAL operation - example

Message stream (for adding one device to the resolving list)		
Symbolic message structure reference can be found in Table 171.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x20 0x0D	GAPM_RAL_MGT_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x29 0x00	41 bytes
operation	0x24	GAPM_ADD_DEV_IN_RAL
nb	0x01	One device is added
devices	0x00 0x84 0x2B 0x93 0x82 0xF8 0x78 0x8a 0x57 0x28 0x35 0xb0 0x91 0xd4 0xf0 0x65 0x01 0x94 0x38 0xca 0x03 0x03 0x67 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	▪ BD address: 0x78F882932B84 ▪ BD Address Type: Public (0x00) ▪ Peer IRK: 0x670303ca38940165f0d491b03528578a ▪ Local IRK: 0x00000000000000000000000000000000
Message string		
0x05 0x20 0x0D 0x0D 0x00 0x10 0x00 0x29 0x00 0x24 0x01 0x00 0x84 0x2B 0x93 0x82 0xF8 0x78 0x8a 0x57 0x28 0x35 0xB0 0x91 0xd4 0xf0 0x65 0x01 0x94 0x38 0xca 0x03 0x03 0x67 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

2. Step 2: Reception of the completion event.

   DA14585/531 GAPM sends a completion event message (GAPM_CMP_EVT) for the operation GAPM_RAL_MGT_CMD.

Table 173. GAPM_CMP_EVT for GAPM_ADD_DEV_IN_RAL operation - example

Message stream: GAPM_CMP_EVT for GAPM_ADD_DEV_IN_RAL operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x24	GAPM_ADD_DEV_IN_RAL
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x24 0x00		

5.3.11 Whitelisting

The GAPM_WHITE_LIST_MGT_CMD command is used to manage LE Low Layer Whitelist. Whitelist can be modified through automatic or selective connections operations. This message API should be used only for advertising or scanning air operations.

Table 174. GAPM_WHITE_LIST_MGT_CMD - symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_WHITE_LIST_MGT_CMD	0x0D0B
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	9 bytes	0x0009
Message parameters			
operation	1	GAPM_GET_WLIST_SIZE	0x08
		GAPM_ADD_DEV_IN_WLIST	0x09
		GAPM_RMV_DEV_FRM_WLIST	0x0A
		GAPM_CLEAR_WLIST	0x0B
nb	1	Number of device information present in the command	0x01 (for adding 1 device)
devices	{nb} × 7	Device address information that can be used to add or remove element in device list. The identity address (6 byte) and address type (1 byte) of each device to add.	BD Address of device (identity address), Address type of the device 0 = public/1 = private random

5.3.11.1 Getting the Whitelist Size

1. Step 1: The GTL Host sends GAPM_WHITE_LIST_MGT_CMD (with an operation of GAPM_GET_WLIST_SIZE) to DA14585/531 to get the whitelist size.

Table 175. GAPM_WHITE_LIST_MGT_CMD with GAPM_GET_WLIST_SIZE operation - example

Message stream (get whitelist size)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0B 0x0D	GAPM_WHITE_LIST_MGT_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0X00	2 bytes
operation	0x08	GAPM_GET_WLIST_SIZE
nb	0x00	Ignored for this example
Message string		
0x05 0x0B 0x0D 0x0D 0x00 0x10 0x00 0x02 0x00 0x08 0x00		

2. Step 2: DA14585/531 responds to the GTL Host with GAPM_WHITE_LIST_SIZE_IND.

Table 176. GAPM_WHITE_LIST_SIZE_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_WHITE_LIST_SIZE_IND	0x0D0C
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	2 bytes	0x0001
Message parameters			
size	1	White List size	

Table 177. GAPM_WHITE_LIST_SIZE_IND after GAPM_WHITE_LIST_MGT_CMD with GAPM_GET_WLIST_SIZE operation

Message stream: GAPM_WHITE_LIST_SIZE_IND		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0C 0x0D	GAPM_WHITE_LIST_SIZE_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x01 0x00	1 byte
size	0x0A	White List size = 10
Message string		
0x05 0x0C 0x0D 0x10 0x00 0x0D 0x00 0x01 0x00 0x0A		

3. Step 3: DA14585/531 sends a completion event message (GAPM_CMP_EVT) for the operation GAPM_GET_WLIST_SIZE

Table 178. GAPM_CMP_EVT for GAPM_GET_WLIST_SIZE - example

Message stream: GAPM_CMP_EVT of GAPM_GET_WLIST_SIZE operation		
Symbolic message structure reference can be found in Table 51		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x08	GAPM_GET_WLIST_SIZE
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x08 0x00		

5.3.11.2 Adding a Device to Whitelist

1. Step 1: The GTL host sends GAPM_WHITE_LIST_MGT_CMD (with an operation of GAPM_ADD_DEV_IN_WLIST) to DA14585/531 to add a device to the whitelist.

Table 179. GAPM_WHITE_LIST_MGT_CMD with GAPM_ADD_DEV_IN_WLIST operation

Message stream (add a device to the whitelist)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0B 0x0D	GAPM_WHITE_LIST_MGT_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x1E 0x00	30 bytes
operation	0x09	GAPM_ADD_DEV_IN_WLIST
nb	0x04	Four devices are added
devices [0]	0x13 0x00 0x70 0xCA 0xEA 0x80 0x01	▪ BD address: 0x80EACA700013 ▪ BD Address Type: Private (0x01)
devices [1]	0x08 0x00 0x70 0xCA 0xEA 0x80 0x00	▪ BD address: 0x80EACA700008 ▪ BD Address Type: Public (0x00)
devices [2]	0x09 0x00 0x70 0xCA 0xEA 0x80 0x00	▪ BD address: 0x80EACA700009 ▪ BD Address Type: Public (0x00)
devices [3]	0x10 0x00 0x70 0xCA 0xEA 0x80 0x00	▪ BD address: 0x80EACA700010 ▪ BD Address Type: Public (0x00)
Message string		
0x05 0x0B 0x0D 0x0D 0x00 0x10 0x00 0x1E 0x00 0x09 0x04 0x13 0x00 0x70 0xCA 0xEA 0x80 0x01 0x08 0x00 0x70 0xCA 0xEA 0x80 0x00 0x09 0x00 0x70 0xCA 0xEA 0x80 0x00 0x10 0x00 0x70 0xCA 0xEA 0x80 0x00		

2. Step 2: DA14585/531 sends a completion event message (GAPM_CMP_EVT) with an operation of GAPM_ADD_DEV_IN_WLIST to the GTL Host.

Table 180. GAPM_CMP_EVT for GAPM_ADD_DEV_IN_WLIST operation – example

Message stream: GAPM_CMP_EVT of GAPM_ADD_DEV_IN_WLIST operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x09	GAPM_ADD_DEV_IN_WLIST
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x09 0x00		

5.3.11.3 Removing a Device from Whitelist

1. Step 1: The GTL Host sends GAPM_WHITE_LIST_MGT_CMD with an operation of GAPM_RMV_DEV_FRM_WLIST to remove a device from the whitelist.

Table 181. GAPM_WHITE_LIST_MGT_CMD with GAPM_RMV_DEV_FRM_WLIST operation - example

Message stream (remove a device from whitelist)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0B 0x0D	GAPM_WHITE_LIST_MGT_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x10 0X00	16 bytes
operation	0x0A	GAPM_RMV_DEV_FRM_WLIST
nb	0x02	One device is removed
devices [0]	0x13 0x00 0x70 0xCA 0xEA 0x80 0x01	▪ BD address: 0x80EACA700013 ▪ BD Address Type: Private (0x01)
devices [1]	0x08 0x00 0x70 0xCA 0xEA 0x80 0x00	▪ BD address: 0x80EACA700008 ▪ BD Address Type: Public (0x00)
Message string		
0x05 0x0B 0x0D 0x0D 0x00 0x10 0x00 0x10 0x00 0x0A 0x02 0x13 0x00 0x70 0xCA 0xEA 0x80 0x01 0x08 0x00 0x70 0xCA 0xEA 0x80 0x00		

2. Step 2: The DA14585/531 sends a completion event message (GAPM_CMP_EVT) with an operation of GAPM_RMV_DEV_FRM_WLIST to the GTL Host.

Table 182. GAPM_CMP_EVT for GAPM_RMV_DEV_FRM_WLIST - example

Message stream: GAPM_CMP_EVT of GAPM_RMV_DEV_FRM_WLIST operation		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x0A	GAPM_RMV_DEV_FRM_WLIST
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x0A 0x00		

5.3.11.4 Clear Whitelist

1. Step 1: External host sends GAPM_WHITE_LIST_MGT_CMD to DA14585/531 to clear the whitelist.

Table 183. GAPM_WHITE_LIST_MGT_CMD with GAPM_CLEAR_WLIST operation - example

Message stream (clear whitelist)		
Symbolic message structure reference can be found in Table 174		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0B 0x0D	GAPM_WHITE_LIST_MGT_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0x00	2 bytes
operation	0x0B	GAPM_CLEAR_WLIST
nb	0x00	Ignored for this example
Message string		
0x05 0x0B 0x0D 0x0D 0x00 0x10 0x00 0x02 0x00 0x0B 0x00		

2. Step 2: The DA14585/531 responds with a completion event message (GAPM_CMP_EVT) for the operation GAPM_CLEAR_WLIST.

Table 184. GAPM_CMP_EVT for GAPM_CLEAR_WLIST

Message stream: GAPM_CMP_EVT with GAPM_CLEAR_WLIST operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x0B	GAPM_CLEAR_WLIST
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x0B 0x00		

5.3.12 Scan Procedure

5.3.12.1 Start Scanning

The **GAPM_START_SCAN_CMD** command is allowed only for devices that support observer or central roles. When information of a peer device is received, an advertising report event is triggered (see Section 5.3.12.2).

Table 185. GAPM_START_SCAN_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_START_SCAN_CMD	0x0D0F
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	12 bytes	0x000C
Message parameters			
op.code	1	GAPM requested operation code	**0x11**: GAPM_SCAN_ACTIVE, start active scan operation **0x12**: GAPM_SCAN_PASSIVE, start passive scan operation
op.addr_src	1	Own BD address source of the device	**0x00**: GAPM_STATIC_ADDR **0x01**: GAPM_GEN_RSLV_ADDR **0x02**: GAPM_GEN_NON_RSLV_ADDR
op.state	2	Dummy data used to retrieve internal operation state (should be set to 0)	0x00 0x00
interval	2	Scan interval N for scanning Value Time = N × 0.625 ms	
window	2	Scan window size N for scanning Value Time = N × 0.625 ms	
mode	1	Scanning mode	**0x00**: GAP_GEN_DISCOVERY, general discovery scanning mode **0x01**: GAP_LIM_DISCOVERY, limited discovery scanning mode **0x02**: GAP_OBSERVER_MODE, observer scanning mode
filt_policy	1	Scan filter policy	**0x00**: SCAN_ALLOW_ADV_ALL, allow advertising packets from anyone **0x01**: SCAN_ALLOW_ADV_WLST, allow advertising packets from whitelist devices only
filter_duplic	1	Scan duplicate filter policy	**0x00**: SCAN_FILT_DUPLIC_DIS, disable filtering of duplicate packets **0x01**: SCAN_FILT_DUPLIC_EN, enable filtering of duplicate packets
{padding}	1		{padding} Ignored. Any value is valid.

Table 186. GAPM_START_SCAN_CMD example

Message stream (GAPM_START_SCAN_CMD)		
Symbolic message structure reference can be found in Table 185		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0F 0x0D	GAPM_START_SCAN_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0C 0x00	12 bytes
op.code	0x12	GAPM_SCAN_PASSIVE
op.addr_src	0x00	GAPM_STATIC_ADDR
op.state	0x00 0x00	dummy data, must be zero
interval	0xA0 0x00	160 × 0.625 = 100 ms
window	0x50 0x00	80 × 0.625 = 50 ms
mode	0x00	GAP_GEN_DISCOVERY
filt_policy	0x00	SCAN_ALLOW_ADV_ALL
filter_duplic	0x00	SCAN_FILT_DUPLIC_DIS
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x0F 0x0D 0x0D 0x00 0x10 0x00 0x0C 0x00 0x12 0x00 0x00 0x00 0xA0 0x00 0x50 0x00 0x00 0x00 0x00 0x00		

5.3.12.2 Reception of Advertising Reports

Several advertising reports (**GAPM_ADV_REPORT_IND**) can be received for a peer device if a scan response is available or if the scan duplicate filter policy is disabled. According to the executed procedure (see parameter mode, filter policy in Section 5.3.12.1) during the scanning, some advertising reports can be filtered by the DA14585/531 GAPM.

Table 187. GAPM_ADV_REPORT_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_ADV_REPORT_IND	0x0D10
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	41 bytes	0x0029

Parameters	Bytes	Description	Data
Header			
Message parameters			
adv_report.evt_type	1	Event type	**0x00**: ADV_CONN_UNDIR, connectable undirected advertising **0x01**: ADV_CONN_DIR, connectable directed advertising **0x02**: ADV_DISC_UNDIR, discoverable undirected advertising **0x03**: ADV_NONCONN_UNDIR, non-connectable undirected advertising **0x04**: ADV_CONN_DIR_LDC, connectable low duty cycle directed advertising
adv_report.adv_addr_type	1	Advertising address type	**0x00**: ADDR_PUBLIC, public BD address **0x01**: ADDR_RAND, random BD Address
adv_report.adv_addr	6	Advertising address value	
adv_report.data_len	1	Data length of advertising packet	
adv_report.data	31	Data of advertising packet	
adv_report.rssi	1	RSSI value for advertising packet	

Table 188. GAPM_ADV_REPORT_IND example

Message stream (advertising report is received)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x10 0x0D	GAPM_ADV_REPORT_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x29 0x00	41 bytes
adv_report.evt_type	0x00	ADV_CONN_UNDIR: connectable undirected advertising
adv_report.adv_addr_type	0x00	ADDR_PUBLIC: public BD address
adv_report.adv_addr	0x1B 0xA0 0x70 0xCA 0xEA 0x80	Advertising BD address: 0x80EACA70A01B
adv_report.data_len	0x1B	27 bytes
adv_report.data	0x02 0x01 0x06 0x11 0x07 0xB7 0x5C 0x49 0xD2 0x04 0xA3 0x40 0x71 0xA0 0xB5 0x35 0x85 0x3E 0xB0 0x83 0x07 0x05 0x09 0x78 0x49 0x4F 0x54 0x00 0x00 0x00 0x00	
adv_report.rssi	0xBD	-67dB
Message string		
0x05 0x10 0x0D 0x10 0x00 0x0D 0x00 0x29 0x00 0x00 0x00 0x1B 0xA0 0x70 0xCA 0xEA 0x80 0x1B 0x02 0x01 0x06 0x11 0x07 0xB7 0x5C 0x49 0xD2 0x04 0xA3 0x40 0x71 0xA0 0xB5 0x35 0x85 0x3E 0xB0 0x83 0x07 0x05 0x09 0x78 0x49 0x4F 0x54 0x00 0x00 0x00 0x00 0xBD		

The DA14585/531 GAPM sends a completion event message (GAPM_CMP_EVT) upon completion of the scan with the operation set to GAPM_SCAN_ACTIVE(0x11) or GAPM_SCAN_PASSIVE(0x12), depending on the type of the completed scan.

Table 189. GAPM_CMP_EVT for GAPM_START_SCAN_CMD example

Message stream: GAPM_CMP_EVT for GAPM_START_SCAN_CMD		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x12	GAPM_SCAN_PASSIVE
status	0x45	GAP_ERR_TIMEOUT
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x12 0x45		

5.3.13 Service Changed Indication

The GTL message GATTC_SVC_CHANGED_CFG_IND is sent to the application each time when the Client Characteristic Configuration Descriptor value of the Service Changed Characteristic (see volume 3, section 7.1 in Ref. [1]) is written by a peer device.

Table 190. GATTC_SVC_CHANGED_CFG_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_SVC_CHANGED_CFG_IND	0x0C12
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
PAR_LEN	2	2 bytes	0x0002
Message parameters			
ind_cfg	2	Client Characteristic Configuration Descriptor value	See Vol 3, Part G, 3.3.3.3, Table 3.11 in Ref. [1].

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 191. GATTC_SVC_CHANGED_CFG_IND example

Message stream: connection parameters update indication		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x12 0x0C	GATTC_SVC_CHANGED_CFG_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
Ind_cfg	0x02 0x00	0x0002: Indications enabled
Message string		
0x05 0x12 0x0C 0x10 0x00 0x0C 0x00 0x02 0x00 0x02 0x00		

5.3.14 Triggering a Disconnection

The GTL Host sends the message GAPC_DISCONNECT_CMD to the Bluetooth LE application to trigger a disconnection.

Table 192. GAPC_DISCONNECT_CMD symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_DISCONNECT_CMD	0x0E04
DST_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	2 bytes	0x0002
Message parameter			
operation	1	Request type	**0x01**: GAPC_DISCONNECT (See Table 372)
reason	1	Reason for the disconnection	**0x13**: CO_ERROR_REMOTE_USER_TERM_CON (See Table 381)

Table 193. GAPC_DISCONNECT_CMD example

Message stream (disconnection command)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x04 0x0E	GAPC_DISCONNECT_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0x00	2 bytes
operation	0x01	0x01: GAPC_DISCONNECT
reason	0x13	CO_ERROR_REMOTE_USER_TERM_CON
Message string		
0x05 0x04 0x0E 0x0E 0x00 0x10 0x00 0x02 0x00 0x01 0x13		

In the example application, this command is used to disconnect the peer central when encryption fails. The GAPC_DISCONNECT_IND message is received in step 1 in the disconnection sequence.

5.3.15 Disconnection Sequence

The following steps describe the messaging following a disconnection of a device that has been connected and performs non-connectable advertising.

1. Step 1: The disconnection takes place.

   After the disconnection takes place, the GTL Host is informed with the message GAPC_DISCONNECT_IND.

Table 194. GAPC_DISCONNECT_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_DISCONNECT_IND	0x0E03
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
PAR_LEN	2	4 bytes	0x0004
Message parameters			
conhdl	2	The handle of the current connection	
reason	1	The reason why the connection is about to be terminated	**0x13**: CO_ERROR_REMOTE_USER_TERM_CON **0x16**: CO_ERROR_CON_TERM_BY_LOCAL_HOST (See Table 381)
{padding}	1	{padding}	Ignored. Can consist of any value

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 195. GAPC_DISCONNECT_IND example

Message stream: GAPC_DISCONNECT_IND		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x0E	GAPC_DISCONNECT_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x04 0x00	0x0004 (4 bytes)
conhdl	0x00 0x00	0x0000 (connection handle 0)
reason	0x16	**0x16**: CO_ERROR_CON_TERM_BY_LOCAL_HOST
{padding}	0x00	Can consist of any value
Message string		
0x05 0x03 0x0E 0x10 0x00 0x0E 0x00 0x04 0x00 0x00 0x00 0x16 0x00		

2. Step 2: Stop the non-connectable advertising.

   Upon reception of the GAPC_DISCONNECT_IND message, the GTL Host sends the GAPM_CANCEL_CMD to stop the non-connectable advertising so that an undirect advertising can start.

   If non-connectable advertising is not used at all (advertising is disabled while the device is connected), this step and, consequently, the next step (step 3) are both skipped.

Table 196. GAPM_CANCEL_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05

Parameters	Bytes	Description	Data
MSG_ID	2	GAPM_CANCEL_CMD	0x0D03
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	The length of the parameters in bytes	0x0001 (one byte)
Message parameters			
operation	1	The GAPM operation code See Table 374	**0x00**: GAPM_NO_OP **0x02**: GAPM_CANCEL

Table 197. GAPM_CANCEL_CMD example

Message stream: GAPM_CANCEL_CMD		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x0D	GAPM_CANCEL_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	0x0001 (1 byte)
operation	0x02	GAPM_CANCEL
Message string		
0x05 0x03 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0x02		

3. Step 3: Receive completion event message for non-connectable advertising having been stopped.

   When the non-connectable advertising stops, the Bluetooth LE application informs the GTL Host with the completion event message (GAPM_CMP_EVT) of the GAPM_ADV_NON_CONN operation.

   If non-connectable advertising is not used at all (advertising is disabled while the device is connected), the previous step (step 2) and, consequently, this step are both skipped.

Table 198. GAPM_CMP_EVT for GAPM_ADV_NON_CONN example

Message stream: GAPM_CMP_EVT for GAPM_ADV_NON_CONN operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x0C	GAPM_ADV_NON_CONN
status	0x44	GAP_ERR_CANCELED
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x0C 0x44		

4. Step 4: Receive completion event message for the disconnection.

   The Bluetooth LE application informs the GTL Host with the completion event message GAPC_CMP_EVT of the GAPC_DISCONNECT operation to inform about the link disconnection.

Table 199. GAPC_CMP_EVT for GAPC_DISCONNECT operation example

Message stream: GAPC_CMP_EVT for GAPC_DISCONNECT operation		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x01	GAPC_DISCONNECT
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x01 0x00		

5. Step 5: The undirected advertising starts.

   The GTL Host sends the command GAPM_START_ADVERTISE_CMD to start the undirected advertising, exactly like in Section 5.3.1.7.

5.3.16 Supplying Information about Local Device (GAPC)

1. Step 1: The Bluetooth LE stack sends a request.

   The Bluetooth LE stack sends GAPC_GET_DEV_INFO_REQ_IND to retrieve information about the local device (DA14585/531 configuration) from the GTL Host.

   This event is triggered if a parameter of the local device configuration (see the operation parameter in Table 200) is requested by a peer device.

Table 200. GAPC_GET_DEV_INFO_REQ_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_GET_DEV_INFO_REQ_IND	0x0E0A
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
PAR_LEN	2	1 byte	0x0001
Message parameters			
operation	1	Requested information	**0x00**: GAPC_DEV_NAME, device name **0x01**: GAPC_DEV_APPEARANCE, device appearance icon **0x02**: GAPC_DEV_SLV_PERF_PARAMS, device slave preferred parameters **0x03**: GAPC_DEV_CENTRAL_RPA, Device Central Address Resolution parameters **0x04**: GAPC_DEV_RPA_ONLY, Device Resolvable Private Address Only parameters

Table 201. GAPC_GET_DEV_INFO_REQ_IND example

Message stream: example to get device name		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0A 0x0E	GAPC_GET_DEV_INFO_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x01 0x00	1 byte
operation	0x00	GAPC_DEV_NAME
Message string		
0x05 0x0A 0x0E 0x10 0x00 0x0E 0x00 0x01 0x00 0x00		

2. Step 2: The GTL Host responds with information about the device.

   The GTL Host responds to the request from the Bluetooth LE stack with the message GAPC_GET_DEV_INFO_CFM.

