Generic Access Profile

Functional Specification

RW-BLE-GAP-SW-FS

Version 11.01

2020-01-17

RivieraWaves
http://www.rivierawaves.com/

Revision History

Version	Date	Revision Description	Author
11.00	2019-6-18	First version with 5.2 features Generic Access profile part of Host document	FBE / LT review
11.01	2020-01-17	Update versions to 5.2 number	FBE

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

Acronym or abbreviation	Writing out in full / Meaning
AD_TYPE	Advertising Data Type (https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile)
BLE	Bluetooth Low Energy. Also termed LE.
CID	Channel Identifier
CSRK	Connection Signature Resolving Key
FS	Functional Specification
FW	Firmware
GAP	Generic Access Profile
GAPC	Generic Access Profile Controller
GAPM	Generic Access Profile Manager
GATT	Generic Attribute Profile
GTL	Generic Transport Layer
HCI	Host Controller Interface
IRK	Identity Resolving Key
L2CAP	Logical Link Control and Adaptation Protocol
LE	(Bluetooth) Low Energy
LE_PSM	Low Energy Protocol/Service Multiplexer
LL	Lower Layer (Link Manager and Link Controller)
LTK	Long Term Key
MITM	Man-In-The-Middle Attack
MPS	Maximum Packet Size
MTU	Maximum Transmission Unit
OOB	Out Of Band
PDU	Protocol Data Unit
PRP	Proximity Profile
PSM	Protocol/Service Multiplexer
RAL	Resolving Address List
RO	Read Only
SC	Secure Connection
SDU	Service Data Unit
SIG	Special Interest Group
SM	Security Manager
SMP	Security Manager Protocol. BLE block responsible for security.
STK	Short Term Key
UUID	Universally Unique Identifier

1 Overview

1.1 Document Overview

This document describes the embedded Software (SW) implementation of the Generic Access Profile (GAP) block as part of the RivieraWaves (RW) Bluetooth Low Energy (BLE) Stack. Its purpose is to explain architecture and behavior of this block and its interactions with other parts of RivieraWaves BLE IPs.
Even if this block is a proprietary block, this document requires the intended audience to have knowledge about the Bluetooth protocol stack. Please refer to the Bluetooth vMilan standard specification (see [1]).

1.2 Architecture Overview

As shown in Figure 1-1, GAP block is part of the BLE Host stack. GAP lies above the Link Layer (or HCI) and L2CAP, also it interfaces with higher layer protocols.

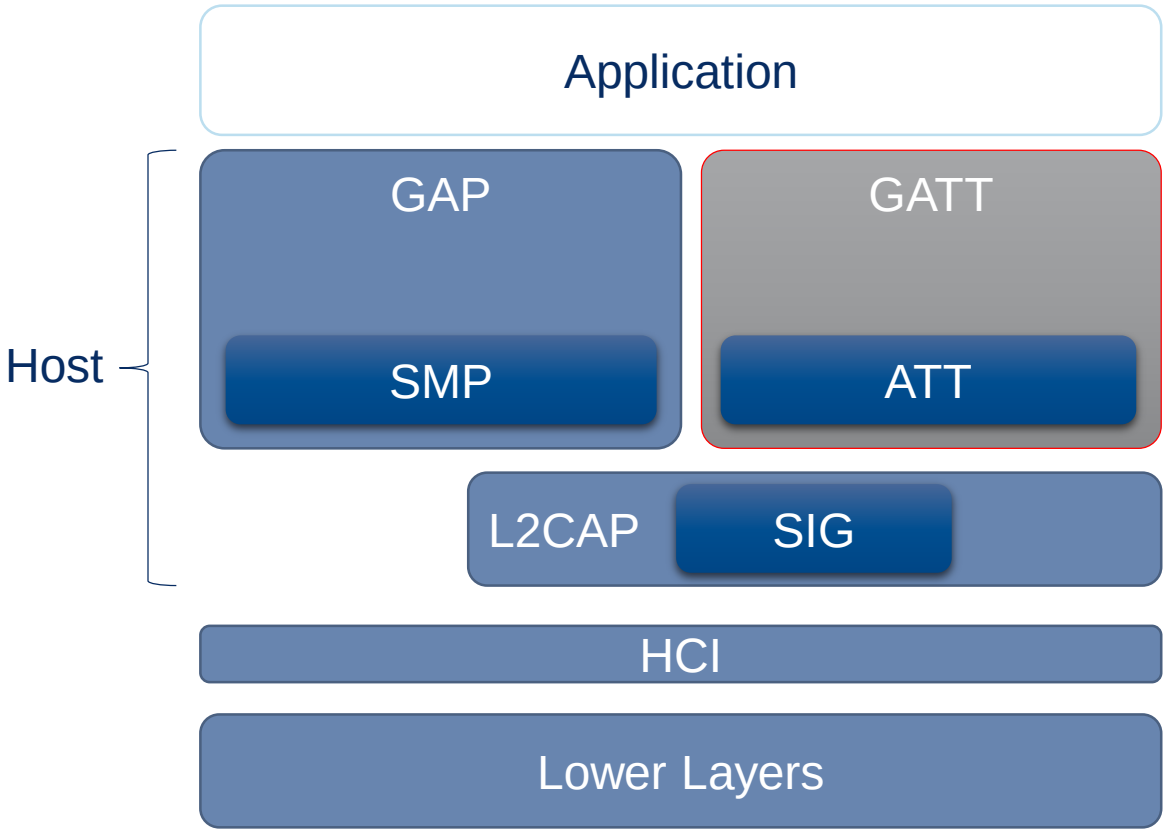

Figure 1-1: Position in the BLE Stack

In order to be able to exchange messages with the HCI task and upper layer application, GAP block has its own task.

1.3 Presentation of GAP Sub-blocks

This profile states the requirements on names, values and coding schemes used for names of parameters and procedures experienced on the user interface level.

This profile describes the general procedures that can be used for establishing connections to other Bluetooth devices that are able to accept connections and service requests.

GAP defines two parties (A and B) in establishing Bluetooth communication.

> A-Party: the device that is either scanning in the link layer scanning state or initiating in the link layer initiating state.
>
> B-Party: the device that is either advertising in the link layer advertising state or accepting the link request.

The GAP allows minimal functionality in absence of other profiles and provides an API when other profiles are present (see [8]).

GAP module is composed by 4 sub-blocks:

- Management of non-connected activities
- Management of connected activities
- Handle low energy security
- Handle life cycle of upper layer profiles

2 Modes and Roles

GAP introduces three device types based on supported Core Configurations.

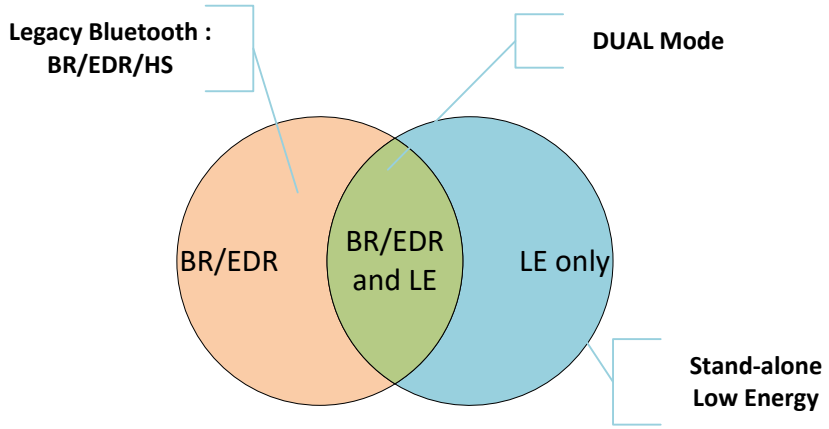

Figure 2-1: Device Types

Devices of type LE-only and BR/EDR/LE are capable of operating over an LE physical channel.

Note: Our implementation of Generic Access Profile supports only LE Only mode.

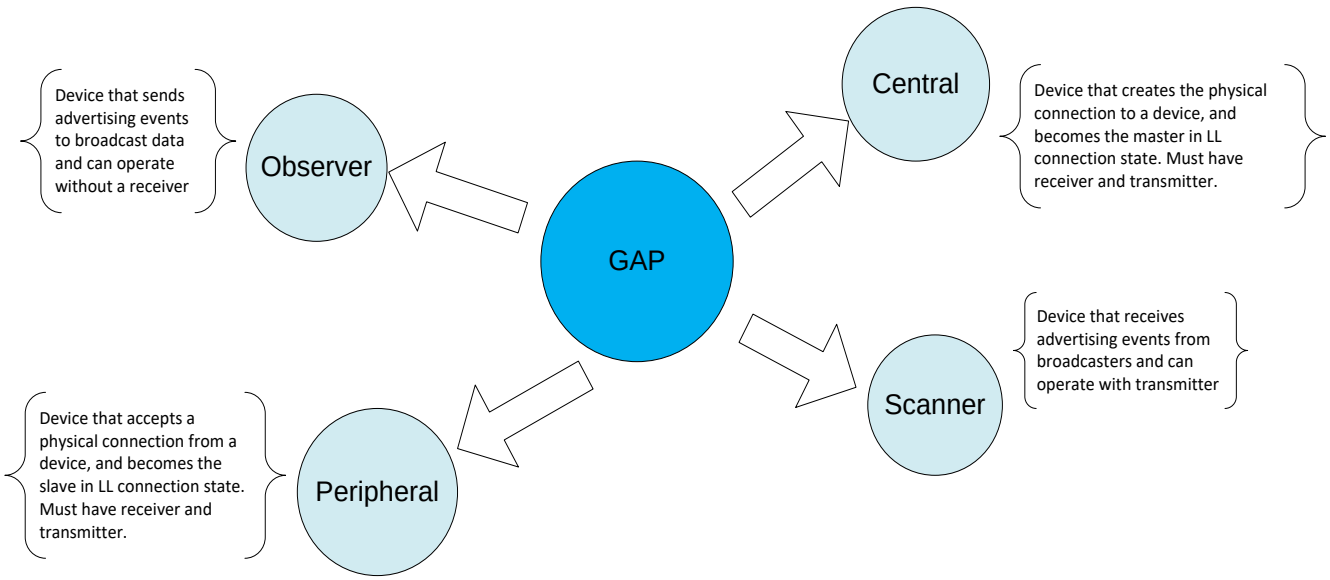

Figure 2-2: GAP Roles

In addition of Figure 2-2, Peripheral is able to broadcast data, and a Central is able to enter in observable mode.

A device can support all roles in same time, so that it can act both as a central (scan + master of a link) and peripheral (advertise + slave of a link).

A device supporting all modes cannot start two non-connected operation (such as advertising, scanning or connection init) in same time.

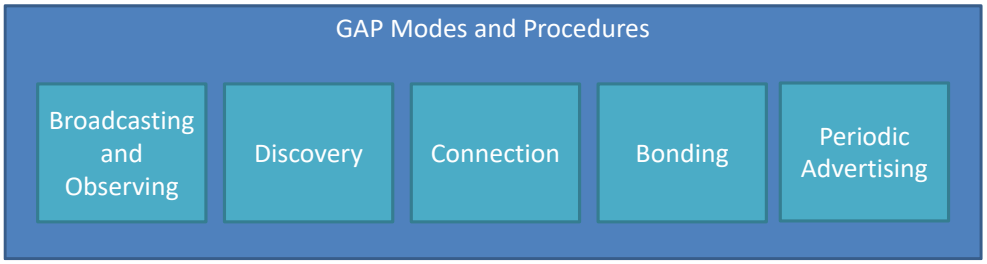

Figure 2-3: LE Operational Modes

GAP defines the general procedures that can be used for discovering identities, names and basic capabilities of other BLE devices that are discoverable. It also describes the ability of a device to be connected and discovered by another device.

3 Non-connected procedures

This chapter describes the API provided to an application aiming to launch non-connected procedures.