Table 202. GAPC_GET_DEV_INFO_CFM symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_GET_DEV_INFO_CFM	0x0E0B
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Depending on req	0xXXXX
Message parameters			
req	1	Requested information	**0x00**: GAPC_DEV_NAME, device name **0x01**: GAPC_DEV_APPEARANCE, device appearance icon **0x02**: GAPC_DEV_SLV_PREF_PARAMS, device slave preferred parameters **0x03**: GAPC_DEV_CENTRAL_RPA, device Central Address Resolution parameters **0x04**: GAPC_DEV_RPA_ONLY, device Resolvable Private Address Only parameters
{padding}	1	Ignored. Any value is valid.	0xXX
req = 0x00 (GAPC_DEV_NAME)			
info.name.length	2	2 bytes	0xXXXX
info.name.value	Depending on name length	Peer device name	
req = 0x01 (GAPC_DEV_APPEARANCE)			
appearance	2	Peer device appearance icon	0xXXXX

Parameters	Bytes	Description	Data
req = 0x02 (GAPC_DEV_SLV_PREF_PARAMS)			
info. slv_params.con_intv_min	2	Connection interval minimum	0xXXXX
info. slv_params.con_intv_max	2	Connection interval maximum	0xXXXX
info. slv_params.slave_latency	2	Slave latency	0xXXXX
info. slv_params.conn_timeout	2	Connection supervision timeout multiplier	0xXXXX
req = 0x03 (GAPC_DEV_CENTRAL_RPA)			
info.central_rpa	1	Central Address Resolution	0xXX
req = 0x04 (GAPC_DEV_RPA_ONLY)			
info.rpa_only	1	Resolvable Private Address Only	0xXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 203. GAPC_GET_DEV_INFO_CFM example

Message stream: GAPC_GET_DEV_INFO_CFM of GAPC_DEV_NAME operation		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0B 0x0E	GAPC_GET_DEV_INFO_CFM
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x27 0x00	39 bytes
req	0x00	GAPC_DEV_NAME
{padding}	0x00	{padding} Ignored. Any value is valid.
length	0x1D 0x00	0x001D = 29 bytes
name	0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x47 0x54 0x4C 0x20 0x6F 0x76 0x65 0x72 0x20 0x53 0x50 0x49 0x48 0x44 0x44 0x52 0x20 0x44 0x65 0x6D 0x6F 0x00	Renesas GTL over SPIHDDR Demo
{padding}	0x00 0x00 0x00 0x00 0x00 0x00	
Message string		
0x05 0x0B 0x0E 0x0E 0x00 0x10 0x00 0x27 0x00 0x00 0x00 0x1D 0x00 0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x47 0x54 0x4C 0x20 0x6F 0x76 0x65 0x72 0x20 0x53 0x50 0x49 0x48 0x44 0x44 0x52 0x20 0x44 0x65 0x6D 0x6F 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

5.3.17 Create a New Elliptic Curve Diffie Hellman (ECDH) Key

The GTL Host sends the GAPM_RESET_CMD message with the operation set to 0x2B (GAPM_KEY_RENEW) to request DA14585/531 to generate a new Elliptic Curve Diffie Hellman (ECDH) key.

Table 204. GAPM_RESET_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_RESET_CMD	0x0D02
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	1	0x0001
Message parameters			
operation	1	GAPM_KEY_RENEW	**0x2B**: GAPM_KEY_RENEW

Table 205. GAPM_RESET_CMD example

Message stream: GAPM_RESET_CMD - GAPM_KEY_RENEW		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x02 0x0D	GAPM_RESET_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x2B	GAPM_KEY_RENEW
Message string		
0x05 0x02 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0X2B		

5.3.18 Random Address Resolution

1. Step 1: The GTL Host requests the Bluetooth LE application for resolution of a device's Bluetooth LE address.

Table 206. GAPM_RESOLV_ADDR_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_RESOLV_ADDR_CMD	0x0D14
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	24 bytes	0x0018
Message parameters			
operation	1	GAPM_RESOLV_ADDR See Table 374	GAPM requested operation: GAPM_RESOLV_ADDR (0x17 = 23(d))
nb_key	1	Count of IRK keys provided	The count of IRKs available (bonding data available, already bonded with hosts).
addr	6	The random address of the central device	

Parameters	Bytes	Description	Data
Header			
irk	16	The list of the IRKs to use for resolution (MSB→LSB)	

2. Step 2: The GTL Host receives GAPM_ADDR_SOLVED_IND upon successful address resolution.

Table 207. GAPM_ADDR_SOLVED_IND – symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_ADDR_SOLVED_IND	0x0D15
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	22 bytes	0x0016
Message parameters			
bd_addr	6	The resolvable random address of the central which has been solved	Central Device random address
IRK	16	The IRK that has correctly solved the random address	

3. Step 3: Reception of the message GAPM_CMP_EVT.

DA14585/531 responds to the GTL Host with the GAPM_CMP_EVT message (Symbolic message structure reference can be found in Table 51) containing the following status error code:

- GAP_ERR_NO_ERROR (0x00), if the central device is identified as a known host.
- GAP_ERR_NOT_FOUND (0x47), if the central device is not identified as a known host.

See Table 44 for an example.

5.3.19 Send a Service Changed Command to a Peer Device

The GTL Host sends the **GATTC_SEND_SVC_CHANGED_CMD** message to indicate that a service has been changed.

Table 208. GATTC_SEND_SVC_CHANGED_CMD symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_SEND_SVC_CHANGED_CMD	0x0C11
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	8 bytes	0x0008
Message parameters			
operation	1	GATTC_SVC_CHANGED	0x14
{padding}	1	{padding}. Ignored	Can consist of any value
seq_num	2	Operation sequence number	0x1234
svc_shdl	2	Service start handle	0x0000
svc_ehdl	2	Service end handle	0x0001

Table 209. GAPM_SEND_SVC_CHANGED_CMD example

Message stream - GATTC_SEND_SVC_CHANGED_CMD		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x11 0x0C	GATTC_SEND_SVC_CHANGED_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x08 0x00	8 bytes
operation	0x14	GATTC_SVC_CHANGED
{padding}	0x00	{padding} Ignored. Any value is valid
seq_num	0x34 0x12	Sequence number 0x1234
svc_shdl	0x01 0x00	Service Start handle
svc_ehdl	0x03 0x00	Service End handle
Message string		
0x05 0x11 0x0C 0x0C 0x00 0x10 0x00 0x08 0x00 0x14 0x00 0x34 0x12 0x01 0x00 0x03 0x00		

5.3.20 Indication of Message with Unknown Destination ID

The Bluetooth LE application sends the **GAPM_UNKNOWN_TASK_IND** message to the GTL Host to indicate that a received message contains an invalid destination ID. The indication contains the received message ID and the received task ID.

Table 210. GAPM_UNKNOWN_TASK_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_UNKNOWN_TASK_IND	0x0D1D
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	4 bytes	0x0004
Message parameters			
msg_id	2	The message id of the command sent	0x0C11
task_id	2	Task id	0x0004 (always TASK_ID_GTL)

Table 211. GAPM_UNKNOWN_TASK_IND example

Message stream GAPM_UNKNOWN_TASK_IND		
In this example, the Bluetooth LE application sends a GATTC_SEND_SVC_CHANGED_CMD with an invalid destination ID to trigger the GAPM_UNKNOWN_TASK_IND.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1D 0x0D	GAPM_UNKNOWN_TASK_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
msg_id	0x11 0x0C	GATTC_SEND_SVC_CHANGED_CMD
task_id	0x04 0x00	Task id
Message string		
0x05 0x1D 0x0D 0x0D 0x00 0x10 0x00 0x04 0x00 0x11 0x0C 0x04 0x00		

5.3.21 Peer Device Request to Modify Local Device Info

5.3.21.1.1 GAPC_SET_DEV_INFO_REQ_IND

1. Step 1: The Bluetooth LE application sends the GAPC_SET_DEV_INFO_REQ_IND message to the GTL Host to indicate that the peer device requests to modify local device info such as the device name or device appearance.

Table 212. GAPC_SET_DEV_INFO_REQ_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_SET_DEV_INFO_REQ_IND	0x0E0C
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPC	0x000E
PAR_LEN	2	Depending on name length	0xXX
Message parameters			
req	1	Requested information	**0x00**: GAPC_DEV_NAME **0x01**: GAPC_DEV_APPEARANCE
{padding}	1	{padding}. Ignored	Can consist of any value
gapc_set_dev_info	Depending on name length	Device information data	
gapc_set_dev_info parameters			
gap_dev_name	Depending on name length	Device name	
appearance	2	Appearance Icon	Not used in this example
gap_dev_name parameters			
length	2	Name length	0xXXXX
value	Depending on name length	Name value	0xXXXX

Table 213. GAPC_SET_DEV_INFO_REQ_IND example

Message stream: GAPC_SET_DEV_INFO_REQ_IND for GAPC_DEV_NAME operation		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0C 0x0E	GAPC_SET_DEV_INFO_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC
PAR_LEN	0x06 0x00	Depends on name length (0x1342)
req	0x00	GAPC_DEV_NAME
{padding}	0x00	{padding} Ignored. Any value is valid
length	0x02 0x00	2 bytes
value	0x34 0x12	Random numbers
Message string		
0x05 0x0C 0x0E 0x10 0x00 0x0E 0x00 0x06 0x00 0x00 0x00 0x02 0x00 0x34 0x12		

5.3.21.1.2 GAPC_SET_DEV_INFO_CFM

2. Step 2: The GTL Host responds to GAPC_SET_DEV_INFO_REQ_IND by a message GAPC_SET_DEV_INFO_CFM. The confirmation message contains the requested information and whether the request has been accepted or not.

Table 214. GAPC_SET_DEV_INFO_CFM symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_SET_DEV_INFO_CFM	0x0E0D
DST_ID	2	TASK_ID_GAPC	0x000E
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	2 bytes	0x0002
Message parameters			
req	1	Requested information	**0x00**: GAPC_DEV_NAME **0x01**: GAPC_DEV_APPEARANCE
status	1	Status code (accepted or not)	

Table 215. GAPC_SET_DEV_INFO_CFM example

Message stream GAPC_SET_DEV_INFO_CFM for GAPC_DEV_NAME operation		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0D 0x0E	GAPC_SET_DEV_INFO_CFM
DST_ID	0x0E 0x00	TASK_ID_GAPC
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0x00	Parameters length
req	0x00	GAPC_DEV_NAME
status	0x48	GAP_ERR_REJECTED
Message string		
0x05 0x0D 0x0E 0x0E 0x00 0x10 0x00 0x02 0x00 0x00 0x48		

5.3.22 Indicate the Current Sign Counters to the Application

The Bluetooth LE application sends a **GAPC_SIGN_COUNTER_IND** message to the GTL Host to indicate the current sign counters (local and peer) after **GATTC_WRITE_CMD** from the peer device after the **GATTC_WRITE_SIGNED** operation takes place in the GATTC Database of DA14585/531.

Table 216. GAPC_SIGN_COUNTER_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_SIGN_COUNTER_IND	0x0E1C
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	8 bytes	0x0008
Message parameters			
local_sign_counter	4	Local SignCounter value	0xXXXXXXXX
peer_sign_counter	4	Peer SignCounter value	0xXXXXXXXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

For an example on the peripheral side, see Table 167.

5.3.23 LE Ping Timeout Expires Indication

The Bluetooth LE application sends **GAPC_LE_PING_TO_IND** to the GTL Host when the "authenticatedPayloadTO" expires.

Table 217. GAPC_LE_PING_TO_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LE_PING_TO_IND	0x0E2A
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E, nn = conidx

Parameters	Bytes	Description	Hex data
Header			
PAR_LEN	2	No params	0x0000

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 218. GAPC_LE_PING_TO_IND example

Message stream GAPC_LE_PING_TO_IND		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x2A 0x0E	GAPC_LE_PING_TO_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x00 0x00	No parameters
Message string		
0x05 0x2A 0x0E 0x10 0x00 0x0E 0x00 0x00 0x00		

5.3.24 Get Information about Local Device

The GTL Host sends this command to its Bluetooth LE application to retrieve information about the local device. The GTL Host then receives an indication that depends on the operation of **GAPM_GET_DEV_INFO_CMD**.

Table 219. GAPM_GET_DEV_INFO_CMD symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_GET_DEV_INFO_CMD	0x0D06
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	1 byte	0x0001
Message parameters			
operation	1	See Table 374	0x05: GAPM_GET_DEV_VERSION 0x06: GAPM_GET_DEV_BDADDR 0x07: GAPM_GET_DEV_ADV_TX_POWER 0x1C: GAPM_DBG_GET_MEM_INFO 0x1F: GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN 0x20: GAPM_GET_MAX_LE_DATA_LEN

Depending on the operation, the Bluetooth LE application responds with the corresponding indication (See Table 220).

Table 220. Indication according to GAPM_GET_DEV_INFO_CMD operation

Hex code	Operation (See Table 374)	Indication
0x05	GAPM_GET_DEV_VERSION	GAPM_DEV_VERSION_IND
0x06	GAPM_GET_DEV_BDADDR	GAPM_DEV_BDADDR_IND
0x07	GAPM_GET_DEV_ADV_TX_POWER	GAPM_DEV_ADV_TX_POWER_IND

Hex code	Operation (See Table 374)	Indication
0x1C	GAPM_DBG_GET_MEM_INFO	GAPM_DBG_MEM_INFO_IND
0x1F	GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN	GAPM_SUGG_DFLT_DATA_LEN_IND
0x20	GAPM_GET_MAX_LE_DATA_LEN	GAPM_MAX_DATA_LEN_IND

5.3.24.1 GAPM_GET_DEV_VERSION

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the version of the local device.

Table 221. GAPM_GET_DEV_INFO_CMD with operation GAPM_GET_DEV_VERSION example

Message stream - GAPM_GET_DEV_INFO_CMD with operation GAPM_GET_DEV_VERSION		
Symbolic message structure reference can be found in Table 219.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0D	GAPM_GET_DEV_INFO_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x05	GAPM_GET_DEV_VERSION
Message string		
0x05 0x06 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0x05		

2. Step 2: The GTL Host receives GAPM_DEV_VERSION_IND from DA14585/531.

   GAPM_DEV_VERSION_IND contains information about the version of the local device.

Table 222. GAPM_DEV_VERSION_IND symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_DEV_VERSION_IND	0x0D07
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	12 bytes	0x000C
Message parameters			
hci_ver	1	HCI version	0xXX
lmp_ver	1	LMP version	0xXX
host_ver	1	Host version	0xXX
{padding}	1	{padding}	Ignored. Any value is valid.
hci_subver	2	HCI revision	0xXX 0xXX
lmp_subver	2	LMP subversion	0xXX 0xXX
host_subver	2	Host revision	0xXX 0xXX
manuf_name	2	Manufacturer name	0xXX 0xXX

Table 223. GAPM_DEV_VERSION_IND example

Message stream: GAPM_DEV_VERSION_IND		
Symbolic message structure reference can be found in Table 222.		
Hex data		
Data token	Hex data (LSB)	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x07 0x0D	GAPM_DEV_VERSION_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x0C 0x00	12 bytes
hci_ver	0x0A	HCI version
lmp_ver	0x0A	LMP version
host_ver	0x08	Host version
{padding}	0x00	Ignored. Any value is valid
hci_subver	0x0F 0x01	HCI revision
lmp_subver	0x0F 0x01	LMP subversion
host_subver	0x0E 0x01	Host revision
manuf_name	0xD2 0x00	Manufacturer name: Renesas Electronics
Message string		
0x05 0x07 0x0D 0x10 0x00 0x0D 0x00 0x0C 0x00 0x0A 0x0A 0x08 0x00 0x0F 0x01 0x0F 0x01 0x0E 0x01 0xD2 0x00		

3. Step 3: After the Bluetooth LE application has sent **GAPM_DEV_VERSION_IND** to the GTL Host, it also sends **GAPM_CMP_EVT** for **GAPM_GET_DEV_INFO_CMD.**

Table 224. GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_DEV_VERSION

Message stream: GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_DEV_VERSION		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x05	GAPM_GET_DEV_VERSION
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x05 0x00		

5.3.24.2 GAPM_GET_DEV_BDADDR

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the BD address of the local device.

Table 225. GAPM_GET_DEV_INFO_CMD with operation GAPM_GET_DEV_BDADDR example

Message stream - GAPM_GET_INFO_CMD with operation GAPM_GET_DEV_BDADDR		
Symbolic message structure reference can be found in Table 219.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0D	GAPM_GET_DEV_INFO_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x06	GAPM_GET_DEV_BDADDR
Message string		
0x05 0x06 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0x06		

2. Step 2: The GTL Host receives GAPM_DEV_BDADDR_IND from DA14585/531.

   GAPM_DEV_BDADDR_IND contains information about the BD address of the local device.

   This event can be triggered when reading local BD Address, but also when starting an air operation (advertising, connecting, scanning) to inform application about the used random address.

Table 226. GAPM_DEV_BDADDR_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_DEV_BDADDR_IND	0x0D08
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	7 bytes	0x0007
Message parameters			
addr.addr	6	BD address of the device	0xXXXXXXXXXXXX
addr_type	1	BD Address type of the device	**0x00**: public **0x01**: random

Table 227. GAPM_DEV_BDADDR_IND example

Message stream: GAPM_DEV_BDADDR_IND		
Symbolic message structure reference can be found in Table 226.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x08 0x0D	GAPM_DEV_BDADDR_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x07 0x00	7 bytes
addr.addr	0x07 0x07 0x70 0xCA 0xEA 0x80	BD address of the device
addr_type	0x00	Public Address
Message string		
0x05 0x08 0x0D 0x10 0x00 0x0D 0x00 0x07 0x00 0x07 0x07 0x70 0xCA 0xEA 0x80 0x00		

3. Step 3: After the Bluetooth LE application has sent **GAPM_DEV_ BDADDR _IND** to the GTL Host, it also sends **GAPM_CMP_EVT** for **GAPM_GET_DEV_INFO_CMD.**

Table 228. GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_DEV_BDADDR

Message stream: GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_DEV_BDADDR		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x06	GAPM_GET_DEV_BDADDR
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x06 0x00		

5.3.24.3 GAPM_GET_DEV_ADV_TX_POWER

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the advertising power level of the local device.

Table 229. GAPM_GET_DEV_INFO_CMD with operation GAPM_GET_DEV_ADV_TX_POWER example

Message stream - GAPM_GET_INFO_CMD with operation GAPM_GET_DEV_ADV_TX_POWER		
Symbolic message structure reference can be found in Table 219.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0D	GAPM_GET_DEV_INFO_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x07	GAPM_GET_DEV_ADV_TX_POWER
Message string		
0x05 0x06 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0x07		

2. Step 2: The GTL Host receives GAPM_DEV_ADV_TX_POWER_IND from DA14585/531.

   GAPM_DEV_ADV_TX_POWER_IND contains information about the advertising power level of the local device.

Table 230. GAPM_DEV_ADV_TX_POWER_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_DEV_ADV_TX_POWER_IND	0x0D09
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	1 byte	0x0001
Message parameters			
power_lvl	1	Advertising channel Tx power level	0xXX

Table 231. GAPM_DEV_ADV_TX_POWER_IND example

Message stream: GAPM_DEV_ADV_TX_POWER_IND		
Symbolic message structure reference can be found in Table 230.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x09 0x0D	GAPM_DEV_ADV_TX_POWER_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x01 0x00	1 byte
power_lvl	0x09	Advertising channel Tx power level
Message string		
0x05 0x09 0x0D 0x10 0x00 0x0D 0x00 0x01 0x00 0x09		

3. Step 3: After the Bluetooth LE application has sent **GAPM_DEV_ADV_TX_POWER_IND** to the GTL Host, it also sends **GAPM_CMP_EVT** for **GAPM_GET_DEV_INFO_CMD.**

Table 232. GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_DEV_ADV_TX_POWER

Message stream: GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_DEV_ADV_TX_POWER		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x07	GAPM_GET_DEV_ADV_TX_POWER
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x07 0x00		

5.3.24.4 GAPM_DBG_GET_MEM_INFO – Debug Only

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the memory usage of the local device.

Table 233. GAPM_GET_DEV_INFO_CMD with operation GAPM_DBG_GET_MEM_INFO example

Message stream - GAPM_GET_INFO_CMD with operation GAPM_DBG_GET_MEM_INFO		
Symbolic message structure reference can be found in Table 219.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0D	GAPM_GET_DEV_INFO_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x1C	GAPM_DBG_GET_MEM_INFO
Message string		
0x05 0x06 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0x1C		

2. Step 2: The GTL Host receives GAPM_DBG_MEM_INFO_IND from DA14585/531.

   GAPM_DBG_MEM_INFO_IND contains information about the memory usage of the local device.

Table 234. GAPM_DBG_MEM_INFO_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_DBG_MEM_INFO_IND	0x0D0A
DST_ID	2	TASK_ID_GTL	0x0010

Parameters	Bytes	Description	Hex data
Header			
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	12 bytes	0x000C
Message parameters			
max_mem_used	4	Peak of memory usage measured	0xXXXXXXXX
mem_used	8	Memory size currently used into each heap	0xXXXX

Table 235. GAPM_DBG_MEM_INFO_IND example

Message stream: GAPM_DBG_MEM_INFO_IND		
Symbolic message structure reference can be found in Table 234.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0A 0x0D	GAPM_DBG_MEM_INFO_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x0C 0x00	12 bytes
max_mem_used	0x78 0x00 0x00 0x00	Peak of memory usage measured
mem_used	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	Memory size currently used into each heap
Message string		
0x05 0x0A 0x0D 0x10 0x00 0x0D 0x00 0x0C 0x00 0x78 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

3. Step 3: After the Bluetooth LE application has sent **GAPM_DBG_MEM_INFO_IND** to the GTL Host, it also sends **GAPM_CMP_EVT** for **GAPM_GET_DEV_INFO_CMD.**

Table 236. GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_DBG_GET_MEM_INFO

Message stream: GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_DBG_GET_MEM_INFO		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x1C	GAPM_DBG_GET_MEM_INFO
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x1C 0x00		

5.3.24.5 GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the suggested default LE data length of the local device.

Table 237. GAPM_GET_DEV_INFO_CMD with operation GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN example

Message stream - GAPM_GET_INFO_CMD with operation GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN		
Symbolic message structure reference can be found in Table 219.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0D	GAPM_GET_DEV_INFO_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x1F	GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN
Message string		
0x05 0x06 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0x1F		

2. Step 2: The GTL Host receives GAPM_SUGG_DFLT_DATA_LEN_IND from DA14585/531.

   GAPM_SUGG_DFLT_DATA_LEN_IND contains information about the suggested default LE data length of the local device.