Management of these non-connected procedures is based on creation of activities representing the different available procedures. Four kinds of activities can be created:

- Advertising activity (see 3.2)
- Scanning activity (see 3.3)
- Initiating activity (see 3.4)
- Periodic Synchronization activity (see 3.5)

3.1 Activity overview

GAP API provides a set of command messages allowing to:

- Create an activity : GAPM_ACTIVITY_CREATE_CMD
- Start a created activity: GAPM_ACTIVITY_START_CMD
- Stop a started activity: GAPM_ACTIVITY_STOP_CMD
- Delete a created activity: GAPM_ACTIVITY_DELETE_CMD

Descriptions of these commands can be found in [8].

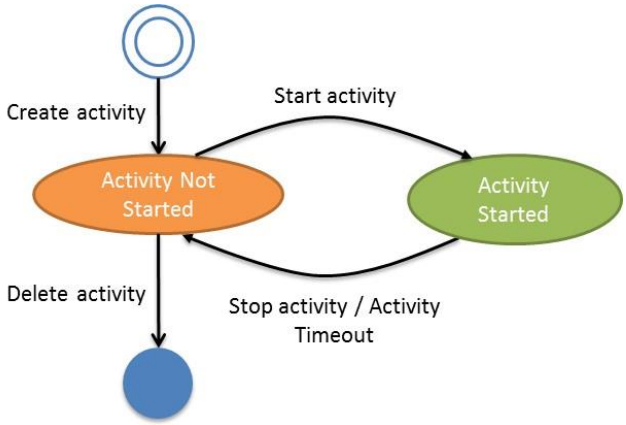

Figure 3-1: Activity life cycle

Figure 3-1 shows life cycle of an activity from an upper view.

- Before being usable an activity must first be created.
- Activity must then be started.
- The activity can be considered as over after several events:
  - Request is received from application
  - Timeout
  - End of requested operation (after connection, synchronization,…)
- Activity can either be started again if application needs to perform the same procedure or it can be deleted so that the allocated structure can be reused by another activity.

Note: *It is not possible to update activity's parameters. To do so, application must create another activity with different parameters.*

The number of activities that can be created in parallel depends on number of activities supported by upper layers.

However few rules exist about the activities that can be started in parallel:

- It is possible to create and start several advertising activities in parallel.
- It is not possible to start two scanning or two initiating activities because due to HCI commands allowing to manage such operations. However it is possible to create them.
- It is possible to start one scanning and one initiating activities in parallel.

The following figure exposes a more detailed state machine for an activity whatever its type and the events triggering update the activity state.

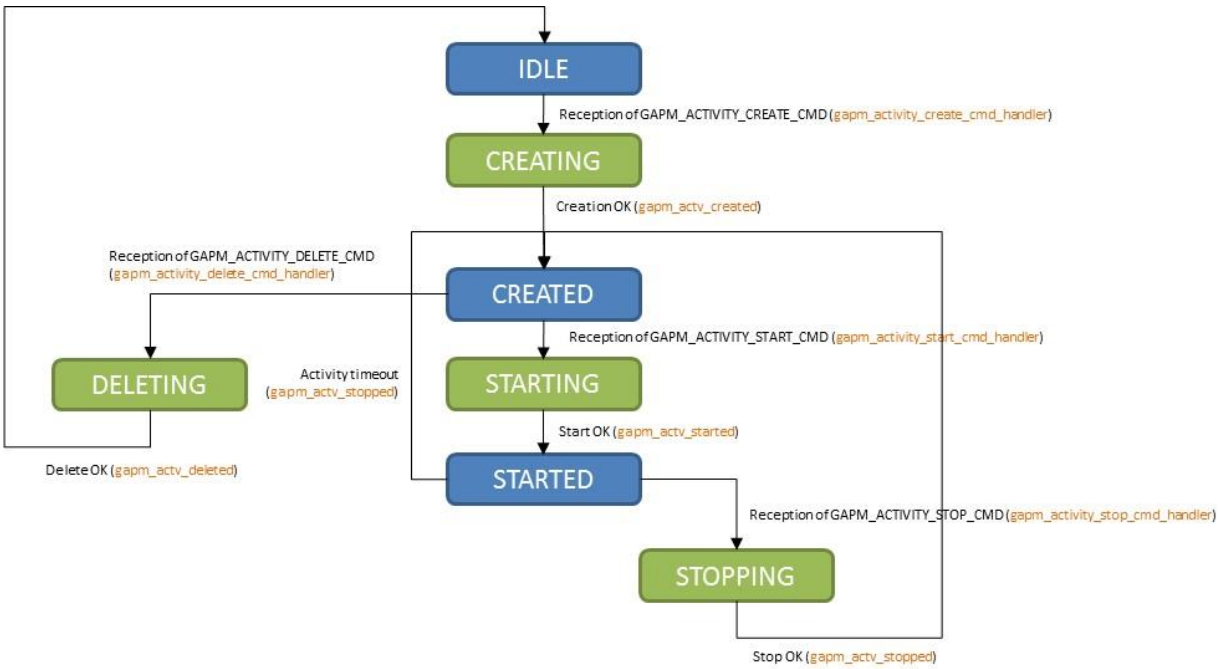

Figure 3-2: Activity state machine

3.2 Advertising activity

An advertising activity can be run on a device at least configured as a broadcaster device.

An advertising activity is defined by its discoverable and its connectable mode as described in Table 3-1.

Discoverable mode	Connectable mode
Non discoverable mode: Procedure that can be limited in time. A device in this mode cannot be found by a general or limited discovery	**Non-connectable mode:** A device in this mode cannot be connected by a central device.

procedure Filtering policy or targeted address can be used in this procedure In AD_TYPE flag of advertising data, LE General and LE Limited discoverable flag are set to zero	
General discoverable mode: Procedure without duration limitation. A device in this mode can be found by a general discovery procedure. *White list shall not be involved in this mode.* In AD_TYPE flag of advertising data, LE General is set to 1 and LE Limited discoverable flag is set to zero	**Undirected-connectable mode:** A device in this mode can accept connection from any device or from device present in the white list
Limited discoverable mode: Procedure with a limited duration. A device in this mode can be found either by a general or limited discovery procedure *White list shall not be involved in this mode.* In AD_TYPE flag of advertising data, LE General is set to zero and LE Limited discoverable flag is set to 1	**Directed-connectable mode:** A device in this mode can accept connection only by the targeted address

Table 3-1 Advertising activity modes

In Table 3-1, the Periodic Advertising Synchronizability mode and Broadcast mode are missing.

- Periodic Advertising Synchronizability mode is neither a connectable mode nor a discoverable mode. A device in this mode shall send synchronization information about periodic advertising.
- Broadcast mode is a non-connectable and non-discoverable mode.

Creation of advertising activity is possible using GAPM_ACTIVITY_CREATE_CMD which allows creation of three different types of advertising:

- Legacy Advertising Activity
- Extended Advertising Activity
- Periodic Advertising Activity

Note: the discoverability modes such as non-discoverable mode, general discoverable mode and limited discoverable mode are considered as a property of the advertising mode.

3.2.1 Advertising Properties Meaning

The advertising properties are used to describe content of advertising packet or behavior of the advertising activity. This section provides a small description for all the configurable properties:

- **Directed**: this property means that a specific device is targeted by the advertiser; targeted device address is present in advertising packet. For legacy advertising this applies only in connectable mode, this is not the case for extended advertising
- **High Duty Cycle**: applies only in legacy direct advertising. The controller advertises direct connectable packet for 1.28s with an advertising interval <= 3.75ms.
- **Scannable**: means that advertising activity opens an RX window to receive a scan request packet and sends in response a scan response packet.

- **Connectable**: means that advertising activity opens an RX window to receive a connect request packet. This property is mandatory to start a connection as slave.
- **Anonymous**: applies only in extended advertising which is neither connectable nor scannable. The device address is not present at all in the advertised packet (but can contain a targeted address).
- **TX power**: applies only in extended advertising. This mean that transmit power is present in advertising packet.
- **Scan Request notification**: When a scan request packet is received over the air, a scan report is triggered to inform application about observer device present over the air.
- **Filter policy**: indicates if the white list is involved or not to accept scan requests or connect requests. This property applies only for non-discoverable mode advertising.

3.2.2 Legacy Advertising Activity

Create Legacy Advertising Activity operation allows an application to start legacy advertising on primary advertising channels (37, 38, and 39) at 1 Mb/s. Table 3-2 shows properties available for each kind of advertising mode supported for legacy advertising.

Modes / Properties **0 – must be disabled** **1 – must be active** **X – could be either active or disabled**	**Disc Mode**	**High Duty Cycle**	**Directed**	**Scannable**	**Connectable**	**TX Packet Type**
Non-connectable	Non-disc ***(Broadcaster Mode)***	0	0	X	0	
	Limited	0	0	X	0	
	General	0	0	X	0	
Undirected Connectable	Non-disc	0	0	1	1	ADV_IND
	Limited	0	0	1	1	
	General	0	0	1	1	
Directed Connectable	Non-disc	X	1	0	1	ADV_DIRECT_IND

Table 3-2 Advertising properties for legacy advertising (primary channel only)

3.2.3 Extended Advertising Activity

Create Extended Advertising Activity operation allows an application to start extended advertising on secondary advertising channels (0 to 36) using 1 Mb/s, 2Mb/s or LE Coded PHY. Table 3-3 shows properties available for each kind of advertising mode supported for extended advertising.

Note: *Advertising extension does not support to have both connected and scannable mode (scannable means that advertiser replies to AUX_SCAN_REQ). So it is possible to set the scan response data only for non-connected modes if the scan property is set.*

Modes / Properties **0 – must be disabled** **1 – must be active** **X – could be either active or disabled**	**Disc Mode**	**Anonymous**	**Directed**	**Scannable**	**Connectable**
Non-connectable	General	0	0	X	0
	Limited	0	0	X	0
	Non-disc ***(Broadcaster Mode)***	0	X	X	0
	Non-disc ***(Broadcaster Mode)***	1	X	0	0
Undirected Connectable	Non-disc	0	0	0	1
	General	0	0	0	1
	Limited	0	0	0	1
Directed Connectable	Non-disc	0	1	0	1

Table 3-3 Advertising properties for extended advertising (usage of secondary channel)

3.2.4 Periodic Advertising Activity

Create Periodic Advertising Activity operation allows an application to start a periodic advertising on secondary advertising channels (0 to 36) using 1 Mb/s, 2Mb/s or LE Coded PHY. In order for a scanner to get information about position of the periodic advertising, a non-connectable and non-scannable extended advertising activity is started.

With this activity it is possible to set advertising data (limited to one fragment) and periodic advertising data.

Table 3-4 shows properties available for periodic advertising.

Modes / Properties **0 – must be disabled** **1 – must be active** **X – could be either active or disabled**	**Disc Mode**	**Anonymous**	**Directed**	**Scannable**	**Connectable**
Periodic Advertising Synchronizability	General	0	0	0	0
	Limited	0	0	0	0
	Non-disc	0	X	0	0

Table 3-4 Advertising properties for periodic advertising (usage of secondary channel)

These configurations have been retrieved from core specification (Vol 3 Part C Chapter 9 and Vol 2 Part E Chapter 7.8.53 see [1]).

3.2.5 Operations Overview

Harmonization between chosen advertising type and provided advertising properties are checked during advertising activity creation procedure based on rules exposed in sections 3.2.2, 3.2.3 and 3.2.4.

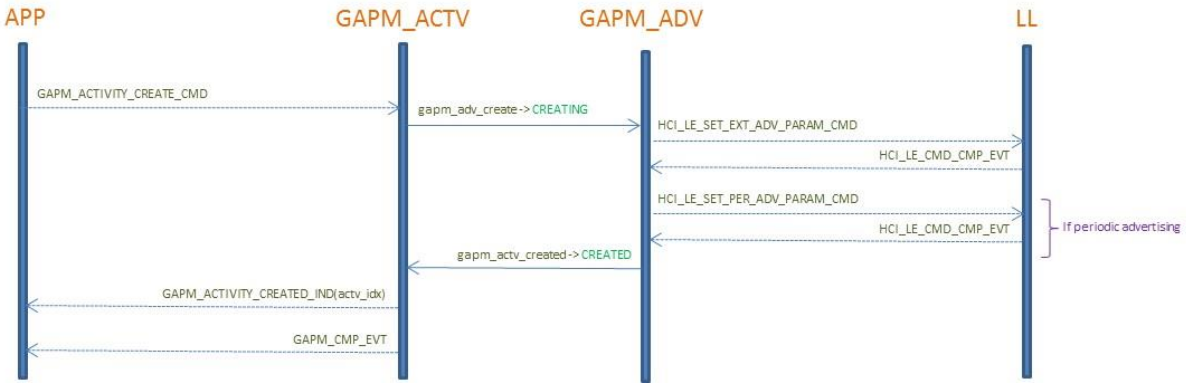

Figure 3-3: Advertising activity creation message sequence chart

Once created the advertising activity can be started using GAPM_ACTIVITY_START_CMD message. Even if advertising data or periodic advertising is empty it has to be set using GAPM_SET_ADV_DATA_CMD (see 3.2.6). Start command will be rejected if the data has not been set.