Table 238. GAPM_SUGG_DFLT_DATA_LEN_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_SUGG_DFLT_DATA_LEN_IND	0x0D1E
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	4 bytes	0x0004
Message parameters			
suggted_max_tx_octets	2	Host's suggested value for the Controller's maximum transmitted number of payload octets	0xXXXX
suggted_max_tx_time	2	Host's suggested value for the Controller's maximum packet transmission time	0xXXXX

Table 239. GAPM_SUGG_DFLT_DATA_LEN_IND example

Message stream: GAPM_SUGG_DFLT_DATA_LEN_IND		
Symbolic message structure reference can be found in Table 238.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1E 0x0D	GAPM_SUGG_DFLT_DATA_LEN_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x04 0x00	4 bytes
suggted_max_tx_octets	0x1B 0x00	0x001B = 27 octects
suggted_max_tx_time	0x48 0x01	0x0148 = 328 ms
Message string		
0x05 0x1E 0x0D 0x10 0x00 0x0D 0x00 0x04 0x00 0x1B 0x00 0x48 0x01		

3. Step 3: After the Bluetooth LE application has sent **GAPM_SUGG_DFLT_DATA_LEN_IND** to the GTL Host, it also sends **GAPM_CMP_EVT** for **GAPM_GET_DEV_INFO_CMD.**

Table 240. GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN

Message stream: GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x1F	GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x1F 0x00		

5.3.24.6 GAPM_GET_MAX_LE_DATA_LEN

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the max LE data length of the local device.

Table 241. GAPM_GET_DEV_INFO_CMD with operation GAPM_GET_MAX_LE_DATA_LEN example

Message stream - GAPM_GET_DEV_INFO_CMD with operation GAPM_GET_MAX_LE_DATA_LEN		
Symbolic message structure reference can be found in Table 219.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0D	GAPM_GET_DEV_INFO_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x20	GAPM_GET_MAX_LE_DATA_LEN
Message string		
0x05 0x06 0x0D 0x0D 0x00 0x10 0x00 0x01 0x00 0x20		

2. Step 2: The GTL Host receives GAPM_MAX_DATA_LEN_IND from DA14585/531.

   GAPM_MAX_DATA_LEN_IND contains information about the memory usage of the local device.

Table 242. GAPM_MAX_DATA_LEN_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_MAX_DATA_LEN_IND	0x0D1F
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	8 bytes	0x0008
Message parameters			
suppted_max_tx_octets	2	Maximum number of payload octets that the local Controller supports for transmission	0xXXXX
suppted_max_tx_time	2	Maximum time, in microseconds, that the local Controller supports for transmission	0xXXXX
suppted_max_rx_octets	2	Maximum number of payload octets that the local Controller supports for reception	0xXXXX
suppted_max_rx_time	2	Maximum time, in microseconds, that the local Controller supports for reception	0xXXXX

Table 243. GAPM_MAX_DATA_LEN_IND example

Message stream: GAPM_MAX_DATA_LEN_IND		
Symbolic message structure reference can be found in Table 242.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1F 0x0D	GAPM_MAX_DATA_LEN_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x08 0x00	8 bytes
suppted_max_tx_octets	0xFB 0x00	0x00FB = 251 octets
suppted_max_tx_time	0x48 0x08	0x0848 = 2120 microseconds
suppted_max_rx_octets	0xFB 0x00	0x00FB = 251 octets
suppted_max_rx_time	0x48 0x08	0x0848 = 2120
Message string		
0x05 0x1F 0x0D 0x10 0x00 0x0D 0x00 0x08 0x00 0xFB 0x00 0x48 0x08 0xFB 0x00 0x48 0x08		

3. Step 3: After the Bluetooth LE application has sent **GAPM_MAX_DATA_LEN_IND** to the GTL Host, it also sends **GAPM_CMP_EVT** for **GAPM_GET_DEV_INFO_CMD.**

Table 244. GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_MAX_LE_DATA_LEN

Message stream: GAPM_CMP_EVT for GAPM_GET_DEV_INFO_CMD with operation of GAPM_GET_MAX_LE_DATA_LEN		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x20	GAPM_GET_MAX_LE_DATA_LEN
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x20 0x00		

5.3.25 Get Information about Peer Device and the Connection

The GTL Host sends this command to its Bluetooth LE application to retrieve information about a peer device or about a currently active link. The GTL Host then receives an indication that depends on the operation of **GAPC_GET_INFO_CMD**.

Table 245. GAPC_GET_INFO_CMD symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_GET_INFO_CMD	0x0E05
DST_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	2 bytes	0x0001
Message parameters			
operation	1	See Table 372	**0x02**: GAPC_GET_PEER_NAME **0x03**: GAPC_GET_PEER_VERSION **0x04:** GAPC_GET_PEER_FEATURES **0x05:** GAPC_GET_PEER_APPEARANCE **0x06:** GAPC_GET_PEER_SLV_PREF_PARAMS: **0x07:** GAPC_GET_CON_RSSI **0x08:** GAPC_GET_CON_CHANNEL_MAP **0x12:** GAPC_GET_LE_PING_TIMEOUT **0x15**: GAPC_GET_PEER_CENTRAL_RPA **0x16**: GAPC_GET_PEER_RPA_ONLY

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Depending on the operation, the Bluetooth LE application responds with the corresponding indication (See Table 246).

Table 246. Indication according to GAPC_GET_INFO_CMD operation

Hex code	Operation (See Table 372)	Indication
0x02	GAPC_GET_PEER_NAME	GAPC_PEER_ATT_INFO_IND
0x03	GAPC_GET_PEER_VERSION	GAPC_PEER_VERSION_IND
0x04	GAPC_GET_PEER_FEATURES	GAPC_PEER_FEATURES_IND
0x05	GAPC_GET_PEER_APPEARANCE	GAPC_PEER_ATT_INFO_IND
0x06	GAPC_GET_PEER_SLV_PREF_PARAMS	GAPC_PEER_ATT_INFO_IND
0x07	GAPC_GET_CON_RSSI	GAPC_CON_RSSI_IND
0x08	GAPC_GET_CON_CHANNEL_MAP	GAPC_CON_CHANNEL_MAP_IND
0x12	GAPC_GET_LE_PING_TIMEOUT	GAPC_LE_PING_TO_VAL_IND
0x15	GAPC_GET_PEER_CENTRAL_RPA	GAPC_PEER_ATT_INFO_IND
0x16	GAPC_GET_PEER_RPA_ONLY	GAPC_PEER_ATT_INFO_IND

5.3.25.1 GAPC_GET_PEER_NAME

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the peer's name.

Table 247. GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_NAME example

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_NAME		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x02	GAPC_GET_PEER_NAME
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x02		

2. Step 2: The GTL Host receives GAPC_PEER_ATT_INFO_IND from DA14585/531.

   GAPC_PEER_INFO_ATT_INFO_IND contains peer device attribute DB info such as Device Name, Appearance or Slave Preferred Parameters.

Table 248. GAPC_PEER_ATT_INFO_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_PEER_ATT_INFO_IND	0x0E06
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC	0xnn0E (nn = conidx) (Note 1)
PAR_LEN	2	Depending on attribute	0xXX 0xX
Message parameters			
req	1	Requested information	**0x00**: GAPC_DEV_NAME **0x01**: GAPC_DEV_APPEARANCE **0x02**: GAPC_DEV_SLV_PREF_PARAMS **0x03**: GAPC_DEV_CENTRAL_RPA **0x04**: GAPC_DEV_RPA_ONLY
{padding}	1	{padding}	Ignored. Any value is valid.
handle	2	Attribute handle	0xXX 0xXX
req = GAPC_DEV_NAME (0x00)			
info.name.length	2	Name Length	0xXX 0xXX
info.name.value	X	Name Value	0xXX 0xXX ….
req = GAPC_DEV_APPEARANCE (0x01)			
info.appearance	2	Appearance Icon	0xXX 0xXX
req = GAPC_DEV_SLV_PREF_PARAMS (0x02)			
info.slv_params.con_intv_min	2	Connection interval minimum	0xXX 0xXX

Parameters	Bytes	Description	Hex data
Header			
info.slv_params.con_intv_max	2	Connection interval maximum	0xXX 0xXX
info.slv_params.slave_latency	2	Slave Latency	0xXX 0xXX
info.slv_params.conn_timeout	2	Connection supervision timeout multiplier	0xXX 0xXX
req = GAPC_DEV_CENTRAL_RPA (0x03)			
info.central_rpa	1	Central Address Resolution	0xXX
req = GAPC_DEV_RPA_ONLY (0x04)			
info.rpa_only	1	Resolvable Private Address Only	0xXX

Table 249. GAPC_PEER_ATT_INFO_IND (GAPC_DEV_NAME) example

Message stream: GAPC_PEER_ATT_INFO_IND - GAPC_DEV_NAME		
Symbolic message structure reference can be found in Table 248.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0E	GAPC_PEER_ATT_INFO_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC
PAR_LEN	0x15 0x00	21 bytes
req	0x00	GAPC_DEV_NAME
{padding}	0x00	Ignored. Any value is valid
handle	0x16 0x00	Attribute Handle = 0x16
info.name.length	0x09 0x00	
info.name.value	0x47 0x61 0x6C 0x61 0x78 0x79 0x20 0x53 0x38	"Galaxy S8"
{padding}	0x00 0x00 0x00 0x00 0x00 0x00	
Message string		
0x05 0x06 0x0E 0x10 0x00 0x0E 0x00 0x15 0x00 0x00 0x00 0x16 0x00 0x09 0x00 0x47 0x61 0x6C 0x61 0x78 0x79 0x20 0x53 0x38 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x02 0x00		

3. Step 3: After the Bluetooth LE application has sent **GAPC_PEER_ATT_INFO_IND** (GAPC_DEV_NAME) to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 250. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_NAME

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_NAME		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x02	GAPC_GET_PEER_NAME
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x02 0x00		

5.3.25.2 GAPC_GET_PEER_VERSION

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the peer device info.

Table 251. GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_VERSION example

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_APPEARANCE		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x03	GAPC_GET_PEER_VERSION
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x03		

2. Step 2: The GTL Host receives **GAPC_PEER_VERSION_IND** from DA14585/531 which contains peer device version info.

Table 252. GAPC_PEER_VERSION_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_PEER_VERSION_IND	0x0E07
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E, nn = conidx
PAR_LEN	2	6 bytes	0x0006

Parameters	Bytes	Description	Hex data
Header			
Message parameters			
compid	2	Manufacturer name	SIG code
lmp_subvers	2	LMP subversion	
lmp_vers	1	LMP version	
{padding}	1	{padding} Ignored.	Any value is valid

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 253. GAPC_PEER_VERSION_IND example

Message stream GAPC_PEER_VERSION_IND		
Symbolic message structure reference can be found in Table 217.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x07 0x0E	GAPC_PEER_VERSION_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x06 0x00	6 bytes
compid	0x0F 0x00	Bluetooth LE Chip Company ID (Broadcom)
lmp_subvers	0x1A 0x41	LMP subversion = 0x411A
lmp_vers	0x09	LMP version = 0x09
{padding}	0x00	{padding} Ignored. Any value is valid
Message string		
0x05 0x07 0x0E 0x10 0x00 0x0E 0x00 0x06 0x00 0x0F 0x00 0x1A 0x41 0x09 0x00		

3. Step 3: After the Bluetooth LE application has sent **GAPC_PEER_VERSION_IND** to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 254. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_VERSION

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_VERSION		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x03	GAPC_GET_PEER_VERSION
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x03 0x00		

5.3.25.3 GAPC_GET_PEER_FEATURES

See Step 4 of Section 5.3.2.5.

5.3.25.4 GAPC_GET_PEER_APPEARANCE

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the peer's appearance.

Table 255. GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_APPEARANCE example

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_APPEARANCE		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x05	GAPC_GET_PEER_APPEARANCE
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x05		

2. Step 2: The GTL Host receives GAPC_PEER_ATT_INFO_IND from DA14585/531.

Table 256. GAPC_PEER_ATT_INFO_IND (GAPC_DEV_APPEARANCE) example

Message stream: GAPC_PEER_ATT_INFO_IND - GAPC_DEV_APPEARANCE		
Symbolic message structure reference can be found in Table 248.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0E	GAPC_PEER_ATT_INFO_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC
PAR_LEN	0x0C 0x00	12 bytes
req	0x01	GAPC_DEV_APPEARANCE
{padding}	0x00	Ignored. Any value is valid
handle	0x18 0x00	Attribute Handle = 0x18
info.appearance	0x00 0x00	Appearance Icon = 0x0000
{padding}	0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid
Message string		
0x05 0x06 0x0E 0x10 0x00 0x0E 0x00 0x0C 0x00 0x01 0x00 0x18 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

3. Step 3: After the Bluetooth LE application has sent **GAPC_PEER_ATT_INFO_IND** (GAPC_DEV_APPEARANCE) to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 257. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_APPEARANCE

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_APPEARANCE		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x05	GAPC_GET_PEER_APPEARANCE
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x05 0x00		

5.3.25.5 GAPC_GET_PEER_SLV_PREF_PARAMS

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the peer's slave preferred parameters.

Table 258. GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_SLV_PREF_PARAMS EXAMPLE

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_SLV_PREF_PARAMS		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x06	GAPC_GET_PEER_SLV_PREF_PARAMS
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x06		

2. Step 2: The GTL Host receives GAPC_PEER_ATT_INFO_IND from DA14585/531.

Table 259. GAPC_PEER_ATT_INFO_IND (GAPC_DEV_SLV_PREF_PARAMS) example

Message stream: GAPC_PEER_ATT_INFO_IND - GAPC_DEV_SLV_PREF_PARAMS		
Symbolic message structure reference can be found in Table 248.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0E	GAPC_PEER_ATT_INFO_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC
PAR_LEN	0x0C 0x00	12 bytes
req	0x02	GAPC_DEV_SLV_PREF_PARAMS
{padding}	0x00	Ignored. Any value is valid
handle	0x07 0x00	Attribute Handle = 0x07
info.slv_params.con_intv_min	0x40 0x01	Connection interval minimum
info.slv_params.con_intv_max	0x40 0x01	Connection interval maximum
info.slv_params.slave_latency	0x00 0x00	Slave Latency
info.slv_params.conn_timeout	0xF4 0x01	Connection supervision timeout
Message string		
0x05 0x06 0x0E 0x10 0x00 0x0E 0x00 0x0C 0x00 0x02 0x00 0x07 0x00 0x40 0x01 0x40 0x01 0x00 0x00 0xF4 0x01		

3. Step 3: After the Bluetooth LE application has sent **GAPC_PEER_ATT_INFO_IND** (GAPC_DEV_SLV_PREF_PARAMS) to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 260. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_SLV_PREF_PARAMS

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_SLV_PREF_PARAMS		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x06	GAPC_GET_PEER_SLV_PREF_PARAMS
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x06 0x00		

5.3.25.6 GAPC_GET_CON_RSSI

1. Step 1: The GTL Host sends the command to the Bluetooth LE application.

Table 261. GAPC_GET_INFO_CMD with operation GAPC_GET_CON_RSSI example

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_CON_RSSI		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x07	GAPC_GET_CON_RSSI
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x07		

2. Step 2: The GTL Host receives an indication depending on the operation. In this example it is **GAPC_CON_RSSI_IND**.

Table 262. GAPC_CON_RSSI_IND example

Message stream: GAPC_CON_RSSI_IND		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x09 0x0E	GAPC_CON_RSSI_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC
PAR_LEN	0x01 0x00	1 byte
rssi	0xE3	RSSI indication value
Message string		
0x05 0x09 0x0E 0x10 0x00 0x0E 0x00 0x01 0x00 0xE3		

3. Step 3: After the Bluetooth LE application has sent **GAPC_CON_RSSI_IND** to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 263. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_CON_RSSI

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_CON_RSSI		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x07	GAPC_GET_CON_RSSI
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x07 0x00		

5.3.25.7 GAPC_GET_CON_CHANNEL_MAP

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve the connection channel map.

Table 264. GAPC_GET_INFO_CMD with operation GAPC_GET_CON_CHANNEL_MAP example

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_CON_CHANNEL_MAP		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x08	GAPC_GET_CON_CHANNEL_MAP
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x08		

2. Step 2: The GTL Host receives **GAPC_CON_CHANNEL_MAP_IND** from DA14585/531 which contains peer device version info.

Table 265. GAPC_CON_CHANNEL_MAP_IND symbolic message structure

Parameters	**Bytes**	**Data**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_CON_CHANNEL_MAP_IND	0x0E1D
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E, nn = conidx
PAR_LEN	2	5 bytes	0x0005
Message parameters			
ch_map	5	5-byte channel map array	0xXXXXXXXXXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 266. GAPC_CON_CHANNEL_MAP_IND example

Message stream GAPC_CON_CHANNEL_MAP_IND		
Symbolic message structure reference can be found in Table 265.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1D 0x0E	GAPC_CON_CHANNEL_MAP_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x05 0x00	5 bytes
ch_map	0xFF 0xFF 0xFF 0xFF 0x1E	Channel map = 0x1EFFFFFFFF
Message string		
0x05 0x1D 0x0E 0x10 0x00 0x0E 0x00 0x05 0x00 0xFF 0xFF 0xFF 0xFF 0x1E		

3. Step 3: After the Bluetooth LE application has sent **GAPC_CON_CHANNEL_MAP_IND** to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 267. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_CON_CHANNEL_MAP

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_CON_CHANNEL_MAP		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x08	GAPC_GET_CON_CHANNEL_MAP
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x08 0x00		

5.3.25.8 GAPC_GET_LE_PING_TIMEOUT

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve the connection channel map.

Table 268. GAPC_GET_INFO_CMD with operation GAPC_GET_LE_PING_TIMEOUT example

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_LE_PING_TIMEOUT		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x12	GAPC_GET_LE_PING_TIMEOUT
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x12		

2. Step 2: The GTL Host receives **GAPC_LE_PING_TO_VAL_IND** from DA14585/531 which contains peer device version info.

Table 269. GAPC_LE_PING_TO_VAL_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LE_PING_TO_VAL_IND	0x0E29
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E, nn = conidx
PAR_LEN	2	2 bytes	0x0002
Message parameters			

Parameters	Bytes	Description	Hex data
Header			
timeout	2	Authenticated payload timeout	0xXXXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 270. GAPC_LE_PING_TO_VAL_IND example

Message stream GAPC_LE_PING_TO_VAL_IND		
Symbolic message structure reference can be found in Table 269.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x29 0x0E	GAPC_LE_PING_TO_VAL_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
timeout	0xB8 0x0B	Timeout =0x0BB8 = 3000 → 30 s
Message string		
0x05 0x29 0x0E 0x10 0x00 0x0E 0x00 0x02 0xB8 0x0B		

3. Step 3: After the Bluetooth LE application has sent **GAPC_LE_PING_TO_VAL_IND** to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 271. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_LE_PING_TIMEOUT

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_LE_PING_TIMEOUT		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x12	GAPC_GET_LE_PING_TIMEOUT
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x12 0x00		

5.3.25.9 GAPC_GET_PEER_CENTRAL_RPA

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the peer's appearance.

Table 272. GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_CENTRAL_RPA example

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_CENTRAL_RPA		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x15	GAPC_GET_PEER_CENTRAL_RPA
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x15		

2. Step 2: The GTL Host receives GAPC_PEER_ATT_INFO_IND from DA14585/531.

Table 273. GAPC_PEER_ATT_INFO_IND (GAPC_DEV_CENTRAL_RPA) example

Message stream: GAPC_PEER_ATT_INFO_IND - GAPC_DEV_CENTRAL_RPA		
Symbolic message structure reference can be found in Table 248.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0E	GAPC_PEER_ATT_INFO_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC
PAR_LEN	0x0C 0x00	12 bytes
req	0x15	GAPC_DEV_CENTRAL_RPA
{padding}	0x00	Ignored. Any value is valid
handle	0x1A 0x00	Attribute Handle = 0x1A
info.central_rpa	0x01	Device Central Address Resolution
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid
Message string		
0x05 0x06 0x0E 0x10 0x00 0x0E 0x00 0x0C 0x00 0x15 0x00 0x1A 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

3. Step 3: After the Bluetooth LE application has sent **GAPC_PEER_ATT_INFO_IND** (GAPC_DEV_CENTRAL_RPA) to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 274. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_CENTRAL_RPA

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_CENTRAL_RPA		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x15	GAPC_GET_PEER_CENTRAL_RPA
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x15 0x00		

5.3.25.10 GAPC_GET_PEER_RPA_ONLY

1. Step 1: The GTL Host sends the command to DA14585/531 to retrieve information about the peer's appearance.

Table 275. GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_RPA_ONLY example

Message stream - GAPC_GET_INFO_CMD with operation GAPC_GET_PEER_RPA_ONLY		
Symbolic message structure reference can be found in Table 245.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x0E	GAPC_GET_INFO_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
operation	0x16	GAPC_GET_PEER_RPA_ONLY
Message string		
0x05 0x05 0x0E 0x0E 0x00 0x10 0x00 0x01 0x00 0x16		

2. Step 2: The GTL Host receives GAPC_PEER_ATT_INFO_IND from DA14585/531.

Table 276. GAPC_PEER_ATT_INFO_IND (GAPC_DEV_RPA_ONLY) example

Message stream: GAPC_PEER_ATT_INFO_IND - GAPC_DEV_RPA_ONLY		
Symbolic message structure reference can be found in Table 248.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0E	GAPC_PEER_ATT_INFO_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC
PAR_LEN	0x0C 0x00	12 bytes
req	0x16	GAPC_DEV_RPA_ONLY
{padding}	0x00	Ignored. Any value is valid
handle	0x07 0x00	Attribute Handle = 0x07
info.rpa_only	0x01	Device Resolvable Private Address Only parameters
{padding}	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	{padding} Ignored. Any value is valid
Message string		
0x05 0x06 0x0E 0x10 0x00 0x0E 0x00 0x0C 0x00 0x16 0x00 0x07 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

3. Step 3: After the Bluetooth LE application has sent **GAPC_PEER_ATT_INFO_IND** (GAPC_DEV_CENTRAL_RPA) to the GTL Host, it also sends **GAPC_CMP_EVT** for **GAPC_GET_INFO_CMD.**

Table 277. GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_RPA_ONLY

Message stream: GAPC_CMP_EVT for GAPC_GET_INFO_CMD with operation of GAPC_GET_PEER_RPA_ONLY		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x16	GAPC_GET_PEER_RPA_ONLY
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x16 0x00		

5.3.26 Retrieve the Permissions for a Specific Handle

5.3.26.1 GATTM_ATT_GET_PERMISSION_REQ

The GLT host sends the message GATTM_ATT_GET_PERMISSION_REQ to DA14585/531 to retrieve the permissions for a specific handle.

Table 278. GATTM_ATT_GET_PERMISSION_REQ symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ATT_GET_PERMISSION_REQ	0x0B06
DST_ID	2	TASK_ID_GATTM	0x000B
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	2 bytes	0x0002
Message parameter			
handle	2	Handle of the attribute	

Table 279. GATTM_ATT_GET_PERMISSION_REQ example

Message stream (GATTM_ATT_GET_PERMISSION_REQ)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x06 0x0B	GATTM_ATT_GET_PERMISSION_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x02 0x00	2
handle	0x03 0x00	Handle 3
Message string		
0x05 0x06 0x0B 0x0B 0x00 0x10 0x00 0x02 0x00 0x03 0x00		

5.3.26.2 GATTM_ATT_GET_PERMISSION_RSP

After receiving GATT_ATT_GET_PERMISSION_REQ from the GTL Host, DA14585/531 sends GATTM_ATT_GET_PERMISSION_RSP.

Table 280. GATTM_ATT_GET_PERMISSION_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ATT_GET_PERMISSION_RSP	0x0B07
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTM	0x000B
PAR_LEN	2	12 bytes	0x000C
Message parameter			
handle	2	Handle of the attribute	
{padding}	2	{padding}	Ignored. Any value is valid.
perm	4	Attribute permission	
status	1	Return status	
{padding}	3	{padding}	Ignored. Any value is valid.

Table 281. GATTM_ATT_GET_PERMISSION_RSP example

Message stream (GATTM_ATT_GET_PERMISSION_RSP)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x07 0x0B	GATTM_ATT_GET_PERMISSION_RSP
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0C 0x00	12 bytes
handle	0x03 0x00	Handle 3
{padding}	0x00 0x00	{padding} Ignored. Any value is valid.
perm	0x01 0x00 0x00 0x00	0x00001
status	0x00	GAP_ERR_NO_ERROR
{padding}	0x00 0x00 0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x07 0x0B 0x0B 0x00 0x10 0x00 0x0C 0x00 0x03 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

5.3.27 Set the Permissions for a Specific Handle

5.3.27.1 GATTM_ATT_SET_PERMISSION_REQ

The external host sends the message GATTM_ATT_SET_PERMISSION_REQ to DA14585/531 to set the permissions for a specific handle.

Table 282. GATTM_ATT_SET_PERMISSION_REQ symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ATT_SET_PERMISSION_REQ	0x0B08
DST_ID	2	TASK_ID_GATTM	0x000B
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	8 bytes	0x0008
Message parameter			
handle	2	Handle of the attribute	0xXX 0xXX
{padding}	2	{padding}	Ignored. Any value is valid.
perm	4	Attribute permission	0xXX 0xXX 0xXX 0xXX

Table 283. GATTM_ATT_SET_PERMISSION_REQ example

Message stream (GATTM_ATT_SET_PERMISSION_REQ)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x08 0x0B	GATTM_ATT_SET_PERMISSION_REQ
DST_ID	0x0B 0x00	TASK_ID_GATTM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x08 0x00	8 bytes
handle	0x32 0x00	Handle 0x0032
{padding}	0x00 0x00	{padding} Ignored. Any value is valid.
permission	0x04 0x02 0x00 0x00	Permissions =0x00000204
Message string		
0x05 0x08 0x0B 0x0B 0x00 0x10 0x00 0x08 0x00 0x32 0x00 0x00 0x00 0x04 0x02 0x00 0x00		

5.3.27.2 GATTM_ATT_SET_PERMISSION_RSP

After receiving GATT_ATT_SET_PERMISSION_REQ from the GTL Host, DA14585/531 sends GATTM_ATT_SET_PERMISSION_RSP.

Table 284. GATTM_ATT_SET_PERMISSION_RSP symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTM_ATT_SET_PERMISSION_RSP	0x0B09
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GATTM	0x000B
PAR_LEN	2	4 bytes	0x0004
Message parameter			
handle	2	Handle of the attribute	
status	1	Return status	
{padding}	1	{padding}	Ignored. Any value is valid.

Table 285. GATTM_ATT_SET_PERMISSION_RSP example

Message stream (GATTM_ATT_SET_PERMISSION_RSP)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x09 0x0B	GATTM_ATT_SET_PERMISSION_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0B 0x00	TASK_ID_GATTM
PAR_LEN	0x04 0x00	4 bytes
handle	0x32 0x00	Handle 0x0032
status	0x00	GAP_ERR_NO_ERROR
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x09 0x0B 0x10 0x00 0x0B 0x00 0x04 0x00 0x32 0x00 0x00 0x00		

5.3.28 Register/Unregister Peer Device Events

The GTL Host sends **GATTC_REG_TO_PEER_EVT_CMD** to DA14585/531 to register or unregister from a peer device events such as indications or notifications on a specific service attribute handle range of a connection. This means that a peer executing this command accepts notifications/indications only within the range in the command.

Table 286. GATTC_REG_TO_PEER_EVT_CMD symbolic message structure

Parameters	**Bytes**	**Description**	**Hex data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GATTC_REG_TO_PEER_EVT_CMD	0x0C0F
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0C (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	8 bytes	0x0008
Message parameters			
operation	1	Request Type	**0x10**: GATTC_REGISTER **0x11**: GATTC_UNREGISTER
{padding}	1	{padding}	Ignored. Any value is valid.
seq_num	2	Operation sequence number	0xXX 0xXX
start_hdl	2	Attribute Start Handle	0xXX 0xXX
end_hdl	2	Attribute End Handle	0xXX 0xXX

1. Step 1: GTL Host sends GATTC_REG_TO_PEER_EVT_CMD (with the GATTC_REGISTER operation) to DA14585/531 to write to the characteristic's value handle.

Table 287. GATTC_REG_TO_PEER_EVT_CMD with operation GATTC_REGISTER – example

Message stream: GATTC_REG_TO_PEER_EVT_CMD with GATT_REGISTER operation		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x0F 0x0C	GATTC_REG_TO_PEER_EVT_CMD
DST_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x08 0x00	8 bytes
operation	0x10	GATTC_REGISTER
{padding}	0x00	{padding} Ignored. Any value is valid.
seq_num	0x0B 0x00	Sequence number 0x000B
start_hdl	0x01 0x00	Attribute Start Handle
end_hdl	0xFF 0xFF	Attribute End Handle
Message string		
0x05 0x0F 0x0C 0x0C 0x00 0x10 0x00 0x08 0x00 0x10 0x00 0x0B 0x00 0x01 0x00 0xFF 0xFF		

2. Step 2: DA14585/531 sends GATTC_CMP_EVT when the procedure is completed.

Table 288. GATTC_CMP_EVT after a GATTC_REG_TO_PEER_EVT_CMD with operation GATTC_REGISTER

Message stream (GATTC_CMP_EVT) on completion of GATTC_READ_CMD		
Symbolic message structure reference can be found in Table 53.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0C	GATTC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0C 0x00	TASK_ID_GATTC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
operation	0x10	GATTC_REGISTER
status	0x00	GATT_ERR_NO_ERROR
seq_num	0x0B 0x00	Sequence number 0x000B
Message string		
0x05 0x00 0x0C 0x10 0x00 0x0C 0x00 0x04 0x00 0x10 0x00 0x0B 0x00		

5.3.29 Set the LE Data Length

5.3.29.1 GAPC_SET_LE_PKT_SIZE_CMD

This command sets the LE data length. When this command is successfully executed, the indication **GAPC_LE_PKT_SIZE_IND** is sent by the Bluetooth LE application to the GTL Host to indicate the new LE data length and data time.

Table 289. GAPC_SET_LE_PKT_SIZE_CMD symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_SET_LE_PKT_SIZE_CMD	0x0E2B
DST_ID	2	nn, TASK_ID_GATTC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	6 bytes	0x0006
Message parameters			
operation	1	Set the LE Data length value	GAPC_SET_LE_PKT_SIZE
{padding}	1	{padding}	Ignored. Any value is valid.
tx_octets	2	Preferred maximum number of payload octets the local Controller should include in a single Link Layer Data Channel PDU	0xXX 0xXX
tx_time	2	Preferred maximum number of microseconds the local Controller should use to transmit a single Link Layer Data Channel PDU	0xXX 0xXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

1. Step 1: GTL Host sends **GAPC_SET_LE_PKT_SIZE_CMD** to DA14585/531 to set the LE data length.

Table 290. GAPC_SET_LE_PKT_SIZE_CMD – example

Message stream: GAPC_SET_LE_PKT_SIZE_CMD		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x2B 0x0E	GAPC_SET_LE_PKT_SIZE_CMD
DST_ID	0x0E 0x00	TASK_ID_GATTC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x06 0x00	6 bytes
operation	0x14	GAPC_SET_LE_PKT_SIZE (See Table 372)
{padding}	0x00	{padding} Ignored. Any value is valid.
tx_octets	0xFB 0x00	Preferred maximum number of payload octets that the local Controller should include in a single Link Layer Data Channel PDU.
tx_time	0x48 0x08	Preferred maximum number of microseconds that the local Controller should use to transmit a single Link Layer Data Channel PDU
Message string		
0x05 0x2B 0x0E 0x0E 0x00 0x10 0x00 0x06 0x00 0x14 0x00 0xFB 0x00 0x48 0x08		

5.3.29.2 GAPC_LE_PKT_SIZE_IND

This command indicates the event triggered when LE data length/time has been changed.

2. Step 2: GTL Host receives **GAPC_LE_PKT_SIZE_IND**.

Table 291. GAPC_LE_PKT_SIZE_IND symbolic message structure

Parameters	Bytes	Description	Hex data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LE_PKT_SIZE_IND	0x0E2C
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000E
PAR_LEN	2	8 bytes	0x0008
Message parameters			
max_tx_octets	2	The maximum number of payload octets in TX	
max_tx_time	2	The maximum time that the local Controller takes to transmit	
max_rx_octets	2	The maximum number of payload octets in RX	
max_rx_time	2	The maximum time that the local Controller takes to receive	

Table 292. GAPC_LE_PKT_SIZE_IND example

Message stream: GAPC_LE_PKT_SIZE_IND		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x2C 0x0E	GAPC_LE_PKT_SIZE_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x08 0x00	8 bytes
max_tx_octets	0xFB 0x00	251 octets
max_tx_time	0x48 0x08	2120 ms
max_rx_octets	0xFB 0x00	251 octets
max_rx_time	0x48 0x08	2120 ms
Message string		
0x05 0x2C 0x0E 0x10 0x00 0x0E 0x00 0x08 0x00 0xFB 0x00 0x48 0x08 0xFB 0x00 0x48 0x08		

3. Step 3: **GAPC_CMP_EVT** with an operation of **GAPC_SET_LE_PKT_SIZE** is sent by the Bluetooth LE application to the GTL Host marking the completion of the **GAPC_SET_LE_PKT_SIZE_CMD** operation.

Table 293. GAPC_CMP_EVT - GAPC_SET_LE_PKT_SIZE_CMD - example

Message stream - GAPC_CMP_EVT of GAPC_SET_LE_PKT_SIZE_CMD operation		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x14	GAPC_SET_LE_PKT_SIZE
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x14 0x00		

5.3.30 LE Credit Based Connections

5.3.30.1 GAPC_LECB_CREATE_CMD

This command creates a LE Credit based channel for a specific LE Protocol/Service Multiplexer on the server side. Some parameters must be within a designated range in order not to receive a GAP_ERR_COMMAND_DISALLOWED from the device. For example, the channel ID (cid) should be in the range for a dynamically allocated channel (0x0040 - 0x007F). Security level (SEC_LVL) parameter is used to indicate the security level of the connection shown in Figure 30.

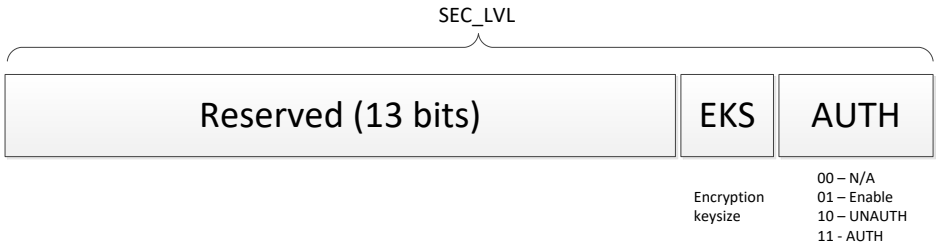


Figure 30. SEC_LEV parameter structure

Table 294. GAPC_LECB_CREATE_CMD symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_CREATE_CMD	0x0E1E
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E, nn =conidx
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	10 bytes	0x000A
Message parameters			
operation	1	GAPC_LE_CB_CREATE	0x0E
pkt_id	1	Internal parameter used to manage internally l2cap packet identifier	
{padding}	2	{padding} Ignored	Any value is valid
le_psm	2	LE Protocol/Service Multiplexer	

Parameters	Bytes	Description	Data
Header			
cid	2	Channel identifier	
credit	2	Credit allocated for the LE Credit Based Connection	

Table 295. GAPC_LECB_CREATE_CMD example

Message stream: GAPC_LECB_CREATE_CMD		
Symbolic Message Structure reference can be found in Table 294.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1E 0x0E	GAPC_LECB_CREATE_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	10 bytes
operation	0x0D	GAPC_LE_CB_CREATE
pkt_id	0x00	First packet
{padding}	0x00 0x00	{padding} Ignored. Any value is valid.
le_psm	0x80 0x00	
cid	0x40 0x00	Channel identifier is 0x40
credit	0x06 00	6 credits initially available
Message string		
0x05 0x1E 0x0E 0x0E 0x00 0x10 0x00 0x0A 0x00 0x0D 0x00 0x00 0x00 0x80 0x00 0x40 0x00 0x06 00		

The Bluetooth LE application sends GAPC_CMP_EVT to the GTL Host when the operation completes.

Table 296. GAPC_CMP_EVT for GAPC_LECB_CREATE_CMD - example

Message stream - GAPC_CMP_EVT for GAPC_LECB_CREATE_CMD operation		
Symbolic message structure reference can be found in Table 52.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x0D	GAPC_LE_CB_CREATE
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x0D 0x00		

5.3.30.2 GAPC_LECB_DESTROY_CMD

This is a command to destroy a non-connected LE Credit Based channel. To destroy connected channels, use the normal disconnection procedure.

Table 297. GAPC_LECB_DESTROY_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_DESTROY_CMD	0x0E1F
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0x000E, nn =conidx
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	4 bytes	0x0004
Message parameters			
operation	1	GAPC_LE_CB_DESTROY	0x0E
{padding}	2	{padding} Ignored	Any value is valid
le_psm	2	LE Protocol/Service Multiplexer	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 298. GAPC_LECB_DESTROY_CMD example

Message stream: GAPC_LECB_DESTROY_CMD		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1F 0x0E	GAPC_LECB_DESTROY_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x04 0x00	4 bytes
operation	0x0E	GAPC_LE_CB_DESTROY
{padding}	0x00	{padding} Ignored. Any value is valid.
le_psm	0x80 0x00	
Message string		
0x05 0x1F 0x0E 0x0E 0x00 0x10 0x00 0x04 0x00 0x0E 0x00 0x80 0x00		

The Bluetooth LE application sends GAPC_CMP_EVT to the GTL Host when the operation completes.

Table 299. GAPC_CMP_EVT for GAPC_LECB_DESTROY_CMD - example

Message stream - GAPC_CMP_EVT of GAPC_LECB_DESTROY_CMD operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x0E	GAPC_LECB_DESTROY
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x0E 0x00		

5.3.30.3 LECB Connection Procedure

Figure 31 shows an abstract representation of the exchange of commands during an LECB connection for two devices that both have a GTL host setup.

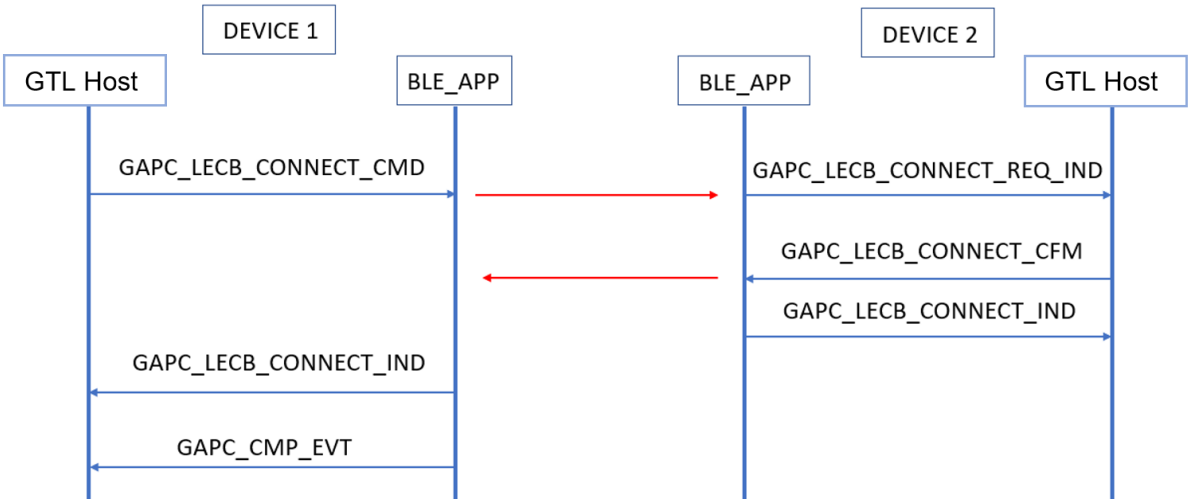


Figure 31. LECB connection procedure

GAPC_LECB_CONNECT_CMD

This is the command that connects to a previously created LE Credit-based channel for specific LE Protocol/Service Multiplexer. Note that the cid parameter is local and could be different from the one used in this Credit-Based Connection creation command.

Table 300. GAPC_LECB_CONNECT_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_CONNECT_CMD	0x0E20
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	8 bytes	8 bytes
Message parameters			
operation	1	GAPC_LE_CB_CONNECTION: LE Credit-Based connection	0x0F
pkt_id	1	Internal parameter used to manage L2CAP packet identifier	
le_psm	2	LE Protocol/Service Multiplexer	
cid	2	Channel identifier	
credit	2	Credit allocated for the LE Credit Based Connection	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 301. GAPC_LECB_CONNECT_CMD example

Message stream: GAPC_LECB_CONNECT_CMD		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x20 0x0E	GAPC_LECB_CONNECT_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x08 0x00	8 bytes
operation	0x0F	GAPC_LE_CB_CONNECTION
pkt_id	0x0F	Dummy Value
le_psm	0x80 0x00	PSM for both devices
cid	0x40 0x00	Local channel identifier
credit	0x06 0x00	Credits upon connection
Message string		
0x05 0x20 0x0E 0x0E 0x00 0x10 0x00 0x08 0x00 0x0F 0x0F 0x80 0x00 0x40 0x00 0x06 0x00		

The Bluetooth LE application (on both devices) responds with GAPC_LECB_CONNECT_IND when the connection is established and with GAPC_CMP_EVT when the operation completes on the side of the initiator of the connection.

GAPC_LECB_CONNECT_IND

Table 302. GAPC_LECB_CONNECT_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_CONNECT_IND	0x0E22
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	8 bytes	0x0008
Message parameters			
le_psm	2	LE Protocol/Service Multiplexer	
dest_credit	2	Destination Credit for the LE Credit Based Connection	
max_sdu	2	Maximum SDU size	
dest_cid	2	Destination CID	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 303. GAPC_LECB_CONNECT_IND - example

Message stream - GAPC_LECB_CONNECT_IND		
Symbolic message structure reference can be found in Table 302.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x22 0x0E	GAPC_LECB_CONNECT_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x08 0x00	8 bytes
le_psm	0x80 0x00	PSM chosen for connection
dest_credit	0x06 0x00	Credits
max_sdu	0xF7 0x00	247 bytes max sdu
dest_cid	0x40 0x00	CID chosen
Message string		
0x05 0x22 0x0E 0x10 0x00 0x0E 0x00 0x08 0x00 0x80 0x00 0x06 0x00 0xF7 0x00 0x40 0x00		

Table 304. GAPC_CMP_EVT for GAPC_LECB_CONNECT_CMD - example

Message stream - GAPC_CMP_EVT of GAPC_LECB_CONNECT_CMD operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx =0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x0F	GAPC_LE_CB_CONNECTION
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x0F 0x00		

GAPC_LECB_CONNECT_REQ_IND

This message indicates to the GTL Host that a peer device requests a LE Credit connection to be established.

Table 305. GAPC_LECB_CONNECT_REQ_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_CONNECT_REQ_IND	0x0E21
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	8 bytes	0x0008

Parameters	Bytes	Description	Data
Header			
Message parameters			
le_psm	2	LE Protocol/Service Multiplexer	
dest_credit	2	Destination Credit for the LE Credit Based Connection	
max_sdu	2	Maximum SDU size	
dest_cid	2	Destination CID	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 306. GAPC_LECB_CONNECT_REQ_IND example

Message stream: GAPC_LECB_CONNECT_REQ_IND		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x21 0x0E	GAPC_LECB_CONNECT_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x08 0x00	8 bytes
le_psm	0x08 0x00	PSM chosen for connection
dest_credit	0x06 0x00	Credits
max_sdu	0xF7 0x00	247 bytes max sdu
dest_cid	0x40 0x00	CID chosen
Message string		
0x05 0x21 0x0E 0x10 0x00 0x0E 0x00 0x08 0x00 0x08 0x00 0x06 0x00 0xF7 0x00 0x40 0x00		

When the GTL Host has received GAPC_LECB_CONNECT_REQ_IND, it sends GAPC_LECB_CONNECT_CFM to confirm whether a connection can be established or not.

GAPC_LECB_CONNECT_CFM

This message is the response of the GTL Host to GAPC_LECB_CONNECT_REQ_IND for accepting or rejecting an incoming LE Credit-Based connection for specific LE Protocol/Service Multiplexer. The parameter "status" reflects the result of the operation and should be filled according to Table 307.

Table 307. GAPC_LECB_CONNECT_CFM symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_CONNECT_CFM	0x0E23
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	4 bytes	0x0004
Message parameters			
le_psm	2	LE Protocol/Service Multiplexer	
status	2	Status of operation	See Table 309

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 308. GAPC_LECB_CONNECT_CFM example

Message stream: GAPC_LECB_CONNECT_CFM		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x23 0x0E	GAPC_LECB_CONNECT_CFM
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x04 0x00	8 bytes
le_psm	0x80 0x00	PSM chosen for connection
status	0x00 0x00	Status (See Table 309)
Message string		
0x05 0x23 0x0E 0x0E 0x00 0x10 0x00 0x04 0x00 0x80 0x00 0x00 0x00		

Table 309. LECB connection status

Value	Description
0x0000	Connection successful
0x0001	Reserved
0x0002	Connection refused – LE_PSM not supported
0x0003	Reserved
0x0004	Connection refused – no resources available
0x0005	Connection refused – insufficient authentication
0x0006	Connection refused – insufficient authorization
0x0007	Connection refused – insufficient encryption key size
0x0008	Connection refused – insufficient encryption
0x0009 – 0xffff	Reserved

5.3.30.4 LECB Add Credits Procedure

Figure 32 shows an abstract representation of the exchange of commands during adding LECB credits, for two devices that both have a GTL host setup.