It is also mandatory to set the Scan Response data for a scannable advertising.

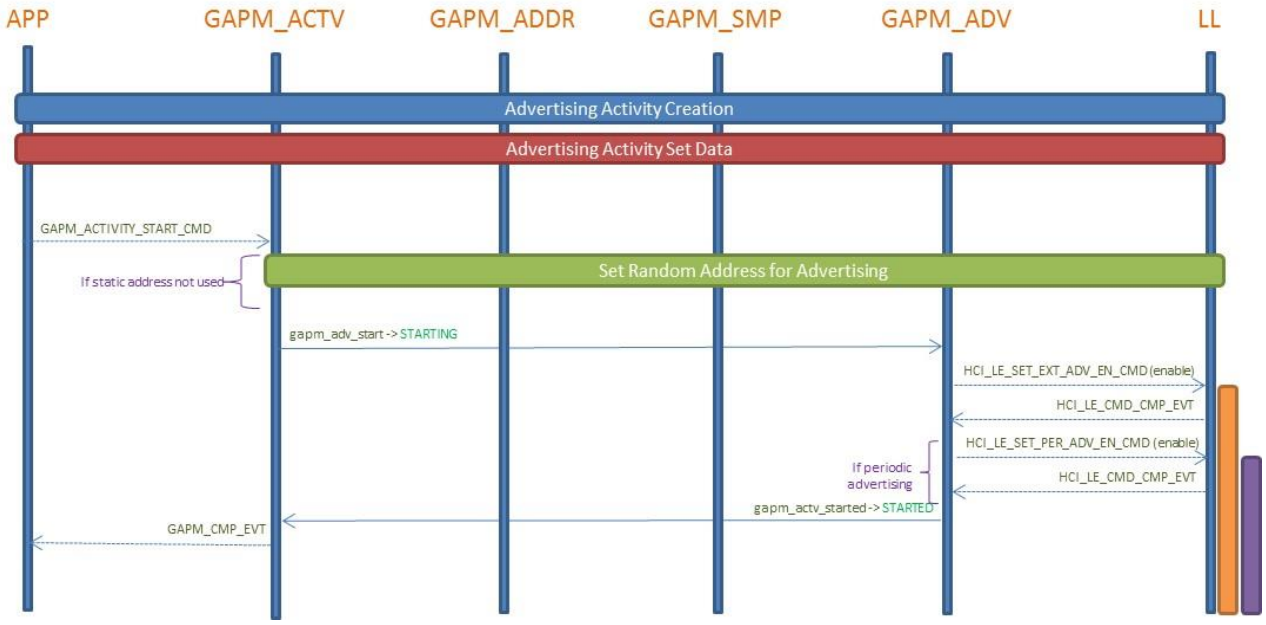

Figure 3-4: Advertising activity start message sequence chart

Application can request when it creates the advertising activity to be notified when a Scan Request packet is received. As shown in Figure 3-5, GAPM_SCAN_REQUEST_IND message is sent to the application after reception of scan request notification from the controller.

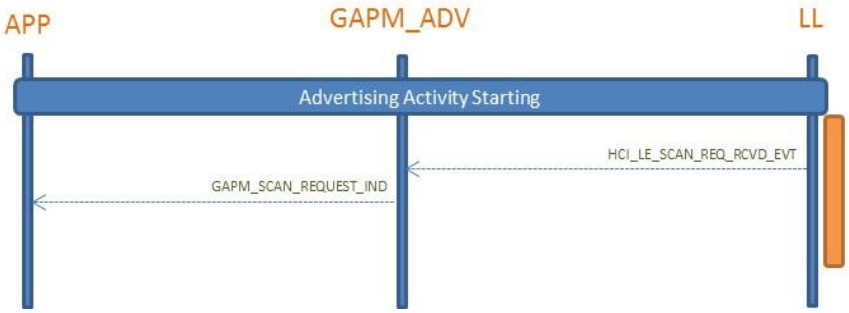

Figure 3-5: Advertising activity, scan request notification message sequence chart

Advertising duration can be limited using two different parameters, the duration and the maximum number of advertising events. For a legacy or an extended advertising, the application is notified about end of advertising by the reception of a GAPM_ACTIVITY_STOPPED_IND message (Figure 3-6).

Figure 3-6: Advertising activity, legacy/extended advertising stopped due to timeout message sequence chart

A periodic advertising is started in parallel with an extended advertising pointing to it and allowing a peer device to catch the synchronization. The periodic advertising (in violet on Figure 3-7) never timeout but extended advertising (in orange) can. In that case application is notified about the timeout using GAPM_ACTIVITY_STOPPED_IND but the message indicates that the periodic advertising is still running. Application can then restart extended advertising when needed using the GAPM_ACTIVITY_START_CMD.

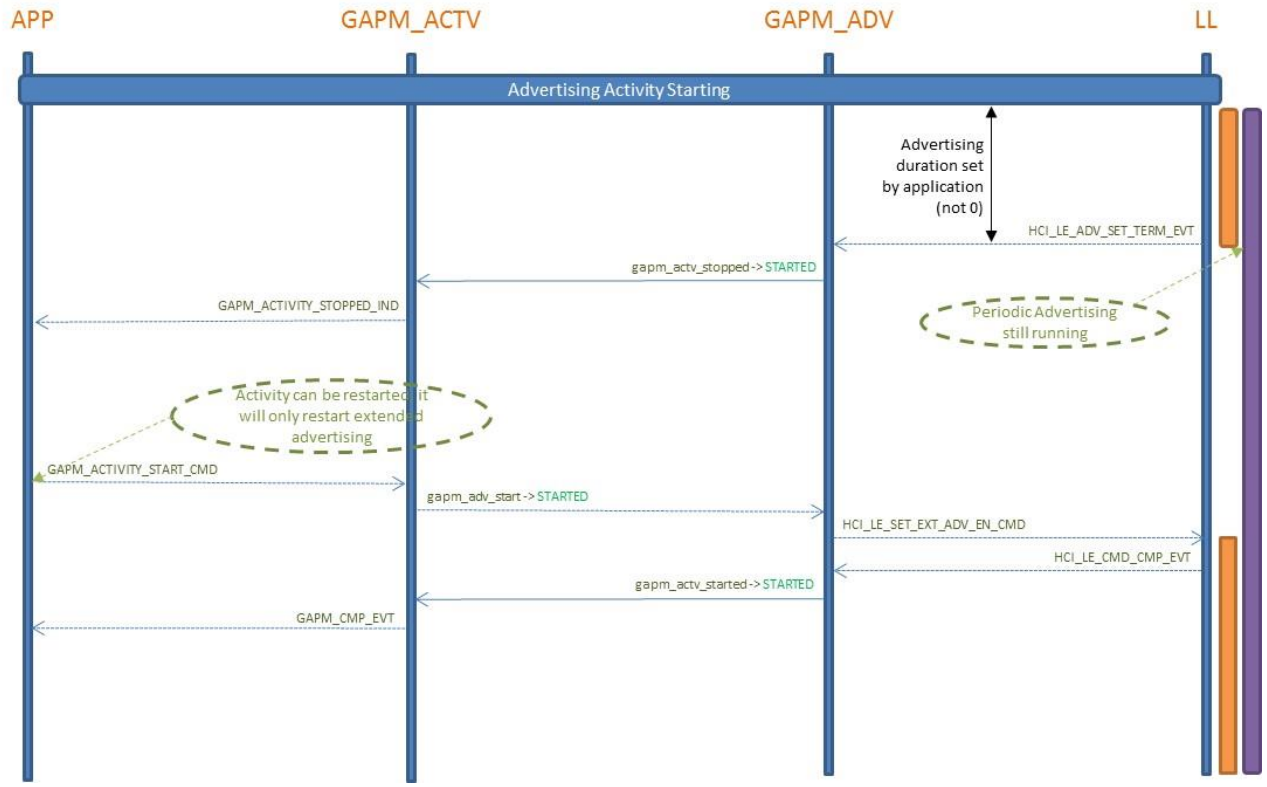

Figure 3-7: Advertising activity, periodic advertising stopped due to timeout message sequence chart

A connectable advertising activity can also be stopped after connection establishment. Upon reception of HCI LE Enhanced Connection Complete Event from controller a new instance of GAPC task is created, this task instance will be in charge of the newly creation connection.

Advertising activity is considered as stopped after reception of the HCI LE Advertising Set Terminated event (Figure 3-8).

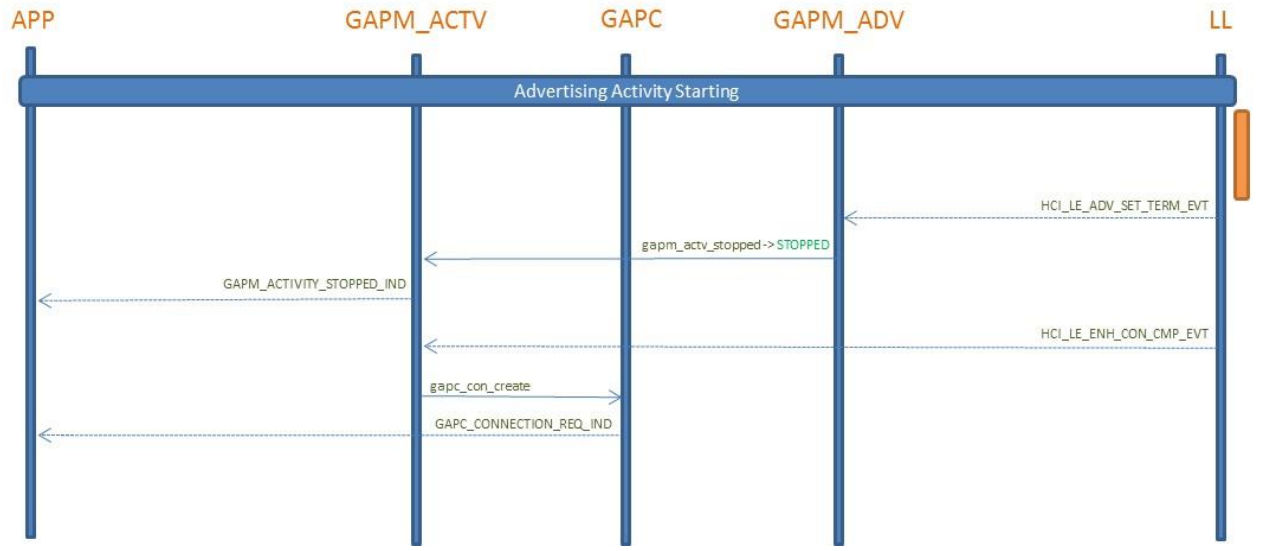

Figure 3-8: Advertising activity stopped due to connection establishment message sequence chart

A started advertising activity can be stopped at any time by the application by using the GAPM_ACTIVITY_STOP_CMD message as shown in Figure 3-9).

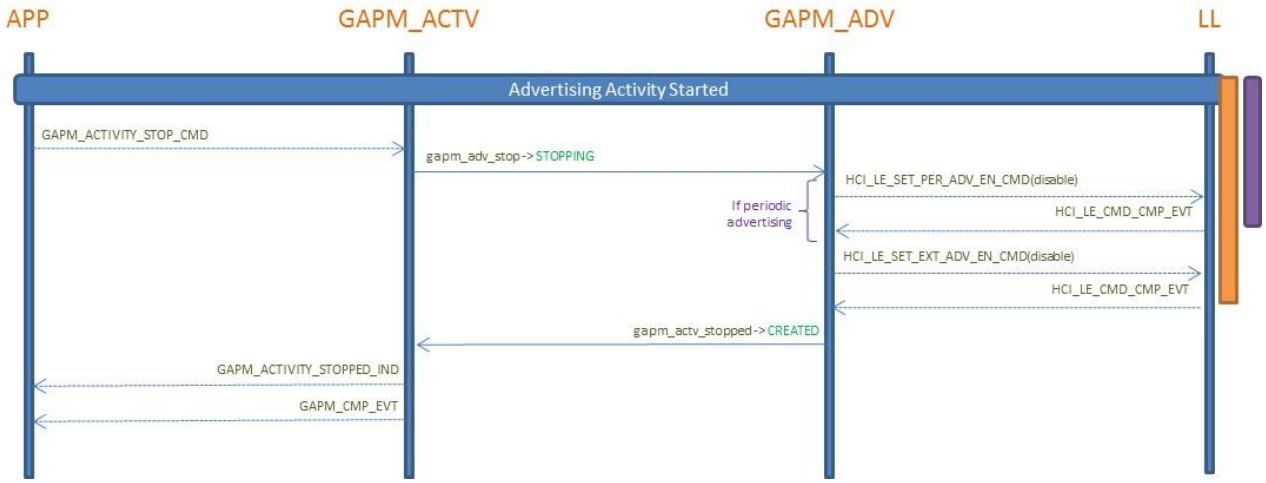

Figure 3-9: Advertising activity stopped upon application request message sequence chart

Once the activity has been stopped, it can be deleted (Figure 3-10) or it can be kept and restarted by the application.

Figure 3-10: Advertising activity deletion message sequence chart

3.2.6 Data Management

All kind of data sent over the air (advertising or Scan response or periodic advertising data) can be set once the advertising activity has been created and can also be updated on-the-fly by application using the GAPM_SET_ADV_DATA_CMD when activity has already been started.

Fragmentation of advertising data over HCI is handled by the GAP.

Note 1: *When advertising procedure is in connectable or scannable mode, or advertising procedure is on-going, it's not possible to set advertising data in more than one air fragment. It means that application cannot set advertising data which exceed size of an air fragment.*

Note 2: For legacy advertising mode it's not possible to have more than 31 bytes of advertising data or scan response data.

The advertising data must follow format described in core specification (Vol 3, Part C, Chapter 11 see [1]). The AD Type Flag must not be present in advertising data; this is added to advertising data by GAPM according to the discoverable mode.

When starting advertising activity, if advertising data has not been initialized by application, an error will be triggered.

An API is available in order to know size of non-connectable data that can be set by application.

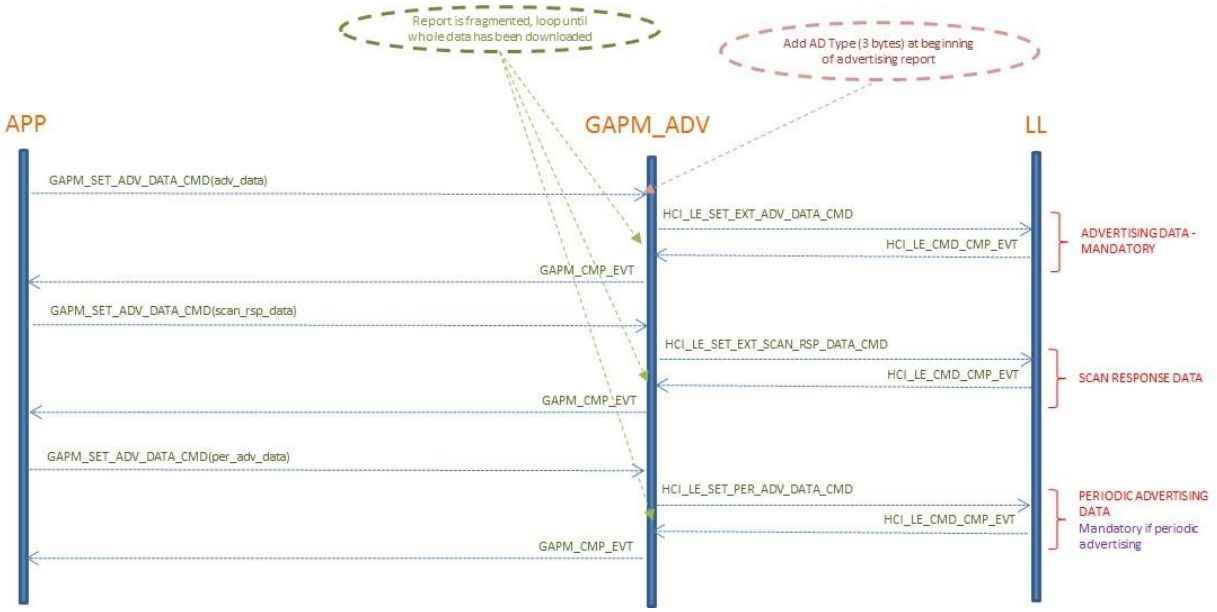

Figure 3-11: Set advertising data message sequence chart

3.3 Scanning activity

Purpose of scanning activity is the reception of advertising packets. This implies that support of at least Observer role is mandatory for creation of a scanning activity. Scanning activities are managed by GAPM SCAN module.

Six scanning modes are available:

- **Observer Mode**: Passive or an active scan procedure with non-limited duration. In this mode, the application is notified about any received advertising data whatever its type.
- **Selective Observer Mode**:
- **General Discovery**: Passive or an active scan procedure with a limited duration. In this mode, device is able to discover advertiser devices broadcasting data in limited or general discoverable mode. Do not use the white list.
- **Limited discovery**: Passive or an active scan procedure with a limited duration. In this mode, device is able to discover advertiser devices broadcasting data in limited discoverable mode. Do not use the white list.
- **General Connectable discovery:** Discover all connectable devices.
- **Selective Connectable discovery:** Discover connectable devices using white list filtering.

Note than due to content of HCI LE Extended Set Scan Param/Enable commands, a scanning activity cannot be started in parallel with another scanning activity.

Scan can be performed on LE 1M PHY or LE Coded PHY or on both PHYs in parallel.

Following figure describes creation of the scanning activity. This step is very simple because the lower layers are not involved in this procedure.

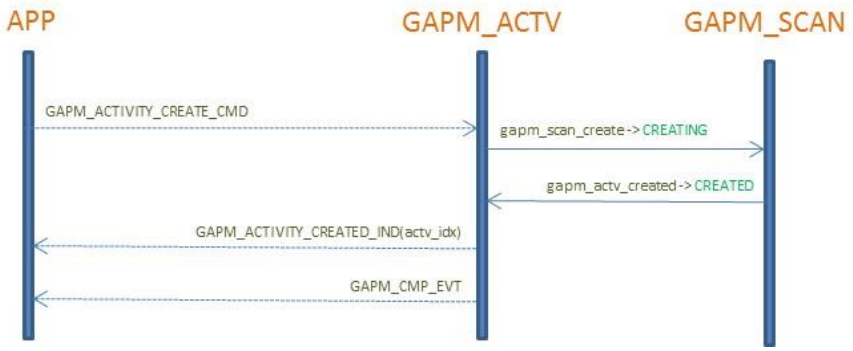

Figure 3-12: Scanning activity creation message sequence chart

Once activity has been started (Figure 3-13), GAPM SCAN module is responsible for reassembly of received fragmented reports (Figure 3-14).

Reports received from several advertiser can be reassemble in parallel. Controller is responsible for ensuring than for each first fragment with INCOMPLETE MORE TO COME status provided to the host, a fragment with COMPLETE status will also be provided in order to prevent the host to be stocked in a waiting state for the last fragment of a report.

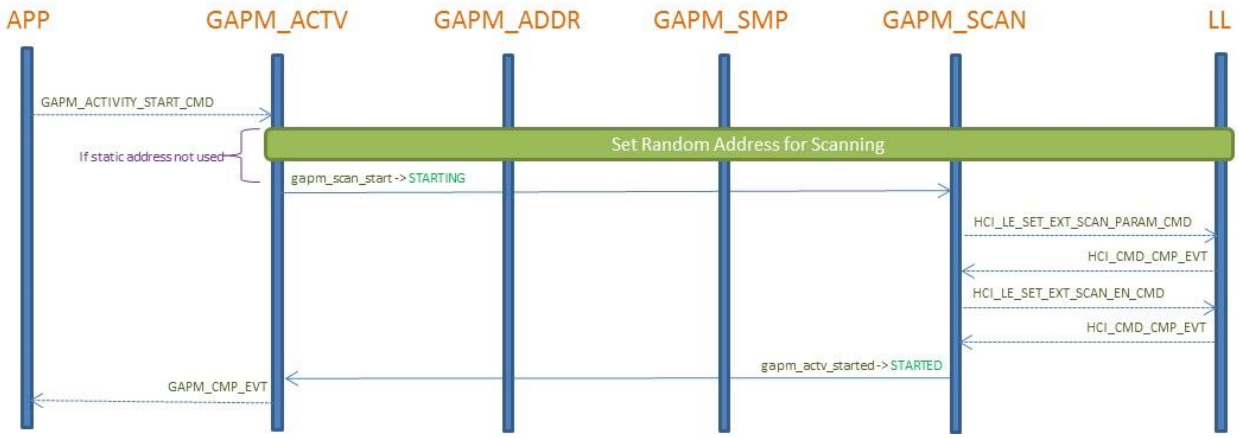

Figure 3-13: Scanning activity creation message sequence chart

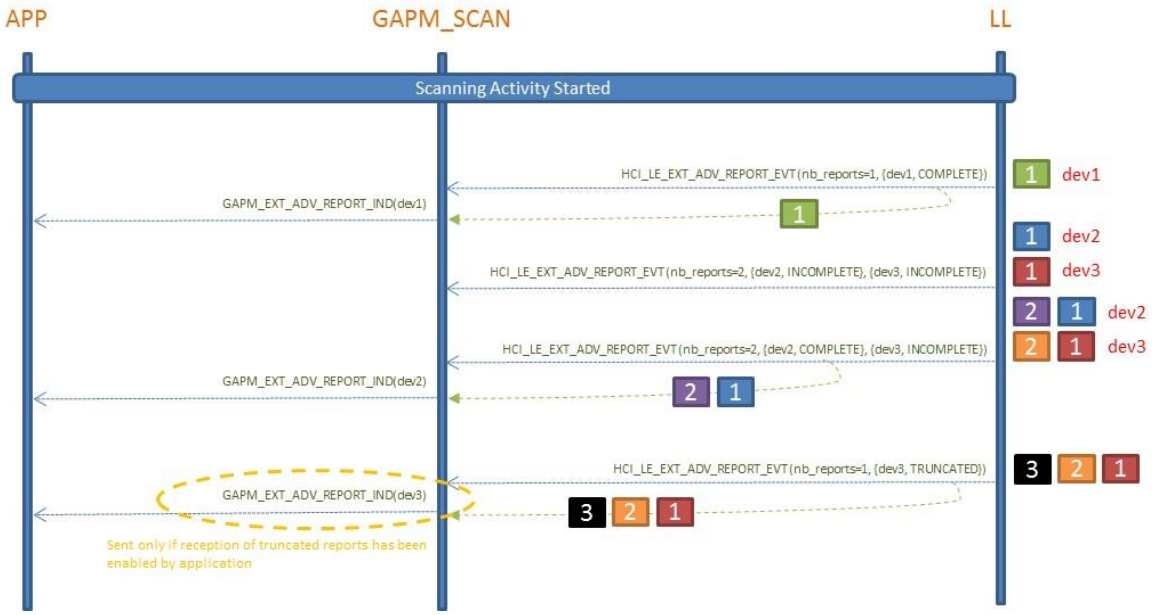

Figure 3-14: Scanning activity, reception of report fragments message sequence chart