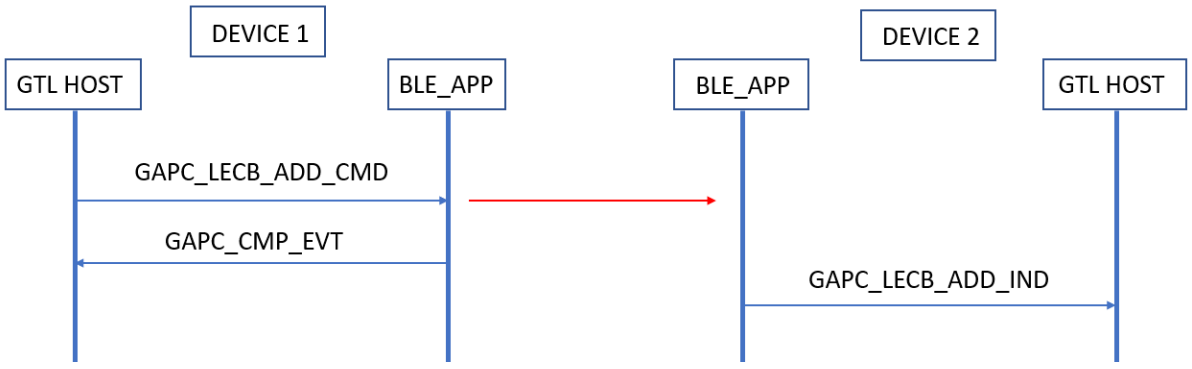


Figure 32. LECB add credits procedure

GAPC_LECB_ADD_CMD

This command informs the peer device that new credits are available for a LE Credit connection.

Table 310. GAPC_LECB_ADD_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_ADD_CMD	0x0E24
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	6 bytes	0x0006
Message parameters			
operation	1	GAPC_LE_CB_ADDITION: Add credit to receive packets from peer device	0x11
pkt_id	1	Internal parameter used to manage internally l2cap packet identifier for signaling	
le_psm	2	LE Protocol/Service Multiplexer	
credit	2	RX credit for the LE credit-based connection	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 311. GAPC_LECB_ADD_CMD example

Message stream: GAPC_LECB_ADD_CMD		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x24 0x0E	GAPC_LECB_ADD_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x06 0x00	6 bytes
operation	0x11	GAPC_LE_CB_ADDITION
pkt_id	0x0F	Dummy Value in this example
le_psm	0x80 0x00	PSM chosen for connection
credit	0x0A 0x00	10 credits added
Message string		
0x05 0x24 0x0E 0x0E 0x00 0x10 0x00 0x06 0x00 0x11 0x0F 0x80 0x00 0x0A 0x00		

The Bluetooth LE application sends GAPC_CMP_EVT to the GTL Host when the operation completes.

Table 312. GAPC_CMP_EVT for GAPC_LECB_ADD_CMD- example

Message stream - GAPC_CMP_EVT of GAPC_LECB_ADD_CMD operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x11	GAPC_LE_CB_ADDITION
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x11 0x00		

GAPC_LECB_ADD_IND

This command informs External Host that the peer device can receive additional LE-Frames for an LE Credit connection.

Table 313. GAPC_LECB_ADD_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_ADD_IND	0x0E25
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	6 bytes	0x0006
Message parameters			
le_psm	2	LE Protocol/Service Multiplexer	
src_credit	2	RX credit for the LE credit-based connection	
dest_credit	2	TX credit for the LE credit-based connection	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 314. GAPC_LECB_ADD_IND - example

Message stream - GAPC_LECB_ADD_IND		
Symbolic message structure reference can be found in Table 313		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x25 0x0E	GAPC_LECB_ADD_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x06 0x00	6 bytes
le_psm	0x80 0x00	PSM chosen for connection
src_credit	0x06 0x00	RX credit for the LE credit-based connection
dest_credit	0x10 0x00	TX credit for the LE credit-based connection
Message string		
0x05 0x25 0x0E 0x0E 0x00 0x10 0x00 0x06 0x00 0x80 0x00 0x06 0x00 0x10 0x00		

5.3.30.5 LECB Disconnection Procedure

Figure 33 shows an abstract representation of the exchange of commands during an LECB disconnection for two devices that both have a GTL host setup.

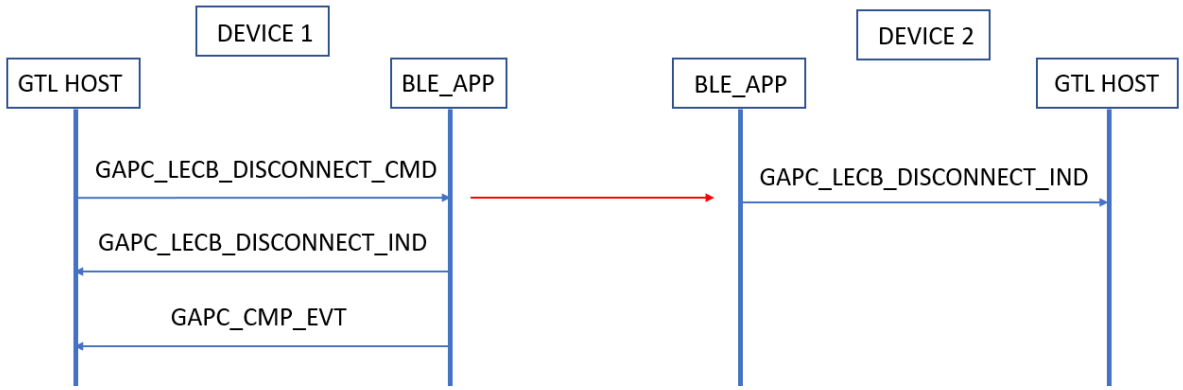


Figure 33. LECB disconnection procedure

GAPC_LECB_DISCONNECT_CMD

This command breaks an LE Credit Based connection between the local device and a peer device.

Table 315. GAPC_LECB_DISCONNECT_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_DISCONNECT_CMD	0x0E26
DST_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	4 bytes	0x0004
Message parameters			
operation	1	GAPC_LE_CB_DISCONNECTION: LE Credit disconnection	0x10
pkt_id	1	Internal parameter used to manage internally l2cap packet identifier	
le_psm	2	LE Protocol/Service Multiplexer	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 316. GAPC_LECB_DISCONNECT_CMD example

Message stream: GAPC_LECB_ DISCONNECT_CMD		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x26 0x0E	GAPC_LECB_DISCONNECT_CMD
DST_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x04 0x00	4 bytes
operation	0x10	GAPC_LE_CB_DISCONNECTION
pkt_id	0x0F	Dummy Value
le_psm	0x80 0x00	PSM chosen for connection
Message string		
0x05 0x26 0x0E 0x0E 0x00 0x10 0x00 0x04 0x00 0x10 0x0F 0x80 0x00		

The Bluetooth LE application (on both devices) responds with **GAPC_LECB_DISCONNECT_IND** when the connection is terminated and with **GAPC_CMP_EVT** when the operation completes on the side of the initiator of the disconnection.

GAPC_LECB_DISCONNECT_IND

This disconnection event is triggered when a LE Credit connection is terminated. The Bluetooth LE application on both devices sends this message to the GTL Host.

Table 317. GAPC_LECB_DISCONNECT_IND symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPC_LECB_DISCONNECT_IND	0x0E27
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_GAPC (Note 1)	0xnn0E (nn = conidx)
PAR_LEN	2	4 bytes	0x0004
Message parameters			
le_psm	2	LE Protocol/Service Multiplexer	
reason	2	Reason for disconnection (see Table 320)	

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 318. GAPC_LECB_DISCONNECT_IND - example

Message stream - GAPC_LECB_CONNECT_IND		
Symbolic message structure reference can be found in Table 302.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x27 0x0E	GAPC_LECB_DISCONNECT_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x04 0x00	4 bytes
le_psm	0x80 0x00	PSM chosen for connection
reason	0x02 0x00	Local device initiates disconnection
Message string		
0x05 0x27 0x0E 0x10 0x00 0x0E 0x00 0x04 0x00 0x80 0x00 0x02 0x00		

Table 319. GAPC_CMP_EVT for GAPC_LECB_DISCONNECT_CMD- example

Message stream - GAPC_CMP_EVT of GAPC_LECB_DISCONNECT_CMD operation		
Symbolic message structure reference can be found in Table 51.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0E	GAPC_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0E 0x00	TASK_ID_GAPC, conidx = 0x00
PAR_LEN	0x02 0x00	2 bytes
operation	0x10	GAPC_LE_CB_DISCONNECTION
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0E 0x10 0x00 0x0E 0x00 0x02 0x00 0x10 0x00		

Table 320. Valid disconnection reasons for LE credit-based connection

Value	**Flag**	**Description**
0x02	L2C_CB_DISC_BY_LOCAL_DEV	Local device initiates disconnection (disconnection requested by application)
0x03	L2C_CB_DISC_BY_REMOTE_DEV	Remote device initiates disconnection
0x04	L2C_CB_DISC_CREDIT_ERROR	No more credits available
0x05	L2C_CB_DISC_INVALID_MTU_SIZE	Invalid MTU (MTU exceeded)
0x06	L2C_CB_DISC_INVALID_MPS_SIZE	Invalid Packet size (MPS exceeded)
0x07	L2C_CB_DISC_INVALID_SDU	Invalid received SDU (SDU size invalid for instance)

5.3.31 Description of Other API Commands, Responses, and Indications

5.3.31.1 GAPM_RAL_SIZE_IND

This indication is sent by the Bluetooth LE application to the GTL Host after the resolving list size has been requested.

Table 321. GAPM_RAL_SIZE_IND parameters

Parameters	Bytes	Description
size	1	Resolving list size

5.3.31.2 GAPM_RAL_ADDR_IND

This command indicates the event triggered when a local or peer Resolvable Private Address (RPA) is requested for a given Identity Address. An RPA may be resolved if the corresponding device's IRK is available. If an RPA is resolved, the device can associate this address with the peer device.

Table 322. GAPM_RAL_ADDR_IND parameters

Parameters	Bytes	Description
operation	1	Resolving list operation ▪ GAPM_GET_RAL_LOC_ADDR: Get the local Resolvable Private Address used for a given peer Identity Address ▪ GAPM_GET_RAL_PEER_ADDR: Get the peer Resolvable Private Address used for a given peer Identity Address
addr[BD_ADDR_LEN]	1	Current peer's Resolvable Private Address

5.3.31.3 GAPM_USE_P256_BLOCK_CMD

It is a command used by the application to generate the DH key required by the pairing procedure so that the application can support Secure Connections.

This command triggers session key generation which involves running EC point multiplication function by using the public key from the peer device and the private key of the local device. This command is called by the security manager when it receives the public key from the peer device. When a session key is generated, the indication `GAPM_USE_P256_BLOCK_IND` is generated.

Table 323. GAPM_USE_P256_BLOCK_CMD parameters and its response

Parameters	Bytes	Description
operation	1	Command Operation Code (must be GAPM_USE_P256_BLOCK).
operand_1[ECDH_KEY_LEN × 2]	1	Peer EC based public key to be used.
Response		
▪ GAPM_USE_P256_BLOCK_IND (see Section 5.3.31.4): Event triggered when P256 DH key is generated. ▪ GAPM_CMP_EVT (see Table 51): When operation completed.		

5.3.31.4 GAPM_USE_P256_BLOCK_IND

This command indicates the event triggered when the DH key has been generated.

Table 324. GAPM_USE_P256_BLOCK_IND

Parameters	Bytes	Description
result [ECDH_KEY_LEN]	1	Generated session key (x parameter of common secret).

5.3.31.5 GAPC_KEYPRESS_NOTIFICATION/GAPC_KEYPRESS_NOTIFICATION_IND/GAPC_KEYPRESS_NOTIFICATION_CMD

This event informs the device that a keypress event has happened at the connected peer.

Table 325. GAPC_KEYPRESS_NOTIFICATION parameter

Parameters	Type	Description
type	1	Type of keypress (see Table 326)

5.3.31.5.1 Keypress Codes

Table 326. GAPC keypress notification types

Value	Flag	Description
0x00	KEY_NOTIFICATION_ENTRY_START	Key entry started
0x01	KEY_NOTIFICATION_DIGIT_ENTERED	Key digit entered
0x02	KEY_NOTIFICATION_DIGIT_ERASED	Key digit erased
0x03	KEY_NOTIFICATION_CLEAR	Key cleared
0x04	KEY_NOTIFICATION_ENTRY_COMPLETE	Key entry completed
0x05	KEY_NOTIFICATION_MAX	Max number of digits entered

GAPC in Bluetooth LE application uses request flag operations embedded in the interface message sent to the GAP controller to ensure correct handling of the operation requests from the application.

6. GATT Services

6.1 Allocate a Task for a Specific Standard DA14585/531 Profile

1. Step 1: Allocate a specific profile task.

   The GTL Host first sends the GAPM_SET_DEV_CONFIG_CMD command, and then sends the GAPM_PROFILE_TASK_ADD_CMD command to the Bluetooth LE application, to allocate a task for a specific profile in the DA14585/531 database (service or client). During the execution of GAPM_PROFILE_TASK_ADD_CMD, attribute database for this profile as well as the environment variables required are allocated in DA14585/531 by the Bluetooth LE application.

IMPORTANT NOTE
Allocated profiles are freed if GAPM_RESET_CMD or GAPM_SET_DEV_CONFIG_CMD is executed in the Bluetooth LE application.

GAPM_PROFILE_TASK_ADD_CMD supports only services that are implemented in the DA14585/531 software.

Table 327. GAPM_PROFILE_TASK_ADD_CMD symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_PROFILE_TASK_ADD_CMD	0x0D1B
DST_ID	2	TASK_ID_GAPM	0x000D
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	8 + size of param (see Message Parameters in this table)	
Message parameters			
operation	1	GAP requested operation	GAPM_PROFILE_TASK_ADD: add new profile task
sec_lvl	1	7 6 5 4 3 2 1 0 {reserved} AUTH EKS MI	▪ **MI**: • 1: Application task is a multi-instantiated task • 0: Mono-Instantiated application task ▪ **EKS**: service needs a 16 bytes encryption key (only applies for service; ignored by collectors) ▪ **AUTH**: • 0: Disable • 1: Enable • 2: Unauth (unauthenticated) • 3: Auth (authenticated)
prf_task_id	2	Profile task identifier of profile to add	
app_task	2	Application task identifier	Application task number that manages the reception of events
start_hdl	2	Service start handle	Only applies for services Ignored by collectors 0: dynamically allocated in attribute database
param			Values to initialize profile (database parameters and others)

2. Step 2: The Bluetooth LE application confirms the allocation of a specific profile task.

The Bluetooth LE application responds to GAPM_PROFILE_TASK_ADD_CMD by sending the GAPM_PROFILE_ADDED_IND message to the GTL Host.

Table 328. GAPM_PROFILE_ADDED_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	GAPM_PROFILE_ADDED_IND	0x0D1C
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_GAPM	0x000D
PAR_LEN	2	6 bytes	0x00
Message parameters			
prf_task_id	2	Profile task identifier	
prf_task_nk	2	Profile task number allocated	
start_hdl	2	Service start handle	

For an example of **GAPM_PROFILE_TASK_ADD_CMD,** see Table 327. For an example of **GAPM_PROFILE_ADDED_IND,** see Table 328.

6.2 Device Information Service Server (DISS)

6.2.1 DISS Profile Task Allocation

1. Step 1: DA14585/531 allocates a task for the DISS profile.

   The GTL Host sends the GAPM_PROFILE_TASK_ADD_CMD command to the Bluetooth LE application for DA14585/531 to allocate a task for the DISS profile and to add an instance of the Device Information Service to the DA14585/531 GATT database. The symbolic message structure reference of the GAPM_PROFILE_TASK_ADD_CMD command can be found in Table 327.

Table 329. GAPM_PROFILE_TASK_ADD_CMD example

Message stream - GAPM_PROFILE_TASK_ADD_CMD		
Symbolic message structure reference can be found in Table 327.		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1B 0D	GAPM_PROFILE_TASK_ADD_CMD
DST_ID	0x0D 0x00	TASK_ID_GAPM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	10 bytes
operation	0x1B	GAPM_PROFILE_TASK_ADD
sec_lvl	0x04	AUTH: Enable
prf_task_id	0x14 0x00	0x0014: TASK_ID_DISS
app_task	0x10 0x00	0x0010: TASK_ID_GTL
start_hdl	0x0000	0: dynamically allocated in Attribute database
param (features)	0x73 0x01	0x0173: the following features are enabled: DIS_MANUFACTURER_NAME_CHAR_SUP DIS_MODEL_NB_STR_CHAR_SUP DIS_SYSTEM_ID_CHAR_SUP DIS_SW_REV_STR_CHAR_SUP DIS_FIRM_REV_STR_CHAR_SUP DIS_PNP_ID_CHAR_SUP
Message string		
0x05 0x1B 0x0D 0x0D 0x00 0x10 0x00 0x0A 0x00 0x1B 0x04 0x14 0x00 0x10 0x00 0x00 0x00 0x73 0x01		

2. Step 2: The Bluetooth LE application confirms the allocation of the DISS profile task.

   The Bluetooth LE application responds to **GAPM_PROFILE_TASK_ADD_CMD** by sending the **GAPM_PROFILE_ADDED_IND** message to the GTL Host. The symbolic message structure reference of **GAPM_PROFILE_ADDED_IND** can be found in Table 328.

Table 330. GAPM_PROFILE_ADDED_IND example

Message stream - GAPM_PROFILE_ADDED_IND		
Symbolic message structure reference can be found in Table 328		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x1C 0x0D	GAPM_PROFILE_ADDED_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x06 0x00	6 bytes
prf_task_id	0x14 0x00	0x0014: TASK_ID_DISS
prf_task_nb	0x10 0x00	0x0010: task number 16
start_hdl	0x0A 0x00	0x000A: handle: 10
Message string		
0x05 0x1C 0x0D 0x10 0x00 0x0D 0x00 0x06 0x00 0x14 0x00 0x10 0x00 0x0A 0x00		

3. Step 3: Completion event message for the allocation of the DISS profile task.

   The Bluetooth LE application also responds to GAPM_PROFILE_TASK_ADD_CMD by sending the GAPM_CMP_EVT message to the GTL Host.

Table 331. GAPM_CMP_EVT for GAPM_PROFILE_TASK_ADD example

Message stream - GAPM_CMP_EVT for GAPM_PROFILE_TASK_ADD		
Symbolic message structure reference can be found in Table 51		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x0D	GAPM_CMP_EVT
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x0D 0x00	TASK_ID_GAPM
PAR_LEN	0x02 0x00	2 bytes
operation	0x1B	GAPM_PROFILE_TASK_ADD
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x00 0x0D 0x10 0x00 0x0D 0x00 0x02 0x00 0x1B 0x00		

6.2.2 Setting DISS Characteristic Values

1. Step 1: GTL Host sets the DISS characteristic values.

   The GTL Host sends the DISS_SET_VALUE_REQ command to DA14585/531 to set the DISS characteristic values. It is suggested to set the DISS characteristic values before DA14585/531 is connected to a peer device, although the values can be set when a peer GATT client device has been connected.

Table 332. DISS_SET_VAL_REQ symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05

Parameters	Bytes	Description	Data
MSG_ID	2	DISS_SET_VAL_REQ	0x1400
DST_ID	2	TASK_ID_DISS	0x0014
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	Length + 2	
Message parameters			
value	1	Characteristic code	
length	1	Value length	
data	Depending on value length	Value	

Table 333. DISS_SET_VAL_REQ example

Message stream: example for DIS characteristic "manufacturer name" (DIS_MANUFACTURER_NAME_CHAR)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x14	DISS_SET_VAL_REQ
DST_ID	0x14 0x00	TASK_ID_DISS
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0F 0x00	16 bytes
value	0x00	0x00: DIS_MANUFACTURER_NAME_CHAR
length	0x0B	The data for the value to be set is 12 bytes long
data	0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x53 0x65 0x6D 0x69 0x00	
{padding}	0x00	{padding} Ignored. Any value is valid.
Message string		
0x05 0x00 0x14 0x14 0x00 0x10 0x00 0x0F 0x00 0x00 0x0B 0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x53 0x65 0x6D 0x69 0x00 0x00		

2. Step 2: The Bluetooth LE application confirms the characteristic values have been set.

   The Bluetooth LE application responds to the DISS_SET_VALUE_REQ command by sending the DISS_SET_VALUE_RSP indication.

Table 334. DISS_SET_VALUE_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	DISS_SET_VALUE_RSP	0x1401
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_DISS	0x0014
PAR_LEN	2	2 bytes	0x0002
Message parameters			
value	1	Value set	**0x00**: DIS_MANUFACTURER_NAME_CHAR **0x01**: DIS_MODEL_NB_STR_CHAR **0x02**: DIS_SERIAL_NB_STR_CHAR **0x03**: DIS_HARD_REV_STR_CHAR **0x04**: DIS_FIRM_REV_STR_CHAR

Parameters	Bytes	Description	Data
			0x05: DIS_SW_REV_STR_CHAR **0x06**: DIS_SYSTEM_ID_CHAR **0x07**: DIS_IEEE_CHAR **0x08**: DIS_PNP_ID_CHAR
status	1	Status of the request	**0x00**: GAP_ERR_NO_ERROR **0x81**: PRF_ERR_INVALID_PARAM **0x82**: PRF_ERR_INEXISTENT_HDL **0x8D**: PRF_ERR_UNEXPECTED_LEN

Table 335. DISS_SET_VALUE_RSP example

Message stream: example for DIS_MANUFACTURER_NAME_CHAR		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x14	DISS_SET_VALUE_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x14 0x00	TASK_ID_DISS
PAR_LEN	0x02 0x00	2 bytes
value	0x00	0x00: DIS_MANUFACTURER_NAME_CHAR
status	0x00	0x00: GAP_ERR_NO_ERROR
Message string		
0x05 0x01 0x14 0x10 0x00 0x14 0x00 0x02 0x00 0x00 0x00		

6.2.3 Reading DISS Characteristic Values

The peer device sends a request to read a profile attribute value. The GTL Host receives a read indication for the DISS characteristic values.

1. Step 1: The Bluetooth LE application sends the **DISS_VALUE_REQ_IND** indication to the GTL Host to read a DISS characteristic value.

Table 336. DISS_VALUE_REQ_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	DISS_VALUE_REQ_IND	0x1402
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_DISS	0x0014
PAR_LEN	2	1 byte	0x0001
Message parameters			
value	1	Request Value	**0x00**: DIS_MANUFACTURER_NAME_CHAR **0x01**: DIS_MODEL_NB_STR_CHAR **0x02**: DIS_SERIAL_NB_STR_CHAR **0x03**: DIS_HARD_REV_STR_CHAR **0x04**: DIS_FIRM_REV_STR_CHAR **0x05**: DIS_SW_REV_STR_CHAR **0x06**: DIS_SYSTEM_ID_CHAR

Parameters	Bytes	Description	Data
			0x07: DIS_IEEE_CHAR **0x08**: DIS_PNP_ID_CHAR

Table 337. DISS_VALUE_REQ_IND example

Message stream: DISS_VALUE_REQ_IND example for DIS_PNP_ID_CHAR		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x02 0x14	DISS_VALUE_REQ_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x14 0x00	TASK_ID_DISS
PAR_LEN	0x01 0x00	1 byte
value	0x08	**0x08**: DIS_PNP_ID_CHAR
Message string		
0x05 0x02 0x14 0x10 0x00 0x14 0x00 0x01 0x00 0x08		

2. Step 2: The GTL Host sends the characteristic value to DA14585/531.

   The GTL Host responds to DA14585/531 by sending the **DISS_VALUE_CFM** message that contains the value of the requested DISS characteristic.

Table 338. DISS_VALUE_CFM symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	DISS_VALUE_CFM	0x1403
DST_ID	2	TASK_ID_DISS	0x0014
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	{depends on the data}	0xXXXX
Message parameters			
value	1	Requested value	**0x00**: DIS_MANUFACTURER_NAME_CHAR **0x01**: DIS_MODEL_NB_STR_CHAR **0x02**: DIS_SERIAL_NB_STR_CHAR **0x03**: DIS_HARD_REV_STR_CHAR **0x04**: DIS_FIRM_REV_STR_CHAR **0x05**: DIS_SW_REV_STR_CHAR **0x06**: DIS_SYSTEM_ID_CHAR **0x07**: DIS_IEEE_CHAR **0x08**: DIS_PNP_ID_CHAR
length	1	Value Length	
data	{length}	Value Data	

Table 339. DISS_VALUE_CFM example

Message stream: DISS_VALUE_CFM example for DIS_PNP_ID_CHAR		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x14	DISS_VALUE_CFM
DST_ID	0x14 0x00	TASK_ID_DISS
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x0A 0x00	10 bytes
value	0x08	0x08: DIS_PNP_ID_CHAR
length	0x07	Value Length
data	0x01 0xD2 0x00 0x80 0x05 0x00 0x01	
{padding}	0x00	{padding} Ignored. Any value is valid
Message string		
0x05 0x03 0x14 0x14 0x00 0x10 0x00 0x0A 0x00 0x08 0x07 0x01 0xD2 0x00 0x80 0x05 0x00 0x01 0x00		

6.3 Device Information Service Client (DISC)

6.3.1 Enabling the DISC Profile

1. Step 1: The GTL Host enables the DISC profile. The GTL Host sends the **DISC_ENABluetooth LE_REQ** command to DA14585/531 to enable the DISC profile. The whole message structure is initialized with zero.

Table 340. DISC_ENABLE_REQ symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	DISC_ENABLE_REQ	0x1500
DST_ID	2	TASK_ID_DISC	0x0015
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	60 bytes	0x3C
Message parameters			
con_type	1	Connection type	**0x00**: PRF_CON_DISCOVERY **0x01**: PRF_CON_NORMAL
{padding}	1	Any value is valid.	{padding} Ignored.
dis.svc.shdl	2	Start handle	0xXXXX
dis.svc.ehdl	2	End handle	0xXXXX
dis.chars[0].char_hdl	2	Characteristic handle	0xXXXX
dis.chars[0].val_hdl	2	Value handle	0xXXXX
dis.chars[0].prop	1	Characteristic properties	0xXX
dis.chars[0]. char_ehdl_off	1	End of characteristic offset	0xXX
{dis.chars [1], dis.char[2]…} Note 1			
dis.chars[8].char_hdl	2	Characteristic handle	0xXXXX
dis.chars[8].val_hdl	2	Value handle	0xXXXX

Parameters	Bytes	Description	Data
dis.chars[8].Prop	1	Characteristic properties	0xXX
dis.chars[8].char_ehdl_off	1	End of characteristic offset	0xXX

Note 1 For index of Device Information Characteristics, see Table 341.

Table 341. DISC characteristics

Characteristic	Description	Index
DISC_MANUFACTURER_NAME_CHAR	Manufacturer Name	0x00
DISC_MODEL_NB_STR_CHAR	Model Number	0x01
DISC_SERIAL_NB_STR_CHAR	Serial Number	0x02
DISC_HARD_REV_STR_CHAR	Hardware Revision	0x03
DISC_FIRM_REV_STR_CHAR	Firmware Revision	0x04
DISC_SW_REV_STR_CHAR	Software Revision	0x05
DISC_SYSTEM_ID_CHAR	System ID	0x06
DISC_IEEE_CHAR	IEEE Certification	0x07
DISC_PNP_ID_CHAR	PnP ID	0x08

Table 342. DISC_ENABLE_REQ example

Message stream: DISC_ENABLE_REQ		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x15	DISC_ENABLE_REQ
DST_ID	0x15 0x00	TASK_ID_DISC
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x3C 0x00	60 bytes
con_type	0x00	PRF_CON_DISCOVERY
{padding}	0x00	{padding} Ignored. Any value is valid.
dis.svc.shdl	0x00 0x00	Start Handle
dis.svc.ehdl	0x00 0x00	End Handle
dis.chars[0].char_hdl	0x00 0x00	Characteristic handle
dis.chars[0].val_hdl	0x00 0x00	Value handle
dis.chars[0].prop	0x00	Characteristic properties
dis.chars[0].char_ehdl_off	0x00	End of characteristic offset
… (All characteristics are initialized with zero)		
dis.chars[8].char_hdl	0x00 0x00	Characteristic handle
dis.chars[8].val_hdl	0x00 0x00	Value handle
dis.chars[8].prop	0x00	Characteristic properties
dis.chars[8].char_ehdl_off	0x00	End of characteristic offset
Message string		
0x05 0x00 0x15 0x15 0x00 0x10 0x00 0x3C 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

2. Step 2: GTL Host receives the **DISC_ENABLE_RSP** message that populates the DISC characteristics.

Table 343. DISC_ENABLE_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	DISC_ENABLE_RSP	0x1501
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_DISC	0x0015
PAR_LEN	2	60	0x3C
Message parameters			
status	1	Response Status	See Table 382, Profile Error Codes
{padding}	1	Ignored. Any value is valid	{padding}
dis.svc.shdl	2	Start handle	0xXXXX
dis.svc.ehdl	2	End handle	0xXXXX
dis.chars[0].char_hdl	2	Characteristic handle	0xXXXX
dis.chars[0].val_hdl	2	Value handle	0xXXXX
dis.chars[0].prop	1	Characteristic properties	0xXX
dis.chars[0]. char_ehdl_off	1	End of characteristic offset	0xXX
… (All characteristics are initialized with zero) Note 1			
dis.chars[8].char_hdl	2	Characteristic handle	0xXXXX
dis.chars[8].val_hdl	2	Value handle	0xXXXX
dis.chars[8].prop	1	Characteristic properties	0xXX
dis.chars[8].char_ehdl_off	1	End of characteristic offset	0xXX

Note 1 For index of Device Information Characteristics, see Table 341.

Table 344. DISC_ENABLE_RSP example

Message stream: example for DISC_ENABLE_RSP		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x15	DISC_ENABLE_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x15 0x00	TASK_ID_DISS
PAR_LEN	0x3C 0x00	60 bytes
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	{padding}. Ignored. Any value is valid
dis.svc.shdl	0x17 0x00	Start Handle: 0x0017
dis.svc.ehdl	0x23 0x00	End Handle: 0x0023
dis.chars[0].char_hdl	0x18 0x00	Characteristic Handle: 0x0018
dis.chars[0].val_hdl	0x19 0x00	Value Handle: 0x0019
dis.chars[0].prop	0x02	ATT_CHAR_PROP_RD
dis.chars[0]. char_ehdl_off	0x01	End of characteristic offset: 0x01

Message stream: example for DISC_ENABLE_RSP		
dis.chars[1].char_hdl	0x1A 0x00	Characteristic Handle: 0x001A
dis.chars[1].val_hdl	0x1B 0x00	Value Handle: 0x001B
dis.chars[1].prop	0x02	ATT_CHAR_PROP_RD
dis.chars[1].char_ehdl_off	0x01	End of characteristic offset: 0x01
dis.chars[2].char_hdl	0x00 0x00	Not Populated
dis.chars[2].val_hdl	0x00 0x00	Not Populated
dis.chars[2].prop	0x00	Not Populated
dis.chars[2].char_ehdl_off	0x00	Not Populated
dis.chars[3].char_hdl	0x00 0x00	Not Populated
dis.chars[3].val_hdl	0x00 0x00	Not Populated
dis.chars[3].prop	0x00	Not Populated
dis.chars[3].char_ehdl_off	0x00	Not Populated
dis.chars[4].char_hdl	0x1C 0x00	Characteristic Handle: 0x001C
dis.chars[4].val_hdl	0x1D 0x00	Value Handle: 0x001D
dis.chars[4].prop	0x02	ATT_CHAR_PROP_RD
dis.chars[4].char_ehdl_off	0x01	End of characteristic offset: 0x01
dis.chars[5].char_hdl	0x1E 0x00	Characteristic Handle: 0x001E
dis.chars[5].val_hdl	0x1F 0x00	Value Handle: 0x001F
dis.chars[5].prop	0x02	ATT_CHAR_PROP_RD
dis.chars[5]. char_ehdl_off	0x01	End of characteristic offset: 0x01
dis.chars[6].char_hdl	0x20 0x00	Characteristic Handle: 0x0020
dis.chars[6].val_hdl	0x21 0x00	Value Handle: 0x0021
dis.chars[6].prop	0x02	ATT_CHAR_PROP_RD
dis.chars[6].char_ehdl_off	0x01	End of characteristic offset: 0x01
dis.chars[7].char_hdl	0x00 0x00	Not Populated
dis.chars[7].val_hdl	0x00 0x00	Not Populated
dis.chars[7].prop	0x00	Not Populated
dis.chars[7].char_ehdl_off	0x00	Not Populated
dis.chars[8].char_hdl	0x22 0x00	Characteristic Handle: 0x0022
dis.chars[8].val_hdl	0x23 0x00	Value Handle: 0x0023
dis.chars[8].prop	0x02	ATT_CHAR_PROP_RD
dis.chars[8].char_ehdl_off	0x01	End of characteristic offset: 0x01
Message string		
0x05 0x01 0x15 0x10 0x00 0x15 0x00 0x3C 0x00 0x00 0x00 0x17 0x00 0x23 0x00 0x18 0x00 0x19 0x00 0x02 0x01 0x1A 0x00 0x1B 0x00 0x02 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x1C 0x00 0x1D 0x00 0x02 0x01 0x1E 0x00 0x1F 0x00 0x02 0x01 0x20 0x00 0x21 0x00 0x02 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x22 0x00 0x23 0x00 0x02 0x01		

6.3.2 Reading a Characteristic Value of the DISC Profile

1. Step 1: The GTL Host sends the **DISC_RD_CHAR_REQ** command to DA14585/531 to read a DISC characteristic value.

Table 345. DISC_RD_CHAR_REQ symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	DISC_RD_CHAR_REQ	0x1502
DST_ID	2	TASK_ID_DISC	0x0015
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	1 byte	0x0001
Message parameters			
char_code	1	Characteristic value code	See Table 341

Table 346. DISC_RD_CHAR_REQ example

Message stream: DISC_RD_CHAR_REQ example for DISC characteristic "manufacturer name" (DISC_MANUFACTURER_NAME_CHAR)		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x02 0x15	DISC_RD_CHAR_REQ
DST_ID	0x15 0x00	TASK_ID_DISC
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
char_code	0x00	DISC_MANUFACTURER_NAME_CHAR
Message string		
0x05 0x00 0x14 0x14 0x00 0x10 0x00 0x0F 0x00 0x00 0x0B 0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x53 0x65 0x6D 0x69 0x00 0x00		

2. Step 2: The Bluetooth LE application responds to the **DISC_RD_CHAR_REQ** command by sending the **DISC_RD_CHAR_RSP** message with the characteristic value to the GTL Host.

Table 347. DISC_RD_CHAR_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	DISC_RD_CHAR_RSP	0x1503
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_DISC	0x0015
PAR_LEN	2	Depending on info.value	0xXXXX
Message parameters			
info.handle	2	Attribute Handle	0xXXXX
info.length	2	Attribute length	0xXXXX
info.status	1	Status of request	See Table 382
info.value	{length}	The characteristic's content	

Table 348. DISC_RD_CHAR_RSP example

Message stream: DISC_RD_CHAR_RSP example		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x15	DISC_RD_CHAR_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x15 0x00	TASK_ID_DISC
PAR_LEN	0x11 0x00	17 bytes
info.handle	0x19 0x00	0x00: DIS_MANUFACTURER_NAME_CHAR
info.length	0x0B 0x00	11 bytes
info.status	0x00	**0x00**: GAP_ERR_NO_ERROR
info.value	0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x53 0x65 0x6D 0x69	"Renesas"
{padding}	0x00	Ignored. Any value is valid
Message string		
0x05 0x03 0x15 0x10 0x00 0x15 0x00 0x11 0x00 0x19 0x00 0x0B 0x00 0x00 0x44 0x69 0x61 0x6C 0x6F 0x67 0x20 0x53 0x65 0x6D 0x69 0x00		

6.4 Proximity Monitor (PROXM) Profile

6.4.1 Enabling the PROXM Profile

1. Step 1: The GTL Host sends the **PROXM_ENABLE_REQ** command to DA14585/531 to enable the PROXM profile. The whole message structure is initialized with zero.

Table 349. PROXM_ENABLE_REQ symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	PROXM_ENABLE_REQ	0x1600
DST_ID	2	TASK_ID_PROXM	0x0016
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	32 bytes	0x0020
Message parameters			
con_type	1	Connection type	**0x00**: PRF_CON_DISCOVERY**0x01**: PRF_CON_NORMAL
{padding}	1	Ignored. Any value is valid.	{padding}
lls.svc.shdl	2	LLS start handle	0xXXXX
lls.svc.ehdl	2	LLS end handle	0xXXXX
lls.characts[0].char_hdl	2	LLS characteristic handle	0xXXXX
lls.characts[0].val_hdl	2	LLS value handle	0xXXXX
lls.characts[0].prop	1	LLS characteristic properties	0xXX
lls.characts[0].char_ehdl_off	1	LLS End of characteristic offset	0xXX

Parameters	Bytes	Description	Data
ias.svc.shdl	2	IAS start handle	0xXXXX
ias.svc.ehdl	2	IAS end handle	0xXXXX
ias.characts[0].char_hdl	2	IAS characteristic handle	0xXXXX
ias.characts[0].val_hdl	2	IAS value handle	0xXXXX
ias.characts[0].prop	1	IAS characteristic properties	0xXX
ias.characts[0].char_ehdl_off	1	IAS End of characteristic offset	0xXX
txps.svc.shdl	2	TXPS start handle	0xXXXX
txps.svc.ehdl	2	TXPS end handle	0xXXXX
txps.characts[0].char_hdl	2	TXPS characteristic handle	0xXXXX
txps.characts[0].val_hdl	2	TXPS value handle	0xXXXX
txps.characts[0].prop	1	TXPS characteristic properties	0xXX
txps.characts[0].char_ehdl_off	1	TXPS End of characteristic offset	0xXX

Table 350. PROXM_ENABLE_REQ example

Message stream: example for PROXM_ENABLE_REQ		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x16	PROXM_ENABLE_REQ
DST_ID	0x16 0x00	TASK_ID_PROXM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x20 0x00	32 bytes
con_type	0x00	PRF_CON_DISCOVERY
{padding}	0x00	{padding} Ignored. Any value is valid.
lls.svc.shdl	0x00 0x00	LLS start handle
lls.svc.ehdl	0x00 0x00	LLS end handle
lls.characts[0].char_hdl	0x00 0x00	LLS characteristic handle
lls.characts[0].val_hdl	0x00 0x00	LLS value handle
lls.characts[0].prop	0x00	LLS characteristic properties
lls.characts[0].char_ehdl_off	0x00	LLS End of characteristic offset
ias.svc.shdl	0x00 0x00	IAS start handle
ias.svc.ehdl	0x00 0x00	IAS end handle
ias.characts[0].char_hdl	0x00 0x00	IAS characteristic handle
ias.characts[0].val_hdl	0x00 0x00	IAS value handle
ias.characts[0].prop	0x00	IAS characteristic properties
ias.characts[0].char_ehdl_off	0x00	IAS End of characteristic offset
txps.svc.shdl	0x00 0x00	TXPS start handle
txps.svc.ehdl	0x00 0x00	TXPS end handle
txps.characts[0].char_hdl	0x00 0x00	TXPS characteristic handle
txps.characts[0].val_hdl	0x00 0x00	TXPS value handle
txps.characts[0].prop	0x00	TXPS characteristic properties
txps.characts[0].char_ehdl_off	0x00	TXPS end of characteristic offset
Message string		
0x05 0x00 0x16 0x16 0x00 0x10 0x00 0x32 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00		

2. Step 2: The Bluetooth LE application sends the **PROXM_ENABLE_RSP** message to the GTL Host. This message contains the values of the RPOXM characteristics.

Table 351. PROXM_ENABLE_RSP symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	PROXM_ENABLE_RSP	0x1601
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_PROXM	0x0016
PAR_LEN	2	32 bytes	0x0020

Parameters	Bytes	Description	Data
Message parameters			
status	1	Response Status	See Table 382, Profile Error Codes
lls.svc.shdl	2	LLS start handle	0xXXXX
lls.svc.ehdl	2	LLS end handle	0xXXXX
lls.characts[0].char_hdl	2	LLS characteristic handle	0xXXXX
lls.characts[0].val_hdl	2	LLS value handle	0xXXXX
lls.characts[0].prop	1	LLS characteristic properties	0xXX
lls.characts[0].char_ehdl_off	1	LLS End of characteristic offset	0xXX
ias.svc.shdl	2	IAS start handle	0xXXXX
ias.svc.ehdl	2	IAS end handle	0xXXXX
ias.characts[0].char_hdl	2	IAS characteristic handle	0xXXXX
ias.characts[0].val_hdl	2	IAS value handle	0xXXXX
ias.characts[0].prop	1	IAS characteristic properties	0xXX
ias.characts[0].char_ehdl_off	1	IAS End of characteristic offset	0xXX
txps.svc.shdl	2	TXPS start handle	0xXXXX
txps.svc.ehdl	2	TXPS end handle	0xXXXX
txps.characts[0].char_hdl	2	TXPS characteristic handle	0xXXXX
txps.characts[0].val_hdl	2	TXPS value handle	0xXXXX
txps.characts[0].prop	1	TXPS characteristic properties	0xXX
txps.characts[0].char_ehdl_off	1	TXPS end of characteristic offset	0xXX

Table 352. PROXM_ENABLE_RSP example

Message stream: example for PROXM_ENABLE_RSP		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x01 0x16	PROXM_ENABLE_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x16 0x00	TASK_ID_PROXM
PAR_LEN	0x20 0x00	32 bytes
status	0x00	ATT_ERR_NO_ERROR
{padding}	0x00	{padding}. Ignored. Any value is valid
lls.svc.shdl	0x0A 0x00	LLS start handle = 0x000A
lls.svc.ehdl	0x0C 0x00	LLS end handle = 0x000C
lls.characts[0].char_hdl	0x0B 0x00	LLS characteristic handle = 0x000B
lls.characts[0].val_hdl	0x0C 0x00	LLS value handle = 0x000C
lls.characts[0].prop	0x0A	LLS characteristic properties = 0x0A (Table 383)
lls.characts[0].char_ehdl_off	0x01	LLS End of characteristic offset = 0x01
ias.svc.shdl	0x0D 0x00	IAS start handle = 0x000D
ias.svc.ehdl	0x0F 0x00	IAS end handle = 0x000F
ias.characts[0].char_hdl	0x0E 0x00	IAS characteristic handle = 0x000E
ias.characts[0].val_hdl	0x0F 0x00	IAS value handle = 0x000F
ias.characts[0].prop	0x04	IAS characteristic properties = 0x04 (Table 383)
ias.characts[0].char_ehdl_off	0x01	IAS End of characteristic offset = 0x01
txps.svc.shdl	0x10 0x00	TXPS start handle = 0x0010
txps.svc.ehdl	0x12 0x00	TXPS end handle = 0x0012
txps.characts[0].char_hdl	0x11 0x00	TXPS characteristic handle = 0x0011
txps.characts[0].val_hdl	0x12 0x00	TXPS value handle = 0x0012
txps.characts[0].prop	0x02	TXPS characteristic properties = 0x02 (Table 383)
txps.characts[0].char_ehdl_off	0x01	TXPS end of characteristic offset = 0x01
Message string		
0x05 0x01 0x16 0x10 0x00 0x16 0x00 0x20 0x00 0x00 0x00 0x0A 0x00 0x0C 0x00 0x0B 0x00 0x0C 0x00 0x0A 0x01 0x0D 0x00 0x0F 0x00 0x0E 0x00 0x0F 0x00 0x04 0x01 0x10 0x00 0x12 0x00 0x11 0x00 0x12 0x00 0x02 0x01		

6.4.2 Read LLS Alert or TX Power Level – Request and Response

1. Step 1: The GTL Host receives the **PROXM_RD_REQ** message from DA14585/531 for the PROMX alert characteristic value.

Table 353. PROXM_RD_REQ symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	PROXM_RD_REQ	0x1602
DST_ID	2	TASK_ID_PROXM	0x0016

Parameters	Bytes	Description	Data
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	1 byte	0x0001
Message parameters			
svc_code	1	Code for the service in which the alert level should be read	**0x00**: LLS **0x01**: TXPS

Table 354. PROXM_RD_REQ example

Message stream: PROXM_RD_REQ example for TXPS value		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x02 0x16	PROXM_RD_REQ
DST_ID	0x16 0x00	TASK_ID_PROXM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
svc_code	0x01	TXPS
Message string		
0x05 0x02 0x16 0x16 0x00 0x10 0x00 0x01 0x00 0x01		

2. Step 2: The Bluetooth LE application responds to the **PROXM_RD_REQ** command by sending the **PROXM_RD_RSP** message to the GTL Host.

Table 355. PROXM_RD_RSP symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	PROXM_RD_RSP	0x1603
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_PROXM	0x0016
PAR_LEN	2	3 bytes	0x0004
Message parameters			
svc_code	1	Code for the service in which the alert level should be read	**0x00**: LLS **0x01**: TXPS
status	1	Response status	0xXX (See Table 382)
value	1	Characteristic Value	**0x00**: PROXM_ALERT_NONE **0x01**: PROXM_ALERT_MILD **0x02**: PROXM_ALERT_HIGH

Table 356. PROXM_RD_RSP example

Message stream: PROXM_RD_RSP example		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x03 0x16	PROXM_RD_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x16 0x00	TASK_ID_DISC
PAR_LEN	0x03 0x00	17 bytes
svc_code	0x01	TXPS
status	0x00	GAP_ERR_NO_ERROR
value	0x00	PROXM_ALERT_NONE
Message string		
0x05 0x03 0x16 0x10 0x00 0x16 0x00 0x03 0x00 0x01 0x00 0x00		

6.4.3 Set the Alert Level – Request and Response

1. Step 1: The GTL Host sends the **PROXM_WR_ALERT_LVL_REQ** command to DA14585/531 to set the value of the PROXM Alert Level characteristic.