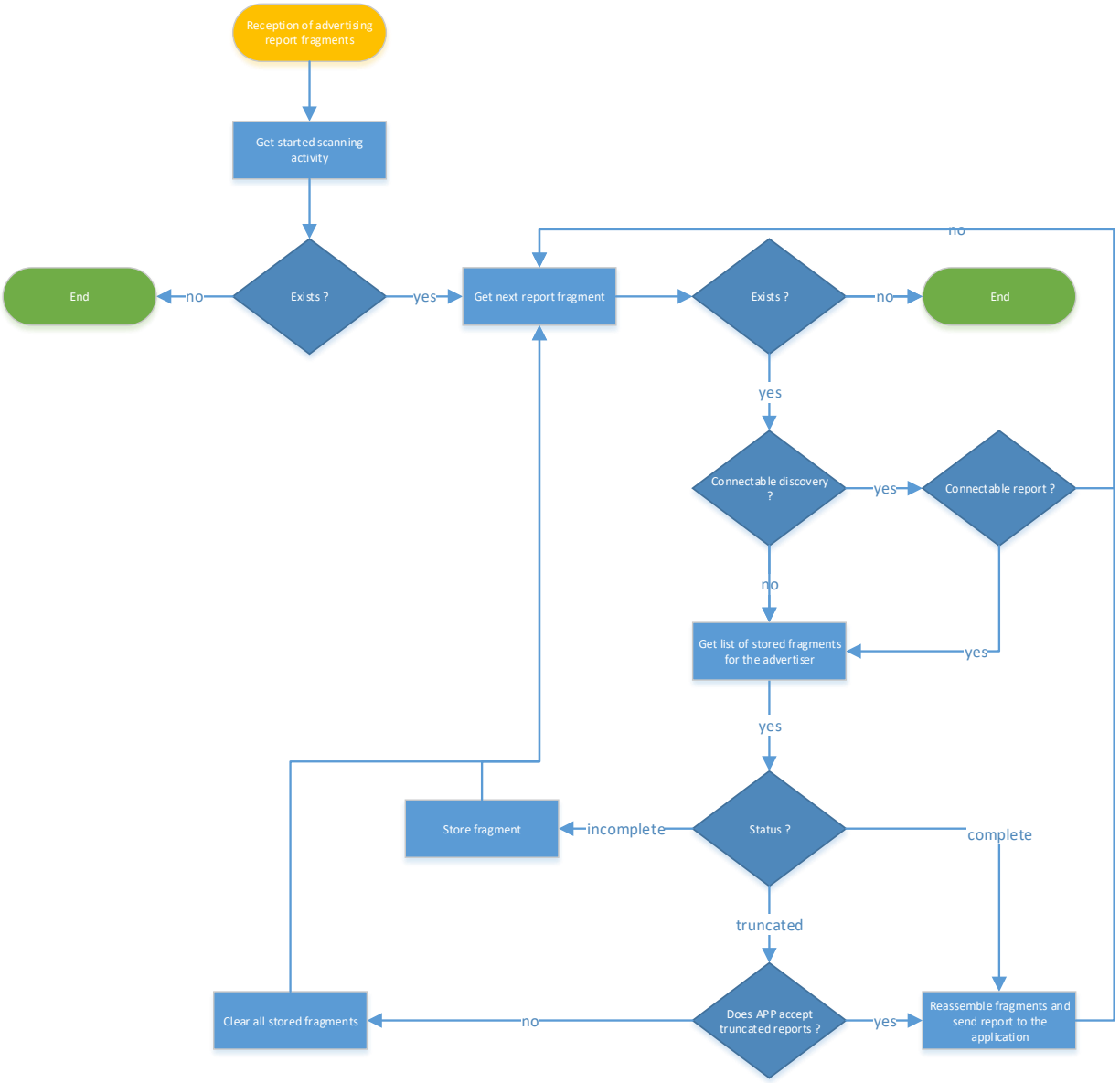

Figure 3-15: Scanning activity, reception of report fragments decision tree

Scan activity can be stopped either after a timeout (if a non-null duration is provided in the GAPM_ACTIVITY_START_CMD) or upon application request using the GAPM_ACTIVITY_STOP_CMD as shown in the following two figures.

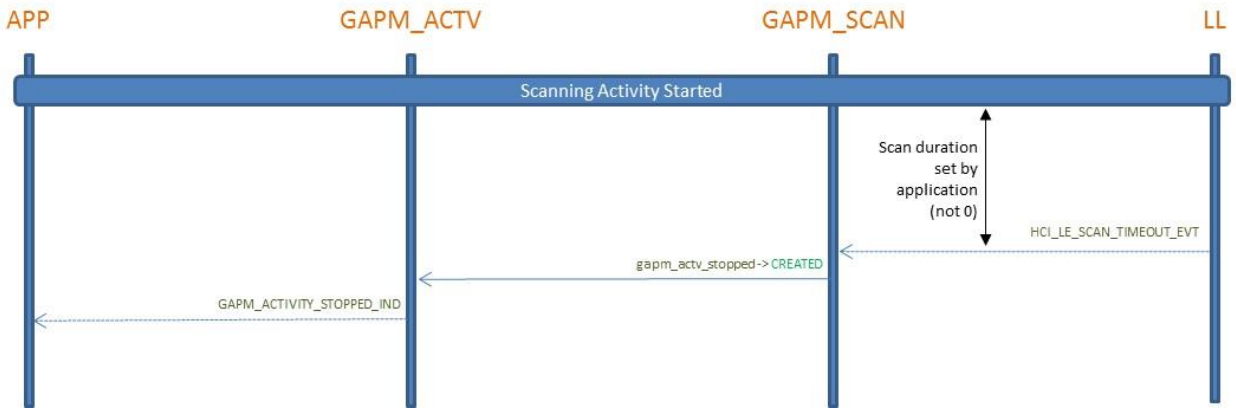

Figure 3-16: Scanning activity stopped due to scan timeout message sequence chart

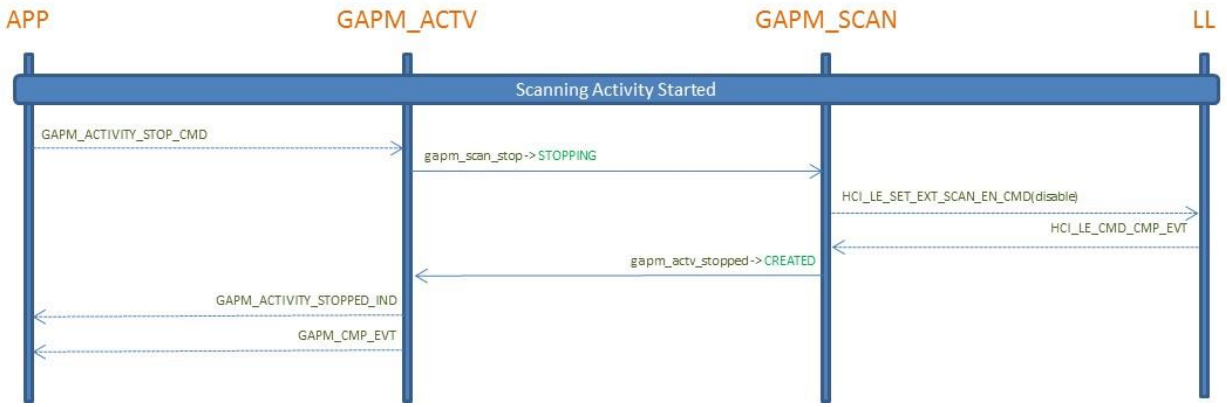

Figure 3-17: Scanning activity stopped due to application request message sequence chart

3.4 Initiating activity

Purpose of initiating activity is the establishment of a BLE connection as master. This implies that support of Central role is mandatory for creation of an initiating activity. Initiating activities are managed by GAPM INIT module.

Three connection modes are available:

- **Direct Connection Establishment:** Initiates a connection with a specific device.
- **Automatic Connection Establishment:** This procedure makes use of the white list to establish a connection with already known devices. Application shall first set the white list before starting this activity (see 4). As soon as connection with one of the devices indicated in the while list is established, the activity autonomously restarts the connection establishment procedure for the next device until all desired connection have been established.
- **Name Discovery:** As soon as connection is established, it performs a read of Device name characteristic (GATT UUID 0x2A00) and finally disconnects. Read device name is sent to the application by using GAPM_PEER_NAME_IND.

Based on content of HCI LE Extended Creation Connection command, only one initiating activity can be started at a given time.

Following figure describes creation of the initiating activity. This step is very simple because the lower layers are not involved in this procedure.

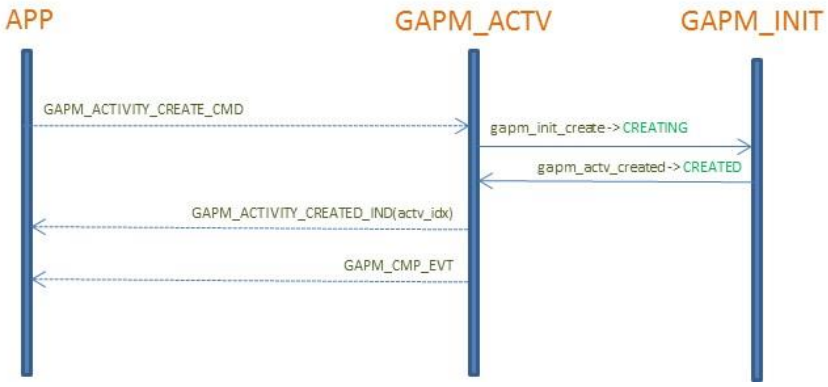

Figure 3-18: Initiating activity creation message sequence chart

No timeout duration is provided for direct connection establishment and name discovery connection modes. It is up to the application to stop the initiating activity by using the GAPM_ACTIVITY_STOP_CMD message.

For automatic connection establishment a timeout duration can be set in the GAPM_ACTIVITY_START_CMD. Once this duration has expired, initiating activity is stopped even if not all connections have been established.

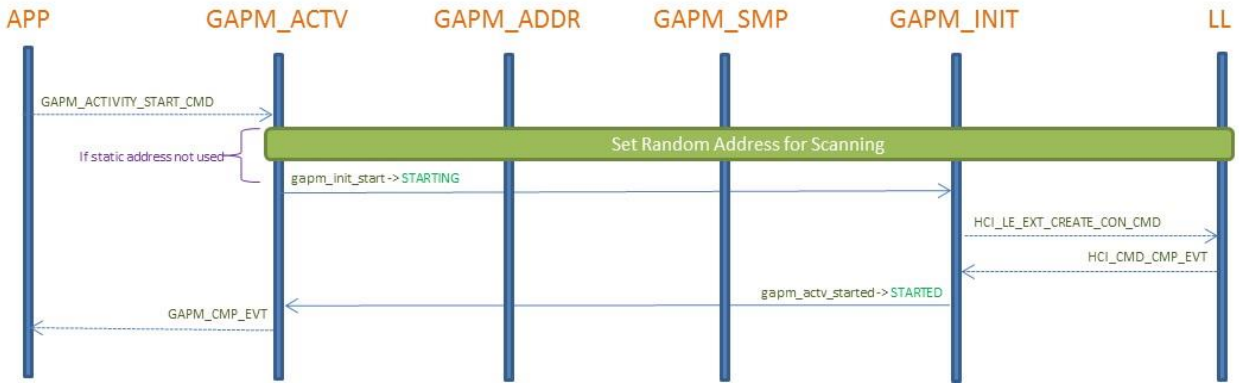

Figure 3-19: Initiating activity start message sequence chart

Each time a connection establishment is reported by the controller, a new instance of GAPC task is created. This task instance will be responsible for the created connection. Each time a connection is created, GAPC_CONNECTION_REQ_IND message containing connection parameters is sent to the application which is supposed to respond with GAPM_CONNECTION_CFM message. This message accepts or reject the connection and provides bond data stored for the connection device if any.

For a Direct Connection Establishment, the initiating activity is considered as stopped as soon as connection establishment is reported by the controller.

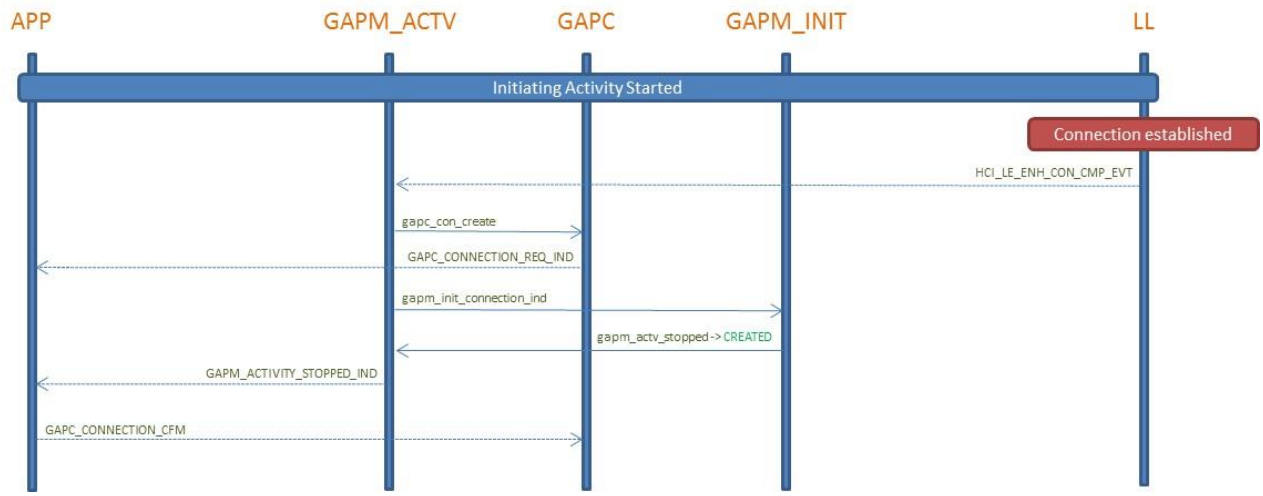

Figure 3-20: Initiating activity, direct connection establishment message sequence chart

For an Automatic Connection Establishment, the initiating activity is considered has stopped either once all connections have been established or upon expiration of timeout if set by the application in the GAPM_ACTIVITY_START_CMD message.

Note that starting of Automatic Connection Establishment is rejected if the white list is empty.

Figure 3-21: Initiating activity, automatic connection establishment message sequence chart

For Name Discovery, application is never informed about connection establishment. This connection/disconnection is entirely managed by GAP so that it can be transparent.

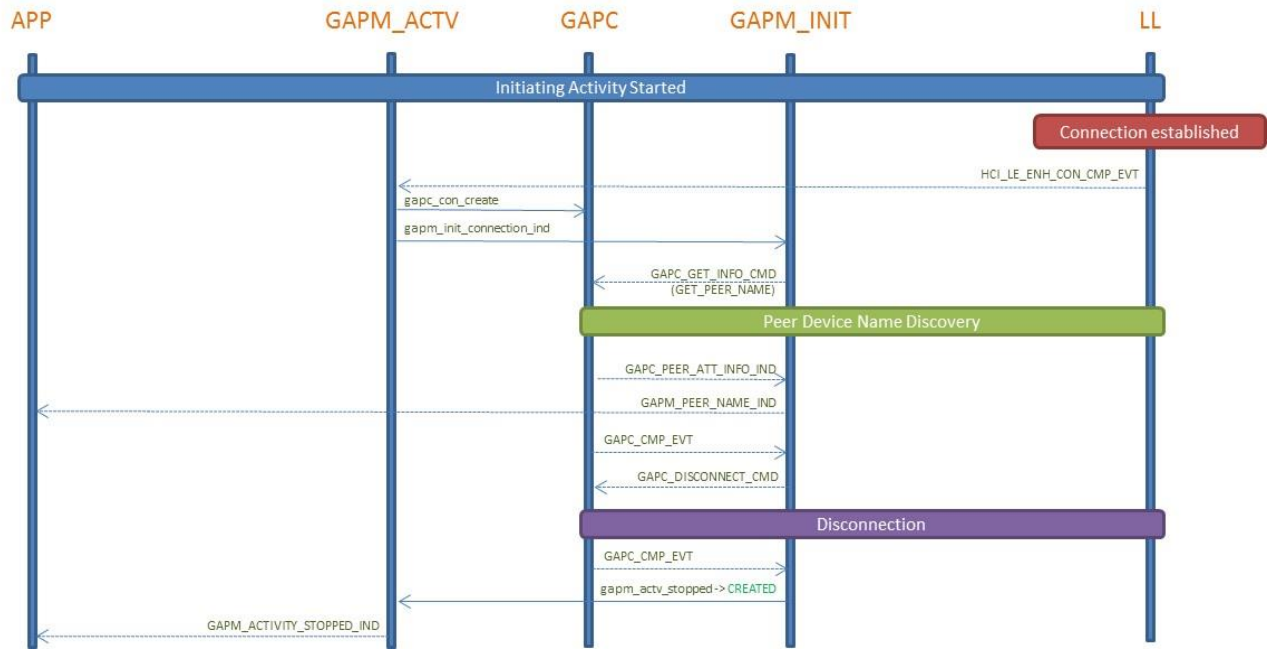

Figure 3-22: Initiating activity, name discovery message sequence chart

3.4.1 Auto Connection Establishment Procedure

As stated in [8], Vol 3, the auto connection establishment procedure allows the Host to configure the Controller to autonomously establish a connection with one or more device in the directed connectable mode or the undirected connectable mode by using the white list filtering.

In order to initiate this procedure the application must perform following operations:

- Set the content of the White List with list of device addresses by using GAPM_LIST_SET_CMD message
- Create an initiating activity by using GAPM_ACTIVITY_CREATE_CMD message
- Start the initiating activity with 'Automatic Connection Establishment' type by using GAPM_ACTIVITY_START_CMD message.

3.4.2 General Connection Establishment Procedure

As stated in [8], Vol 3, the general connection establishment procedure allows the Host to establish a connection with a set of known devices in the directed connectable mode or the undirected connectable mode.

In order to initiate this procedure the application must perform following operations:

- Create a scanning activity by using GAPM_ACTIVITY_CREATE_CMD message
- Start the scanning activity with either 'General Connectable Discovery' mode by using GAPM_ACTIVITY_START_CMD message
- Once report sent by expected device is received (GAPM_EXT_ADV_REPORT_IND), stop scanning activity by using GAPM_ACTIVITY_STOP_CMD
- Create an initiating activity by using GAPM_ACTIVITY_CREATE_CMD message
- Start the initiating activity with 'Direct Connection Establishment' type by using GAPM_ACTIVITY_START_CMD message. Advertiser address must be the address of the device with which a connection has to be established.

3.4.3 Selective Connection Establishment Procedure

As stated in [8], Vol 3, the selective connection establishment procedure allows the Host to establish a connection a set of devices whose addresses are present in the white list.

In order to initiate this procedure the application must perform following operations:

- Set the content of the White List with list of device addresses by using GAPM_LIST_SET_CMD message
- Create a scanning activity by using GAPM_ACTIVITY_CREATE_CMD message
- Start the scanning activity with either 'Selective Connectable Discovery' mode by using GAPM_ACTIVITY_START_CMD message
- Once report sent by expected device is received (GAPM_EXT_ADV_REPORT_IND), stop scanning activity by using GAPM_ACTIVITY_STOP_CMD
- Create an initiating activity by using GAPM_ACTIVITY_CREATE_CMD message
- Start the initiating activity with 'Direct Connection Establishment' type by using GAPM_ACTIVITY_START_CMD message. Advertiser address must be the address of the device with which a connection has to be established.

3.4.4 Direct Connection Establishment Procedure

As stated in [8], Vol 3, the selective connection establishment procedure allows the Host to establish a connection a single peer device

In order to initiate this procedure the application must perform following operations:

- Create an initiating activity by using GAPM_ACTIVITY_CREATE_CMD message
- Start the initiating activity with 'Direct Connection Establishment' type by using GAPM_ACTIVITY_START_CMD message. Advertiser address must be the address of the device with which a connection has to be established.

3.5 Periodic synchronization activity

The periodic synchronization activity is used to perform a periodic advertising synchronization establishment procedure. Periodic synchronization activity is managed by GAPM PER SYNC module. Observer mode has to be supported for creation of a periodic synchronization activity.

In order to establish a synchronization, there is two possible solutions, waiting information from an existing link using periodic advertising sync transfer from a peer device, or without a connection. In such case, ***it is mandatory to start a scan activity in parallel***. This scan activity can be started before or after the periodic synchronization activity.

In case of a periodic sync transfer, activity must be started with connection index before expecting sync information from peer device.

Once the activity has been created, it can be started using the GAPM_ACTIVITY_START_CMD message to synchronize with either one specific device (general type) or any of the devices present in the periodic advertising list (selective type) (see 4).

Note that it is not allowed to have two periodic synchronization activities waiting for synchronization at a same time.

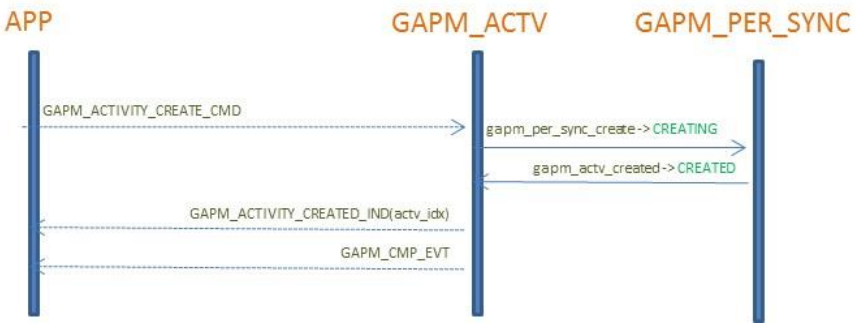

Figure 3-23: Periodic synchronization activity creation message sequence chart

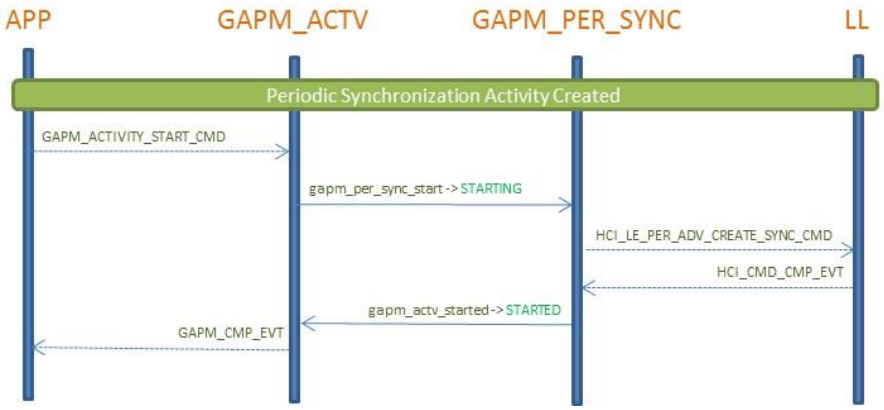

Figure 3-24: Periodic synchronization activity start message sequence chart

Once synchronization has been successfully established by the controller, a GAPM_SYNC_ESTABLISHED_IND is sent to application that can restart the activity in order to synchronize with another periodic advertiser. If no more synchronization has to be established, the scan activity can be stopped.

Received periodic advertising reports are forwarded to the application using GAPM_EXT_ADV_REPORT_IND message. GAPM PER SYNC is responsible for reassembling of received fragmented reports. Note that it is possible to receive truncated reports from the controller. Forwarding of such report to the application can be enabled or disabled when the activity is started.