Table 357. PROXM_WR_ALERT_LVL_REQ symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	PROXM_RD_REQ	0x1604
DST_ID	2	TASK_ID_PROXM	0x0016
SRC_ID	2	TASK_ID_GTL	0x0010
PAR_LEN	2	2 bytes	0x0002
Message parameters			
svc_code	1	Code for the service in which the alert level should be read	**0x00**: LLS **0x01**: TXPS
lvl	1	Alert Level	**0x00**: PROXM_ALERT_NONE **0x01**: PROXM_ALERT_MILD **0x02**: PROXM_ALERT_HIGH

Table 358. PROXM_WR_ALERT_LVL_REQ example

Message stream: PROXM_WR_ALERT_LVL_REQ example		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x04 0x16	PROXM_WR_ALERT_LVL_REQ
DST_ID	0x16 0x00	TASK_ID_PROXM
SRC_ID	0x10 0x00	TASK_ID_GTL
PAR_LEN	0x01 0x00	1 byte
svc_code	0x01	TXPS
lvl	0x02	PROXM_ALERT_HIGH
Message string		
0x05 0x04 0x16 0x16 0x00 0x10 0x00 0x01 0x00 0x01 0x02		

2. Step 2: The Bluetooth LE application responds to the **PROXM_WR_ALERT_LVL_REQ** command by sending the **PROXM_WR_ALERT_LVL_RSP** message to the GTL Host.

Table 359. PROXM_WR_ALERT_LVL_RSP symbolic message structure

Parameters	**Bytes**	**Description**	**Data**
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	PROXM_WR_ALERT_LVL_RSP	0x1605
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	TASK_ID_PROXM	0x0016
PAR_LEN	2	1 byte	0x0001
Message parameters			
status	1	Response status	0xXX (See Table 382)

Table 360. PROXM_WR_ALERT_LVL_RSP example

Message stream: PROXM_WR_ALERT_LVL_RSP example		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x05 0x16	PROXM_WR_ALERT_LVL_RSP
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x16 0x00	TASK_ID_DISC
PAR_LEN	0x01 0x00	1 byte
status	0x00	GAP_ERR_NO_ERROR
Message string		
0x05 0x05 0x16 0x10 0x00 0x16 0x00 0x01 0x00 0x00		

6.5 Proximity Reporter (PROXR) Profile

The PROXR profile sends **PROXR_ALERT_IND** to the GTL Host when the peripheral changes the alert level characteristic value.

Table 361. PROXR_ALERT_IND symbolic message structure

Parameters	Bytes	Description	Data
Header			
Initiator	1	GTL Initiator	0x05
MSG_ID	2	PROXR_ALERT_IND	0x1700
DST_ID	2	TASK_ID_GTL	0x0010
SRC_ID	2	nn, TASK_ID_PROXR (Note 1)	0xnn17, (nn = conidx)
PAR_LEN	2	3 bytes	0x0003
Message parameters			
conidx	1	Connection index	0xXX
alert_lvl	1	Alert Level	**0x00**: PROXR_ALERT_NONE **0x01**: PROXR_ALERT_MILD **0x02**: PROXR_ALERT_HIGH
char_code	1	Char Code - Indicate if LLS or IAS	0xXX

Note 1 "nn" refers to an incrementing connection index. 0 for the first connection, 1 for the second connection, and so forth.

Table 362. PROXR_ALERT_IND example

Message stream: PROXR_ALERT_IND example		
Hex data		
Data token	**Hex data (LSB)**	**Comments**
Initiator	0x05	GTL Initiator
MSG_ID	0x00 0x17	PROXR_ALERT_IND
DST_ID	0x10 0x00	TASK_ID_GTL
SRC_ID	0x17 0x00	TASK_ID_GATTC, conidx = 0
PAR_LEN	0x03 0x00	2 bytes
conidx	0x00	condix = 0
alert_lvl	0x01	Alert Level = PROXR_ALERT_MILD
char_code	0x01	IAS alert
Message string		
0x05 0x00 0x17 0x10 0x00 0x17 0x00 0x03 0x00 0x00 0x01 0x01		

7. Tasks, Command IDs, and Their Documentation Status

Table 363. TASK_ID_GATTM (id: 0x0B)

Command mnemonic	Command ID (Hex)	Documentation status
GATTM_ADD_SVC_REQ	0x0B00	Detailed in this document
GATTM_ADD_SVC_RSP	0x0B01	
GATTM_SVC_GET_PERMISSION_REQ	0x0B02	
GATTM_SVC_GET_PERMISSION_RSP	0x0B03	
GATTM_SVC_SET_PERMISSION_REQ	0x0B04	
GATTM_SVC_SET_PERMISSION_RSP	0x0B05	
GATTM_ATT_GET_PERMISSION_REQ	0x0B06	
GATTM_ATT_GET_PERMISSION_RSP	0x0B07	
GATTM_ATT_SET_PERMISSION_REQ	0x0B08	
GATTM_ATT_SET_PERMISSION_RSP	0x0B09	
GATTM_ATT_GET_VALUE_REQ	0x0B0A	
GATTM_ATT_GET_VALUE_RSP	0x0B0B	
GATTM_ATT_SET_VALUE_REQ	0x0B0C	
GATTM_ATT_SET_VALUE_RSP	0x0B0D	
GATTM_DESTROY_DB_REQ	0x0B0E	Not included in this document, for debugging only
GATTM_DESTROY_DB_RSP	0x0B0F	
GATTM_SVC_GET_LIST_REQ	0x0B10	
GATTM_SVC_GET_LIST_RSP	0x0B11	
GATTM_ATT_GET_INFO_REQ	0x0B12	Not included in this document
GATTM_ATT_GET_INFO_RSP	0x0B13	

Table 364. TASK_ID_GATTC: (id: 0x0C)

Command mnemonic	Command ID (Hex)	Documentation status
GATTC_CMP_EVT	0x0C00	Detailed in this document
GATTC_EXC_MTU_CMD	0x0C01	
GATTC_MTU_CHANGED_IND	0x0C02	
GATTC_DISC_CMD	0x0C03	
GATTC_DISC_SVC_IND	0x0C04	
GATTC_DISC_SVC_INCL_IND	0x0C05	
GATTC_DISC_CHAR_IND	0x0C06	
GATTC_DISC_CHAR_DESC_IND	0x0C07	
GATTC_READ_CMD	0x0C08	
GATTC_READ_IND	0x0C09	
GATTC_WRITE_CMD	0x0C0A	
GATTC_EXECUTE_WRITE_CMD	0x0C0B	
GATTC_EVENT_IND	0x0C0C	
GATTC_EVENT_REQ_IND	0x0C0D	
GATTC_EVENT_CFM	0x0C0E	
GATTC_REG_TO_PEER_EVT_CMD	0x0C0F	
GATTC_SEND_EVT_CMD	0x0C10	
GATTC_SEND_SVC_CHANGED_CMD	0x0C11	
GATTC_SVC_CHANGED_CFG_IND	0x0C12	
GATTC_READ_REQ_IND	0x0C13	
GATTC_READ_CFM	0x0C14	
GATTC_WRITE_REQ_IND	0x0C15	
GATTC_WRITE_CFM	0x0C16	
GATTC_ATT_INFO_REQ_IND	0x0C17	
GATTC_ATT_INFO_CFM	0x0C18	
GATTC_SDP_SVC_DISC_CMD	0x0C19	
GATTC_SDP_SVC_IND	0x0C1A	
GATTC_TRANSACTION_TO_ERROR_IND	0x0C1B	Not included in this document
GATTC_CLIENT_RTX_IND	0x0C1C	Internal, not documented
GATTC_SERVER_RTX_IND	0x0C1D	

Table 365. TASK_ID_GAPM: (id: 0x0D)

Command mnemonic	Command ID (Hex)	Documentation status
GAPM_CMP_EVT	0x0D00	Detailed in this document
GAPM_DEVICE_READY_IND	0x0D01	
GAPM_RESET_CMD	0x0D02	
GAPM_CANCEL_CMD	0x0D03	
GAPM_SET_DEV_CONFIG_CMD	0x0D04	
GAPM_SET_CHANNEL_MAP_CMD	0x0D05	Not included in this document
GAPM_GET_DEV_INFO_CMD	0x0D06	Detailed in this document
GAPM_DEV_VERSION_IND	0x0D07	
GAPM_DEV_BDADDR_IND	0x0D08	
GAPM_DEV_ADV_TX_POWER_IND	0x0D09	
GAPM_DBG_MEM_INFO_IND	0x0D0A	
GAPM_WHITE_LIST_MGT_CMD	0x0D0B	
GAPM_WHITE_LIST_SIZE_IND	0x0D0C	
GAPM_START_ADVERTISE_CMD	0x0D0D	
GAPM_UPDATE_ADVERTISE_DATA_CMD	0x0D0E	Not included in this document
GAPM_START_SCAN_CMD	0x0D0F	Detailed in this document
GAPM_ADV_REPORT_IND	0x0D10	
GAPM_START_CONNECTION_CMD	0x0D11	
GAPM_PEER_NAME_IND	0x0D12	Not included in this document
GAPM_CONNECTION_CFM	0x0D13	
GAPM_RESOLV_ADDR_CMD	0x0D14	Detailed in this document
GAPM_ADDR_SOLVED_IND	0x0D15	
GAPM_GEN_RAND_ADDR_CMD	0x0D16	Not included in this document
GAPM_USE_ENC_BLOCK_CMD	0x0D17	
GAPM_USE_ENC_BLOCK_IND	0x0D18	
GAPM_GEN_RAND_NB_CMD	0x0D19	
GAPM_GEN_RAND_NB_IND	0x0D1A	
GAPM_PROFILE_TASK_ADD_CMD	0x0D1B	Detailed in this document
GAPM_PROFILE_ADDED_IND	0x0D1C	
GAPM_UNKNOWN_TASK_IND	0x0D1D	
GAPM_SUGG_DFLT_DATA_LEN_IND	0x0D1E	
GAPM_MAX_DATA_LEN_IND	0x0D1F	
GAPM_RAL_MGT_CMD	0x0D20	
GAPM_RAL_SIZE_IND	0x0D21	Included in this document
GAPM_RAL_ADDR_IND	0x0D22	
GAPM_LIM_DISC_TO_IND	0x0D23	Internal, not documented
GAPM_SCAN_TO_IND	0x0D24	
GAPM_ADDR_RENEW_TO_IND	0x0D25	
GAPM_UNKNOWN_TASK_MSG	0x0D26	

Command mnemonic	Command ID (Hex)	Documentation status
GAPM_USE_P256_BLOCK_CMD	0x0D27	Included in this document
GAPM_USE_P256_BLOCK_IND	0x0D28	

Table 366. TASK_ID_GAPC: (id: 0x0E)

Command mnemonic	Command ID (Hex)	Documentation status
GAPC_CMP_EVT	0x0E00	Detailed in this document
GAPC_CONNECTION_REQ_IND	0x0E01	
GAPC_CONNECTION_CFM	0x0E02	
GAPC_DISCONNECT_IND	0x0E03	
GAPC_DISCONNECT_CMD	0x0E04	
GAPC_GET_INFO_CMD	0x0E05	
GAPC_PEER_ATT_INFO_IND	0x0E06	
GAPC_PEER_VERSION_IND	0x0E07	
GAPC_PEER_FEATURES_IND	0x0E08	
GAPC_CON_RSSI_IND	0x0E09	
GAPC_GET_DEV_INFO_REQ_IND	0x0E0A	
GAPC_GET_DEV_INFO_CFM	0x0E0B	
GAPC_SET_DEV_INFO_REQ_IND	0x0E0C	
GAPC_SET_DEV_INFO_CFM	0x0E0D	
GAPC_PARAM_UPDATE_CMD	0x0E0E	
GAPC_PARAM_UPDATE_REQ_IND	0x0E0F	
GAPC_PARAM_UPDATE_CFM	0x0E10	
GAPC_PARAM_UPDATED_IND	0x0E11	
GAPC_BOND_CMD	0x0E12	Not included in this document
GAPC_BOND_REQ_IND	0x0E13	Detailed in this document
GAPC_BOND_CFM	0x0E14	
GAPC_BOND_IND	0x0E15	
GAPC_ENCRYPT_CMD	0x0E16	Not included in this document
GAPC_ENCRYPT_REQ_IND	0x0E17	Detailed in this document
GAPC_ENCRYPT_CFM	0x0E18	
GAPC_ENCRYPT_IND	0x0E19	
GAPC_SECURITY_CMD	0x0E1A	
GAPC_SECURITY_IND	0x0E1B	Not included in this document
GAPC_SIGN_COUNTER_IND	0x0E1C	Detailed in this document
GAPC_CON_CHANNEL_MAP_IND	0x0E1D	
GAPC_LECB_CREATE_CMD	0x0E1E	
GAPC_LECB_DESTROY_CMD	0x0E1F	
GAPC_LECB_CONNECT_CMD	0x0E20	
GAPC_LECB_CONNECT_REQ_IND	0x0E21	
GAPC_LECB_CONNECT_IND	0x0E22	
GAPC_LECB_CONNECT_CFM	0x0E23	

Command mnemonic	Command ID (Hex)	Documentation status
GAPC_LECB_ADD_CMD	0x0E24	
GAPC_LECB_ADD_IND	0x0E25	
GAPC_LECB_DISCONNECT_CMD	0x0E26	
GAPC_LECB_DISCONNECT_IND	0x0E27	
GAPC_SET_LE_PING_TO_CMD	0x0E28	Not included in this document
GAPC_LE_PING_TO_VAL_IND	0x0E29	Detailed in this document
GAPC_LE_PING_TO_IND	0x0E2A	
GAPC_SET_LE_PKT_SIZE_CMD	0x0E2B	
GAPC_LE_PKT_SIZE_IND	0x0E2C	
GAPC_SIGN_CMD	0x0E2D	Internal, not documented
GAPC_SIGN_IND	0x0E2E	
GAPC_PARAM_UPDATE_TO_IND	0x0E2F	
GAPC_SMP_TIMEOUT_TIMER_IND	0x0E30	
GAPC_SMP_REP_ATTEMPTS_TIMER_IND	0x0E31	
GAPC_LECB_CONN_TO_IND	0x0E32	
GAPC_LECB_DISCONN_TO_IND	0x0E33	
GAPC_KEYPRESS_NOTIFICATION	0x0E34	Included in this document
GAPC_KEYPRESS_NOTIFICATION_CMD		
GAPC_KEYPRESS_NOTIFICATION_IND		

Table 367. TASK_ID_GTL: (id: 0x10)

Command mnemonic	Command ID (Hex)	Documentation status
Virtual task ID for GTL Host application. No messages	-	

Table 368. TASK_ID_DISS: (id: 0x14)

Command mnemonic	Command ID (Hex)	Documentation status
DISS_SET_VALUE_REQ	0x1400	Detailed in this document
DISS_SET_VALUE_RSP	0x1401	
DISS_VALUE_REQ_IND	0x1402	
DISS_VALUE_CFM	0x1403	

Table 369. TASK_ID_DISC: (id: 0x15)

Command mnemonic	Command ID (Hex)	Documentation status
DISC_ENABLE_REQ	0x1500	Detailed in this document
DISC_ENABLE_RSP	0x1501	
DISC_RD_CHAR_REQ	0x1502	
DISC_RD_CHAR_RSP	0x1503	

Table 370. TASK_ID_PROXM: (id: 0x16)

Command mnemonic	Command ID (Hex)	Documentation status
PROXM_ENABLE_REQ	0x1500	Detailed in this document
PROXM_ENABLE_RSP	0x1501	
PROXM_RD_REQ	0x1502	

PROXM_RD_RSP	0x1503	
PROXM_WR_ALERT_LVL_REQ	0x1504	
PROXM_WR_ALERT_LVL_RSP	0x1505	

Table 371. TASK_ID_PROXR: (id: 0x17)

Command mnemonic	Command ID (Hex)	Documentation status
PROXR_ALERT_IND	0x1700	Detailed in this document

Appendix A Reference Tables

A.1 GAPC Operation Flags

Table 372. GAPC operation flags

Value	Flag	Description
0x00	GAPC_NO_OP	No operation
Connection management		
0x01	GAPC_DISCONNECT	Disconnect link
Connection information		
0x02	GAPC_GET_PEER_NAME	Retrieve name of peer device
0x03	GAPC_GET_PEER_VERSION	Retrieve peer device version info
0x04	GAPC_GET_PEER_FEATURES	Retrieve peer device features
0x05	GAPC_GET_PEER_APPEARANCE	Retrieve device appearance
0x06	GAPC_GET_PEER_SLV_PREF_PARAMS	Retrieve peer device Slave Preferred Parameters
0x07	GAPC_GET_CON_RSSI	Retrieve connection RSSI
0x08	GAPC_GET_CON_CHANNEL_MAP	Retrieve connection channel map
Connection parameters update		
0x09	GAPC_UPDATE_PARAMS	Perform update of connection parameters
Security procedures		
0x0A	GAPC_BOND	Start bonding procedure
0x0B	GAPC_ENCRYPT	Start encryption procedure
0x0C	GAPC_SECURITY_REQ	Start security request procedure
LE credit-based L2CAP connection		
0x0D	GAPC_LE_CB_CREATE	LE credit based - creation
0x0E	GAPC_LE_CB_DESTROY	LE credit based - destruction
0x0F	GAPC_LE_CB_CONNECTION	LE credit based – connection request
0x10	GAPC_LE_CB_DISCONNECTION	LE credit based – disconnection request
0x11	GAPC_LE_CB_ADDITION	LE credit based – add credit
LE ping management		
0x12	GAPC_GET_LE_PING_TO	Get Timer timeout value
0x13	GAPC_SET_LE_PING_TO	Set Timer timeout value
LE DLE		
0x14	GAPC_SET_LE_PKT_SIZE	LE set data length
0x15	GAPC_GET_PEER_CENTRAL_RPA	Get Peer device central address resolution
0x16	GAPC_GET_PEER_RPA_ONLY	Get Peer Resolvable Private Address only

A.2 GATTC Operation Flags

Table 373. GATTC operation flags

Value	Flag	Description
0x00	GATTC_NO_OP	No operation

Value	Flag	Description
MTU negotiation		
0x01	GATTC_MTU_EXCH	Perform MTU exchange
Attribute discovery		
0x02	GATTC_DISC_ALL_SVC	Discover all services
0x03	GATTC_DISC_BY_UUID_SVC	Discover services by UUID
0x04	GATTC_DISC_INCLUDED_SVC	Discover included services
0x05	GATTC_DISC_ALL_CHAR	Discover all characteristics
0x06	GATTC_DISC_BY_UUID_CHAR	Discover characteristic by UUID
0x07	GATTC_DISC_DESC_CHAR	Discover characteristic descriptor
Read attribute		
0x08	GATTC_READ	Read attribute
0x09	GATTC_READ_LONG	Read long attribute
0x0A	GATTC_READ_BY_UUID	Read attribute by UUID
0x0B	GATTC_READ_MULTIPLE	Read multiple attribute
Write attribute		
0x0C	GATTC_WRITE	Write attribute
0x0D	GATTC_WRITE_NO_RESPONSE	Write no response
0x0E	GATTC_WRITE_SIGNED	Write signed
0x0F	GATTC_EXEC_WRITE	Execute write
Registering to peer device event		
0x10	GATTC_REGISTER	Register to peer device events
0x11	GATTC_UNREGISTER	Unregister from peer device events
Sending events to peer device		
0x12	GATTC_NOTIFY	Send an attribute notification
0x13	GATTC_INDICATE	Send an attribute indication
0x14	GATTC_SVC_CHANGED	Send a service changed indication
Service discovery procedure		
0x15	GATTC_SDP_DISC_SVC	Search specific service
0x16	GATTC_SDP_DISC_SVC_ALL	Search for all services
0x17	GATTC_SDP_DISC_CANCEL	Cancel Service Discovery Procedure

A.3 GAPM Operation Flags

Table 374. GAPM operation flags

Value	Flag	Description
0x00	GAPM_NO_OP	No operation
Default operations		
0x01	GAPM_RESET	Reset Bluetooth LE subsystem: LL and HL
0x02	GAPM_CANCEL	Cancel currently executed operation
Configuration operations		
0x03	GAPM_SET_DEV_CONFIG	Set device configuration

Value	Flag	Description
0x04	GAPM_SET_CHANNEL_MAP	Set device channel map
Retrieve device information		
0x05	GAPM_GET_DEV_VERSION	Get Local device version
0x06	GAPM_GET_DEV_BDADDR	Get Local device BD Address
0x07	GAPM_GET_DEV_ADV_TX_POWER	Get device advertising power level
Operation on whitelist		
0x08	GAPM_GET_WLIST_SIZE	Get Whitelist Size
0x09	GAPM_ADD_DEV_IN_WLIST	Add devices in whitelist
0x0A	GAPM_RMV_DEV_FRM_WLIST	Remove devices form whitelist
0x0B	GAPM_CLEAR_WLIST	Clear all devices from whitelist
Advertise mode operations		
0x0C	GAPM_ADV_NON_CONN	Start non-connectable advertising
0x0D	GAPM_ADV_UNDIRECT	Start undirected connectable advertising
0x0E	GAPM_ADV_DIRECT	Start directed connectable advertising
0x0F	GAPM_ADV_DIRECT_LDC	Start directed connectable advertising using Low Duty Cycle
0x10	GAPM_UPDATE_ADVERTISE_DATA	Update on-the-fly advertising data
Scan mode operations		
0x11	GAPM_SCAN_ACTIVE	Start active scan operation
0x12	GAPM_SCAN_PASSIVE	Start passive scan operation
Connection mode operations		
0x13	GAPM_CONNECTION_DIRECT	Direct connection operation
0x14	GAPM_CONNECTION_AUTO	Automatic connection operation
0x15	GAPM_CONNECTION_SELECTIVE	Selective connection operation
0x16	GAPM_CONNECTION_NAME_REQUEST	Name Request operation (requires starting a direct connection)
Security/encryption toolbox		
0x17	GAPM_RESOLV_ADDR	Resolve device address
0x18	GAPM_GEN_RAND_ADDR	Generate a random address
0x19	GAPM_USE_ENC_BLOCK	Use the controller's AES-128 block
0x1A	GAPM_GEN_RAND_NB	Generate an 8-byte random number
Profile management		
0x1B	GAPM_PROFILE_TASK_ADD	Create new task for specific profile
DEBUG		
0x1C	GAPM_DBG_GET_MEM_INFO	Get memory usage
0x1D	GAPM_PLF_RESET	Perform a platform reset
Data length		
0x1E	GAPM_SET_SUGGESTED_DFLT_LE_DATA_LEN	Set Suggested Default LE Data Length
0x1F	GAPM_GET_SUGGESTED_DFLT_LE_DATA_LEN	Get Suggested Default LE Data Length
0x20	GAPM_GET_MAX_LE_DATA_LEN	Get Maximum LE Data Length

Value	Flag	Description
Operation on resolving list		
0x21	GAPM_GET_RAL_SIZE	Get resolving address list size
0x22	GAPM_GET_RAL_LOC_ADDR	Get resolving local address
0x23	GAPM_GET_RAL_PEER_ADDR	Get resolving peer address
0x24	GAPM_ADD_DEV_IN_RAL	Add device in resolving address list
0x25	GAPM_RMV_DEV_FRM_RAL	Remove device from resolving address list
0x26	GAPM_CLEAR_RAL	Clear resolving address list
Various		
0x27	GAPM_USE_P256_BLOCK	Use the controller's DHKEY P256 block
0x28	GAPM_NETWORK_MODE_RAL	Set Network Privacy Mode for peer in resolving list
0x29	GAPM_DEVICE_MODE_RAL	Set Device Privacy Mode for peer in resolving list

A.4 GAP Device Role

Table 375. GAP device role

Value	Flag	Description
0x00	GAP_ROLE_NONE	No role set yet
0x01	GAP_ROLE_OBSERVER	Observer role
0x02	GAP_ROLE_BROADCASTER	Broadcaster role
0x05	GAP_ROLE_CENTRAL	Master/Central role (has also observer role)
0x0A	GAP_ROLE_PERIPHERAL	Peripheral/Slave role (has also broadcaster role)
0x0F	GAP_ROLE_ALL	Device has all role, both peripheral and central
0x80	GAP_ROLE_DBG_LE_4_0	Debug Mode Only: force LL configuration on Bluetooth LE 4.0

A.5 GAP Authentication Requirements

Table 376. GAP authentication requirements

Value	Flag	Description
0x00	GAP_AUTH_REQ_NO_MITM_NO_BOND	No Man in The Middle (MITM) protection/No Bonding
0x01	GAP_AUTH_REQ_NO_MITM_BOND	No MITM/Bonding
0x04	GAP_AUTH_REQ_MITM_NO_BOND	MITM/No Bonding
0x05	GAP_AUTH_REQ_MITM_BOND	MITM/Bonding
0x08	GAP_AUTH_REQ_SECURE_CONNECTION	Secure Connection
0x10	GAP_AUTH_REQ_KEYPRESS_NOTIFICATION (Note 1)	Key Press Notification

Note 1 This option can only be used with Secure Connections Pairing. So, it must be ORed with GAP_AUTH_REQ_SECURE_CONNECTION.