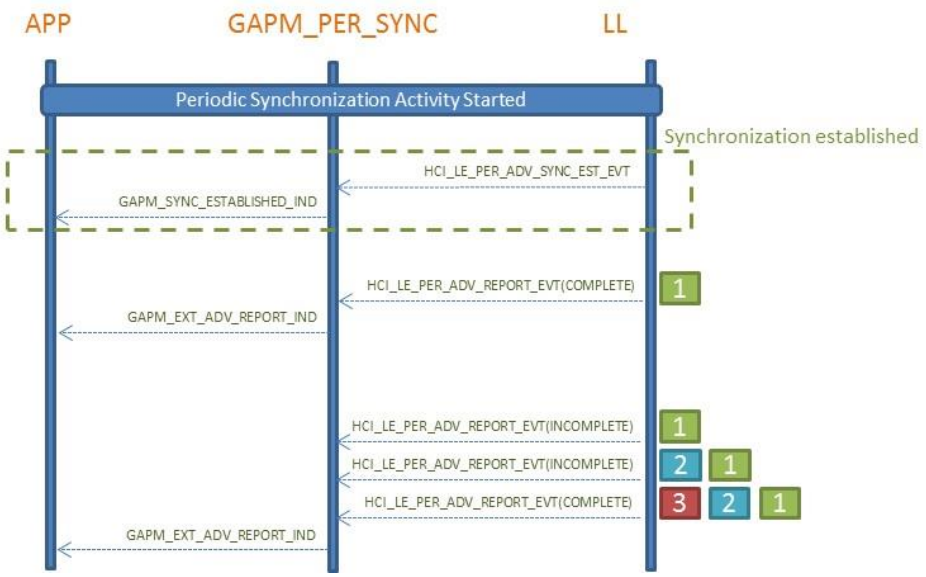

Figure 3-25: Periodic synchronization activity started, synchronization and report notification message sequence chart

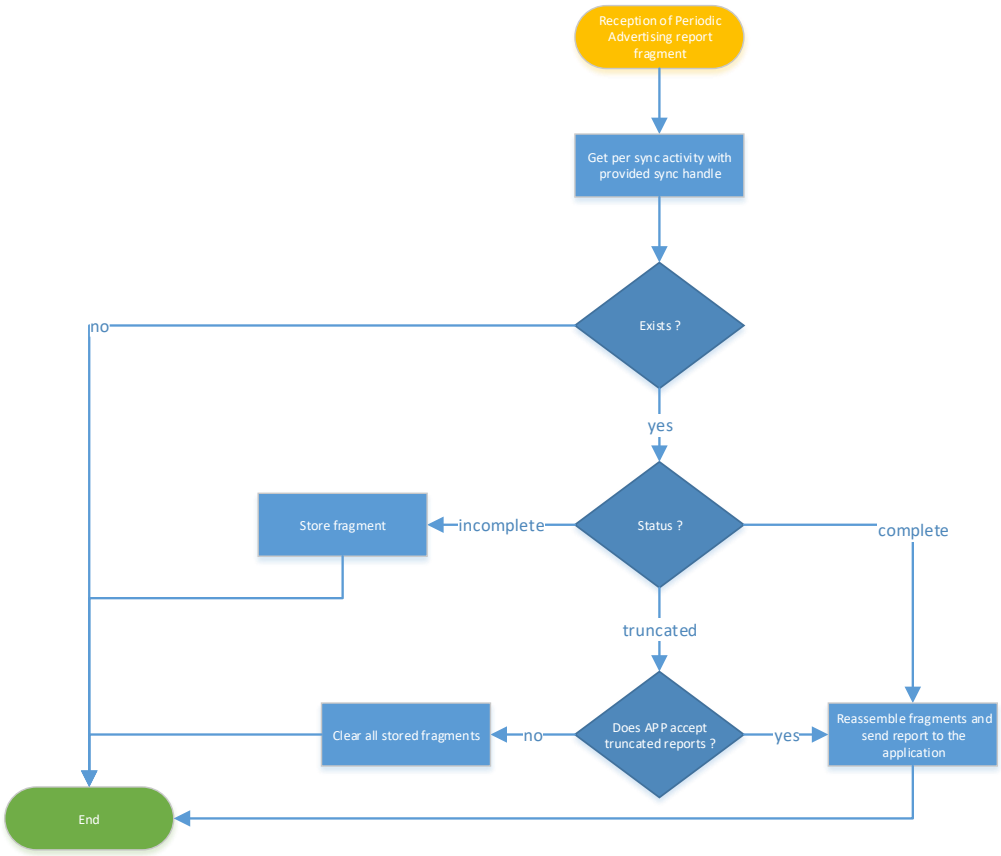

Figure 3-26: Periodic advertising report reception handling decision tree

If synchronization is lost, periodic synchronization activity is considered as stopped and application is notified using the GAPM_ACTIVITY_STOPPED_IND message.

As shown in figures below a synchronization attempt can be stopped by the application by using the GAPM_ACTIVITY_STOP_CMD message. This message can also be used at any time once synchronization has been established in order to terminate the synchronization.

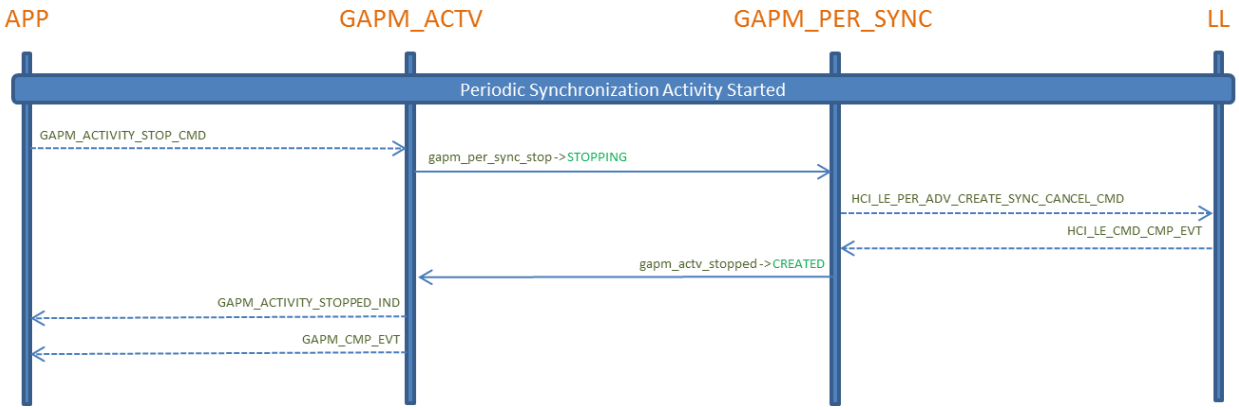

Figure 3-27: Periodic synchronization activity stopped by application (not synchronized) message sequence chart

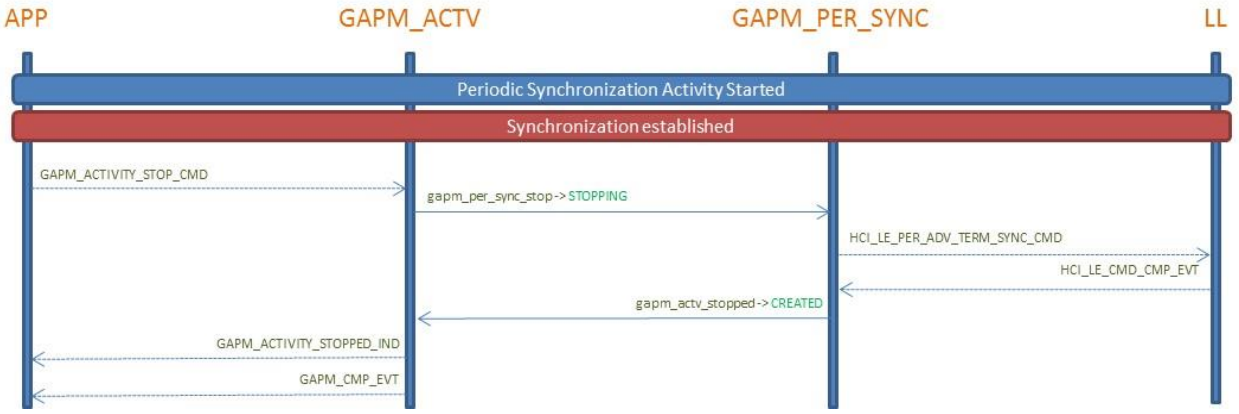

Figure 3-28: Periodic synchronization activity stopped by application (synchronized) message sequence chart

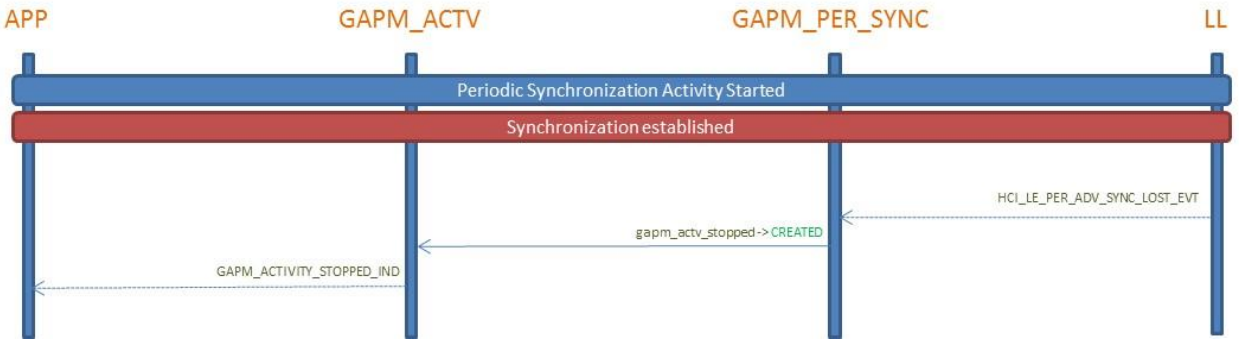

Figure 3-29: Periodic synchronization activity stopped after synchronization lost message sequence chart

4 Lists Management

A single API message is provided in order to manage the white list, the periodic advertiser list and the resolving list: GAPM_SET_LIST_CMD. This message does not allow adding or removing devices one by one. It is intended to be used to provide the whole content of the indicated list. It shall also be used for clearing content of a list.

Management of lists content is performed by GAPM LIST module.

Note that content of the lists can be set only if no activity is started.

Following figures expose the different operations performed by the host when upon reception of this message.