A.6 GAP Bonding Procedure Request or Information Code

Table 377. GAP bonding procedure request or information code

Value	Flag	Description
0x00	GAPC_PAIRING_REQ	Bond Pairing request
0x01	GAPC_PAIRING_RSP	Respond to Pairing request

Value	Flag	Description
0x02	GAPC_PAIRING_SUCCEED	Pairing Finished information
0x03	GAPC_PAIRING_FAILED	Pairing Failed information
0x04	GAPC_TK_EXCH	Used to retrieve pairing Temporary Key
0x05	GAPC_IRK_EXCH	Used for Identity Resolving Key exchange
0x06	GAPC_CSRK_EXCH	Used for Connection Signature Resolving Key exchange
0x07	GAPC_LTK_EXCH	Used for Long-term Key exchange
0x08	GAPC_REPEATED_ATTEMPT	Bond Pairing request issue, Repeated attempt

A.7 GAP I/O Capabilities

Table 378. GAP I/O capabilities

Value	Flag	Description
0x00	GAP_IO_CAP_DISPLAY_ONLY	Display Only
0x01	GAP_IO_CAP_DISPLAY_YES_NO	Display Yes No
0x02	GAP_IO_CAP_KB_ONLY	Keyboard Only
0x03	GAP_IO_CAP_NO_INPUT_NO_OUTPUT	No Input No Output
0x04	GAP_IO_CAP_KB_DISPLAY	Keyboard Display

A.8 GAP Key Distribution Flags

Table 379. GAP key distribution flags

Value	Flag	Description
0x00	GAP_KDIST_NONE	No Keys to distribute
0x01	GAP_KDIST_ENCKEY	Encryption key in distribution
0x02	GAP_KDIST_IDKEY IRK	(ID key) in distribution
0x04	GAP_KDIST_SIGNKEY CSRK	(Signature key) in distribution

A.9 GAP Device Security Requirements

Table 380. GAP device security requirements

Value	Flag	Description
0x00	GAP_NO_SEC	No security (no authentication and encryption)
0x01	GAP_SEC1_NOAUTH_PAIR_ENC	Unauthenticated pairing with encryption
0x02	GAP_SEC1_AUTH_PAIR_ENC	Authenticated pairing with encryption
0x03	GAP_SEC2_NOAUTH_DATA_SGN	Unauthenticated pairing with data signing
0x04	GAP_SEC2_AUTH_DATA_SGN	Authentication pairing with data signing

A.10 GAP Valid Disconnection Reasons

Table 381. GAP valid disconnection reasons

Value	Flag	Description
0x05	CO_ERROR_AUTH_FAILURE	The Authentication Failure error code indicates that pairing or authentication failed due to incorrect results in the pairing or authentication procedure. This could be due to an incorrect PIN or Link Key.

Value	Flag	Description
0x13	CO_ERROR_REMOTE_USER_TERM_CON	The Remote User Terminated Connection error code indicates that the user on the remote device terminated the connection.
0x14	CO_ERROR_REMOTE_DEV_TERM_LOW_RESOURCES	The Remote Device Terminated Connection due to Low Resources error code indicates that the remote device terminated the connection because of low resources.
0x15	CO_ERROR_REMOTE_DEV_POWER_OFF	The Remote Device Terminated Connection due to Power Off error code indicates that the remote device terminated the connection because the device is about to power off.
0x16	CO_ERROR_CON_TERM_BY_LOCAL_HOST	The Connection was terminated by the Local Host.
0x1A	CO_ERROR_UNSUPPORTED_REMOTE_FEATURE	The Unsupported Remote Feature error code indicates that the remote device does not support the feature associated with the issued command or LMP PDU.
0x29	CO_ERROR_PAIRING_WITH_UNIT_KEY_NOT_SUP	The Pairing With Unit Key Not Supported error code indicates that it was not possible to pair as a unit key was requested and it is not supported.
0x3B	CO_ERROR_UNACCEPTABLE_CONN_INT	The Unacceptable Connection Interval error code indicates that the remote device terminated the connection because of an unacceptable connection interval.

A.11 Host Stack Error Codes

Table 382. Host stack error codes

Value	Error	Description
0x00	GAP_ERR_NO_ERROR	No error
ATT error codes		
0x01	ATT_ERR_INVALID_HANDLE	Handle is invalid
0x02	ATT_ERR_READ_NOT_PERMITTED	Read permission disabled
0x03	ATT_ERR_WRITE_NOT_PERMITTED	Write permission disabled
0x04	ATT_ERR_INVALID_PDU	Incorrect PDU
0x05	ATT_ERR_INSUFF_AUTHEN	Authentication privilege not enough
0x06	ATT_ERR_REQUEST_NOT_SUPPORTED	Request not supported or not understood
0x07	ATT_ERR_INVALID_OFFSET	Incorrect offset value
0x08	ATT_ERR_INSUFF_AUTHOR	Authorization privilege not enough
0x09	ATT_ERR_PREPARE_QUEUE_FULL	Capacity queue for reliable write reached
0x0A	ATT_ERR_ATTRIBUTE_NOT_FOUND	Attribute requested not existing
0x0B	ATT_ERR_ATTRIBUTE_NOT_LONG	Attribute requested not long
0x0C	ATT_ERR_INSUFF_ENC_KEY_SIZE	Encryption size not sufficient
0x0D	ATT_ERR_INVALID_ATTRIBUTE_VAL_LEN	Invalid length of the attribute value
0x0E	ATT_ERR_UNLIKELY_ERR	Operation not fit to condition
0x0F	ATT_ERR_INSUFF_ENC	Attribute requires encryption before operation
0x10	ATT_UNSUPP_GRP_TYPE	Attribute grouping not supported
0x11	ATT_INSUFF_RESOURCE	Resources not sufficient to complete the request
0x80	ATT_ERR_APP_ERROR	Application Error

Value	Error	Description
L2C specific error		
0x30	L2C_ERR_CONNECTION_LOST	Message cannot be sent because connection lost. (disconnected)
0x31	L2C_ERR_INVALID_MTU_EXCEED	Invalid PDU length exceed MTU
0x32	L2C_ERR_INVALID_CID	Invalid Channel ID
0x33	L2C_ERR_INVALID_PDU	Invalid PDU
0x34	L2C_ERR_NO_RES_AVAIL	Connection refused - no resources available
0x35	L2C_ERR_INSUFF_AUTHEN	Connection refused - insufficient authentication
0x36	L2C_ERR_INSUFF_AUTHOR	Connection refused - insufficient authorization
0x37	L2C_ERR_INSUFF_ENC_KEY_SIZE	Connection refused - insufficient encryption key size
0x38	L2C_ERR_INSUFF_ENC	Connection Refused - insufficient encryption
0x39	L2C_ERR_LEPSM_NOT_SUPP	Connection refused - LE_PSM not supported
0x3A	L2C_ERR_NO_MORE_CREDIT	No more credit
0x3B	L2C_ERR_NOT_UNDERSTOOD	Command not understood by peer device
GAP error codes		
0x40	GAP_ERR_INVALID_PARAM	Invalid parameters set
0x41	GAP_ERR_PROTOCOL_PROBLEM	Problem with protocol exchange, get unexpected response
0x42	GAP_ERR_NOT_SUPPORTED	Request not supported by software configuration
0x43	GAP_ERR_COMMAND_DISALLOWED	Request not allowed in current state
0x44	GAP_ERR_CANCELED	Requested operation canceled
0x45	GAP_ERR_TIMEOUT	Requested operation timeout
0x46	GAP_ERR_DISCONNECTED	Link connection lost during operation
0x47	GAP_ERR_NOT_FOUND	Search algorithm finished, but no result found
0x48	GAP_ERR_REJECTED	Request rejected by peer device
0x49	GAP_ERR_PRIVACY_CFG_PB	Problem with privacy configuration
0x4A	GAP_ERR_ADV_DATA_INVALID	Duplicate or invalid advertising data
0x4B	GAP_ERR_INSUFF_RESOURCES	Insufficient resources
0x4C	GAP_ERR_UNEXPECTED	Unexpected Error
GATT error codes		
0x50	GATT_ERR_INVALID_ATT_LEN	Problem with ATTC protocol response
0x51	GATT_ERR_INVALID_TYPE_IN_SVC_SEARCH	Error in service search
0x52	GATT_ERR_WRITE	Invalid write data
0x53	GATT_ERR_SIGNED_WRITE	Signed write error
0x54	GATT_ERR_ATT_CLIENT_MISSING	No attribute client defined
0x55	GATT_ERR_ATT_SERVER_MISSING	No attribute server defined
0x56	GATT_ERR_INVALID_PERM	Permission set in service/attribute are invalid
SMP errors detected on local device		
0x61	SMP_ERROR_LOC_PASSKEY_ENTRY_FAILED	The user input of passkey failed, for example, the user cancelled the operation
0x62	SMP_ERROR_LOC_OOB_NOT_AVAILABLE	The OOB Data is not available

Value	Error	Description
0x63	SMP_ERROR_LOC_AUTH_REQ	The pairing procedure cannot be performed as authentication requirements cannot be met due to IO cap
0x64	SMP_ERROR_LOC_CONF_VAL_FAILED	The confirm value does not match the calculated confirm value
0x65	SMP_ERROR_LOC_PAIRING_NOT_SUPP	Pairing is not supported by the device
0x66	SMP_ERROR_LOC_ENC_KEY_SIZE	The resultant encryption key size is insufficient for the security requirements of this device
0x67	SMP_ERROR_LOC_CMD_NOT_SUPPORTED	The SMP command received is not supported on this device
0x68	SMP_ERROR_LOC_UNSPECIFIED_REASON	Pairing failed due to an unspecified reason
0x69	SMP_ERROR_LOC_REPEATED_ATTEMPTS	Pairing or Authentication procedure is disallowed because too little time has elapsed since last pairing
0x6A	SMP_ERROR_LOC_INVALID_PARAM	The command length is invalid, or a parameter is outside of the specified range
0x6B	SMP_ERROR_LOC_REQ_DISALLOWED	The request sent by the HL cannot be handled for some reasons (unauthorized source task, role)
0x6C	SMP_ERROR_LOC_LL_ERROR	An error has been received from the controller upon an encryption request
0x6D	SMP_ERROR_LOC_ADDR_RESOLV_FAIL	The provided resolvable address has not been resolved
0x6E	SMP_ERROR_LOC_SIGN_VERIF_FAIL	The Signature Verification Failed
0x6F	SMP_ERROR_LOC_TIMEOUT	The command cannot be executed because a SMP timeout has been raised during the connection
SMP errors detected by remote device		
0x71	SMP_ERROR_REM_PASSKEY_ENTRY_FAILED	The user input of passkey failed, for example, the user cancelled the operation
0x72	SMP_ERROR_REM_OOB_NOT_AVAILABLE	The OOB Data is not available
0x73	SMP_ERROR_REM_AUTH_REQ	The pairing procedure cannot be performed as authentication requirements cannot be met due to IO cap
0x74	SMP_ERROR_REM_CONF_VAL_FAILED	The confirm value does not match the calculated confirm value
0x75	SMP_ERROR_REM_PAIRING_NOT_SUPP	Pairing is not supported by the device
0x76	SMP_ERROR_REM_ENC_KEY_SIZE	The resultant encryption key size is insufficient for the security requirements of this device
0x77	SMP_ERROR_REM_CMD_NOT_SUPPORTED	The SMP command received is not supported on this device
0x78	SMP_ERROR_REM_UNSPECIFIED_REASON	Pairing failed due to an unspecified reason
0x79	SMP_ERROR_REM_REPEATED_ATTEMPTS	Pairing or Authentication procedure is disallowed because too little time has elapsed since last pairing
0x7A	SMP_ERROR_REM_INVALID_PARAM	The command length is invalid, or a parameter is outside of the specified range
0x7B	SMP_ERROR_REM_ENC_KEY_MISSING	The encryption procedure failed because the slave device did not find the LTK needed to start an encryption
0x7C	SMP_ERROR_REM_ENC_NOT_SUPPORTED	The encryption procedure failed because the slave device does not support the encryption feature
0x7D	SMP_ERROR_REM_ENC_TIMEOUT	A timeout has occurred during the start encryption session

Value	Error	Description
Profile error codes		
0x80	PRF_APP_ERROR	Application Error
0x81	PRF_ERR_INVALID_PARAM	Invalid parameter in request
0x82	PRF_ERR_INEXISTENT_HDL	Nonexistent handle for sending a read/write characteristic request
0x83	PRF_ERR_STOP_DISC_CHAR_MISSING	Discovery stopped due to missing attribute according to specification
0x84	PRF_ERR_MULTIPLE_SVC	Too many SVC instances found > protocol violation
0x85	PRF_ERR_STOP_DISC_WRONG_CHAR_PROP	Discovery stopped due to found attribute with incorrect properties
0x86	PRF_ERR_MULTIPLE_CHAR	Too many Char. instances found > protocol violation
0x87	PRF_ERR_NOT_WRITABLE	Attribute write not allowed
0x88	PRF_ERR_NOT_READABLE	Attribute read not allowed
0x89	PRF_ERR_REQ_DISALLOWED	Request not allowed
0x8A	PRF_ERR_NTF_DISABLED	Notification Not Enabled
0x8B	PRF_ERR_IND_DISABLED	Indication Not Enabled
0x8C	PRF_ERR_FEATURE_NOT_SUPPORTED	Feature not supported by profile
0x8D	PRF_ERR_UNEXPECTED_LEN	Read value has an unexpected length
0x8E	PRF_ERR_DISCONNECTED	Disconnection occurs
0x8F	PRF_ERR_PROC_TIMEOUT	Procedure Timeout
0xFD	PRF_CCCD_IMPR_CONFIGURED	Client characteristic configuration improperly configured
0xFE	PRF_PROC_IN_PROGRESS	Procedure already in progress
0xFF	PRF_OUT_OF_RANGE	Out of Range
LL error codes conveyed to upper layer		
0x91	LL_ERR_UNKNOWN_HCI_COMMAND	Unknown HCI Command
0x92	LL_ERR_UNKNOWN_CONNECTION_ID	Unknown Connection Identifier
0x93	LL_ERR_HARDWARE_FAILURE	Hardware Failure
0x94	LL_ERR_PAGE_TIMEOUT	BT Page Timeout
0x95	LL_ERR_AUTH_FAILURE	Authentication failure
0x96	LL_ERR_PIN_MISSING	Pin code missing
0x97	LL_ERR_MEMORY_CAPA_EXCEED	Memory capacity exceed
0x98	LL_ERR_CON_TIMEOUT	Connection Timeout
0x99	LL_ERR_CON_LIMIT_EXCEED	Connection limit Exceed
0x9A	LL_ERR_SYNC_CON_LIMIT_DEV_EXCEED	Synchronous Connection limit exceed
0x9B	LL_ERR_ACL_CON_EXISTS ACL	Connection exits
0x9C	LL_ERR_COMMAND_DISALLOWED	Command Disallowed
0x9D	LL_ERR_CONN_REJ_LIMITED_RESOURCES	Connection rejected due to limited resources
0x9E	LL_ERR_CONN_REJ_SECURITY_REASONS	Connection rejected due to security reason
0x9F	LL_ERR_CONN_REJ_UNACCEPTABLE_BDADDR	Connection rejected due to unacceptable BD Address
0xA0	LL_ERR_CONN_ACCEPT_TIMEOUT_EXCEED	Connection rejected due to Accept connection timeout
0xA1	LL_ERR_UNSUPPORTED	Not Supported

Value	Error	Description
0xA2	LL_ERR_INVALID_HCI_PARAM	Invalid parameters
0xA3	LL_ERR_REMOTE_USER_TERM_CON	Remote user terminate connection
0xA4	LL_ERR_REMOTE_DEV_TERM_LOW_RESOURCE	Remote device terminate connection due to low resources
0xA5	LL_ERR_REMOTE_DEV_POWER_OFF	Remote device terminate connection due to power off
0xA6	LL_ERR_CON_TERM_BY_LOCAL_HOST	Connection terminated by local host
0xA7	LL_ERR_REPEATED_ATTEMPTS	Repeated attempts
0xA8	LL_ERR_PAIRING_NOT_ALLOWED	Pairing not Allowed
0xA9	LL_ERR_UNKNOWN_LMP_PDU	Unknown PDU Error
0xAA	LL_ERR_UNSUPPORTED_REMOTE_FEATURE	Unsupported remote feature
0xAB	LL_ERR_SCO_OFFSET_REJECTED	SCO Offset rejected
0xAC	LL_ERR_SCO_INTERVAL_REJECTED	SCO Interval Rejected
0xAD	LL_ERR_SCO_AIR_MODE_REJECTED	SCO air mode Rejected
0xAE	LL_ERR_INVALID_LMP_PARAM	Invalid LMP parameters
0xAF	LL_ERR_UNSPECIFIED_ERROR	Unspecified error
0xB0	LL_ERR_UNSUPPORTED_LMP_PARAM_VALUE	Unsupported LMP Parameter value
0xB1	LL_ERR_ROLE_CHANGE_NOT_ALLOWED	Role Change Not allowed
0xB2	LL_ERR_LMP_RSP_TIMEOUT LMP	Response timeout
0xB3	LL_ERR_LMP_COLLISION LMP	Collision
0xB4	LL_ERR_LMP_PDU_NOT_ALLOWED	LMP PDU not allowed
0xB5	LL_ERR_ENC_MODE_NOT_ACCEPT	Encryption mode not accepted
0xB6	LL_ERR_LINK_KEY_CANT_CHANGE	Link Key Cannot be changed
0xB7	LL_ERR_QOS_NOT_SUPPORTED	Quality of Service not supported
0xB8	LL_ERR_INSTANT_PASSED	Error, instant passed
0xB9	LL_ERR_PAIRING_WITH_UNIT_KEY_NOT_SU	Pairing with unit key not supported
0xBA	LL_ERR_DIFF_TRANSACTION_COLLISION	Transaction collision
0xBC	LL_ERR_QOS_UNACCEPTABLE_PARAM	Unacceptable parameters
0xBD	LL_ERR_QOS_REJECTED	Quality of Service rejected
0xBE	LL_ERR_CHANNEL_CLASS_NOT_SUP	Channel class not supported
0xBF	LL_ERR_INSUFFICIENT_SECURITY	Insufficient security
0xC0	LL_ERR_PARAM_OUT_OF_MAND_RANGE	Parameters out of mandatory range
0xC2	LL_ERR_ROLE_SWITCH_PEND	Role switch pending
0xC4	LL_ERR_RESERVED_SLOT_VIOLATION	Reserved slot violation
0xC5	LL_ERR_ROLE_SWITCH_FAIL	Role Switch fail
0xC6	LL_ERR_EIR_TOO_LARGE	Error, EIR too large
0xC7	LL_ERR_SP_NOT_SUPPORTED_HOST	Simple pairing not supported by host
0xC8	LL_ERR_HOST_BUSY_PAIRING	Host pairing is busy
0xCA	LL_ERR_CONTROLLER_BUSY	Controller is busy
0xCB	LL_ERR_UNACCEPTABLE_CONN_INT	Unacceptable connection initialization
0xCC	LL_ERR_DIRECT_ADV_TO Direct	Advertising Timeout
0xCD	LL_ERR_TERMINATED_MIC_FAILURE	Connection Terminated due to a MIC failure

Value	Error	Description
0xCE	LL_ERR_CONN_FAILED_TO_BE_EST	Connection failed to be established

A.12 Characteristics Properties

IMPORTANT NOTE
When the characteristic has more than one property the values are ORed.

Table 383. Characteristics properties bit

Characteristic property	Value
ATT_CHAR_PROP_BCAST	0x01
ATT_CHAR_PROP_RD	0x02
ATT_CHAR_PROP_WR_NO_RESP	0x04
ATT_CHAR_PROP_WR	0x08
ATT_CHAR_PROP_NTF	0x10
ATT_CHAR_PROP_IND	0x20
ATT_CHAR_PROP_AUTH	0x40
ATT_CHAR_PROP_EXT_PROP	0x80

Revision History

Revision	Date	Description
1.1	Aug 6, 2024	Added support for DA14535.
1.0	Sept 14, 2022	First release.

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Renesas Electronics suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.