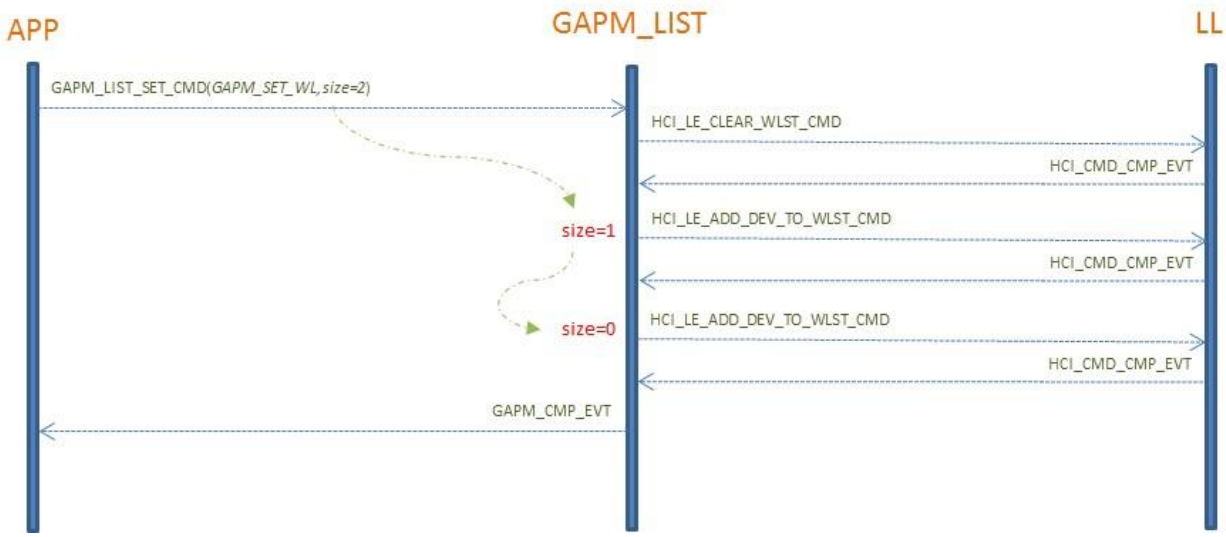

Figure 4-1: Set white list content message sequence chart

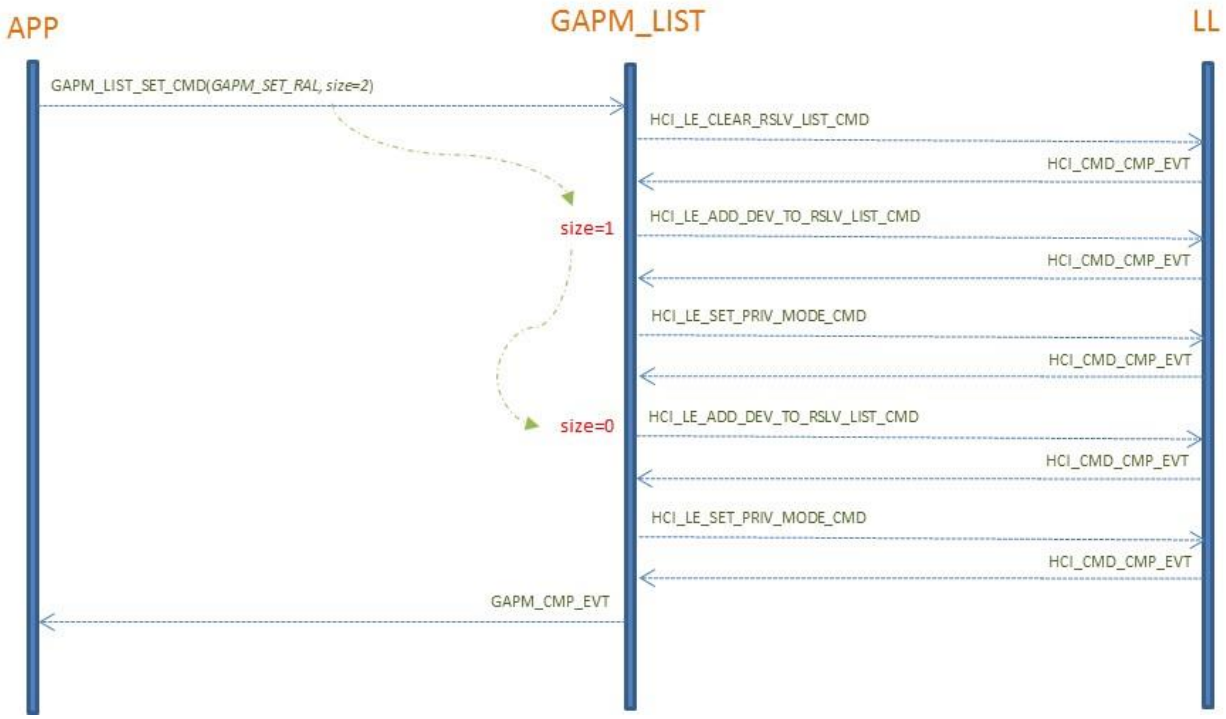

Figure 4-2: Set resolving list content message sequence chart

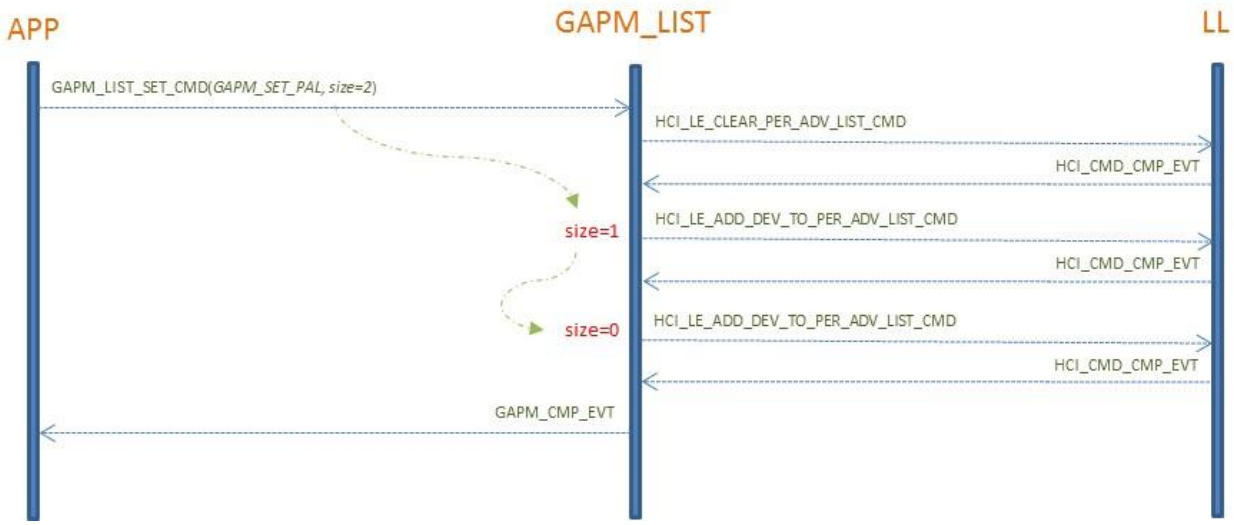

Figure 4-3: Set periodic advertiser list content message sequence chart

5 Random Addresses Management

An activity (except periodic synchronization activity that does not need to use an address) can use any one of the following address type:

- Static address (public or random)
- Private resolvable address
- Private non-resolvable address (only in non-connectable mode)

An advertising activity can also be configured so that it does not use an address (anonymous mode, only in non-connectable mode)

If a private address is used by an activity, it has to be renewed after a renewal period set at device configuration (GAPM_SET_DEV_CONFIG_CMD).

If controller privacy is enabled, the host will still generate private addresses for cases where the controller is no more able to find needed information for address generation in its resolving list. In that case the controller can use random addresses set by the host using either the HCI LE Set Random Address command (for scanning and initiating activities) or the HCI LE Set Advertising Set Random Address command (for advertising activities).

GAPM ADDR module is responsible for generating random addresses, providing the generated address to the controller by using the appropriate HCI command following the kind of activity for which address is used, renewing the random addresses once the renewal period has elapsed.

Each time a new private address is generated by the host for an activity, a GAPM_DEV_BDADDR_IND message is sent to the application.

Following schemes shows the state machine in charge of generating and setting an address upon GAPM ACTV module request and the messages sent in order to perform these operations.

Some considerations must be taken in account:

- Each advertising activity can have its own private random address (due to content of HCI LE Set Advertising Set Random Address command)
- If a scanning activity and an initiating activity are running in parallel and are using a private random address, they must share the same address. HCI LE Set Random Address command does not allow to make a difference between those two activities.

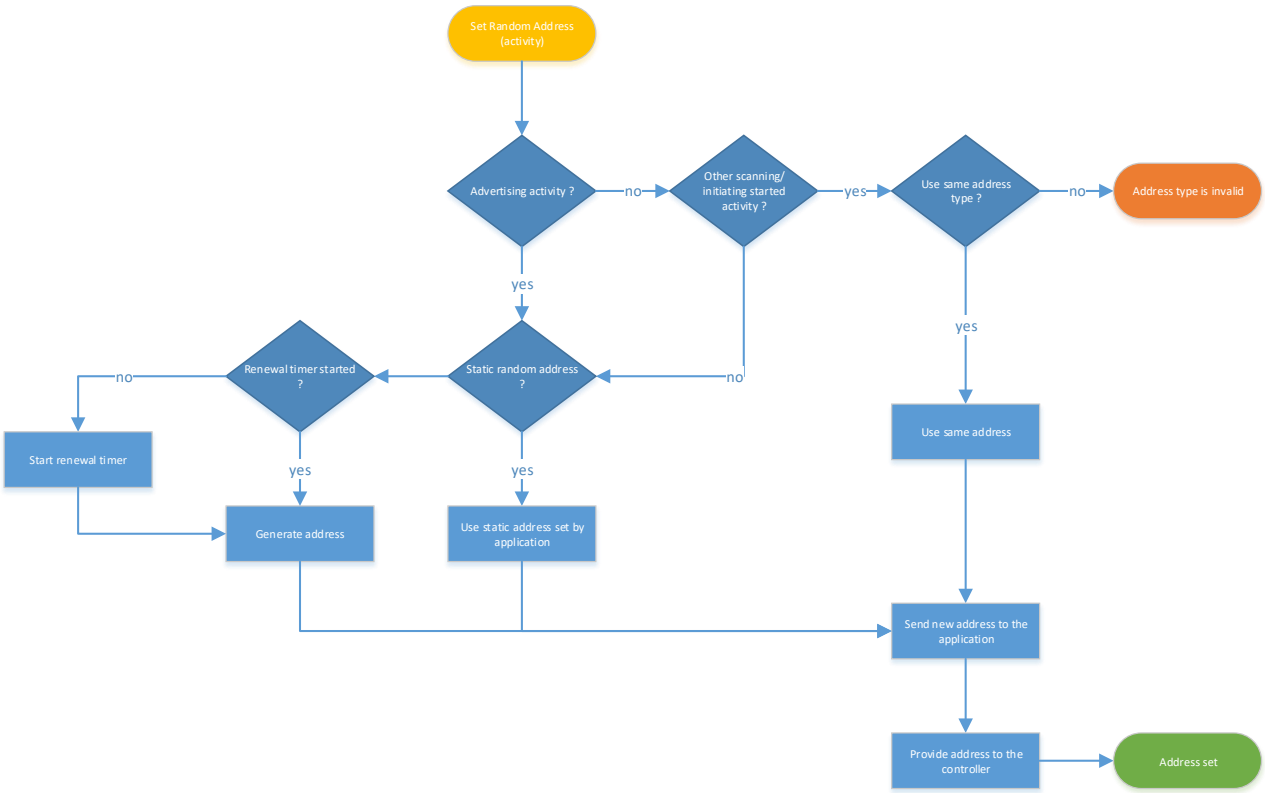

Figure 5-1: Set random address state machine

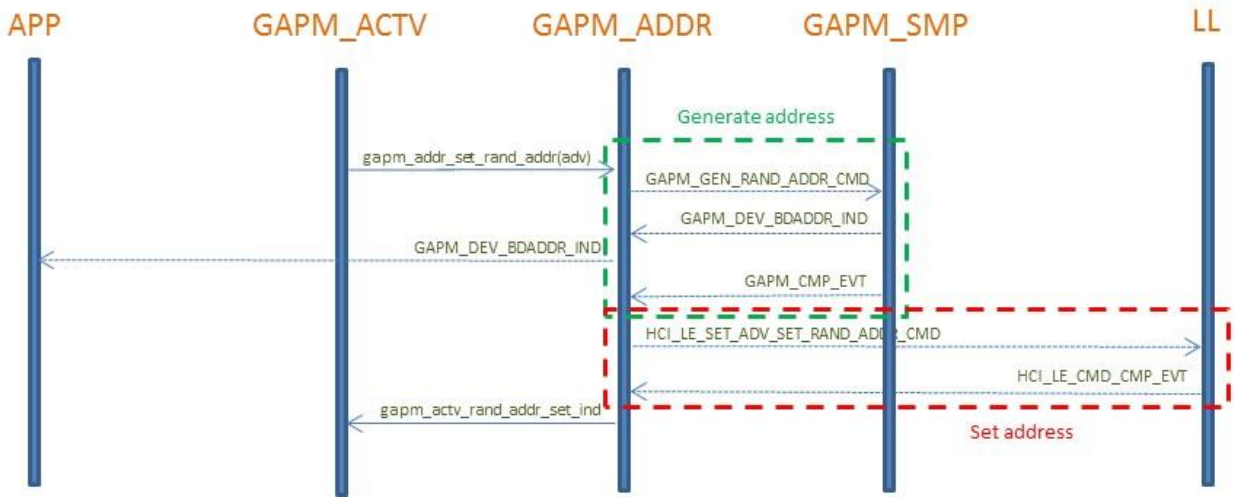

Figure 5-2: Set random address message sequence chart (advertising activity case)

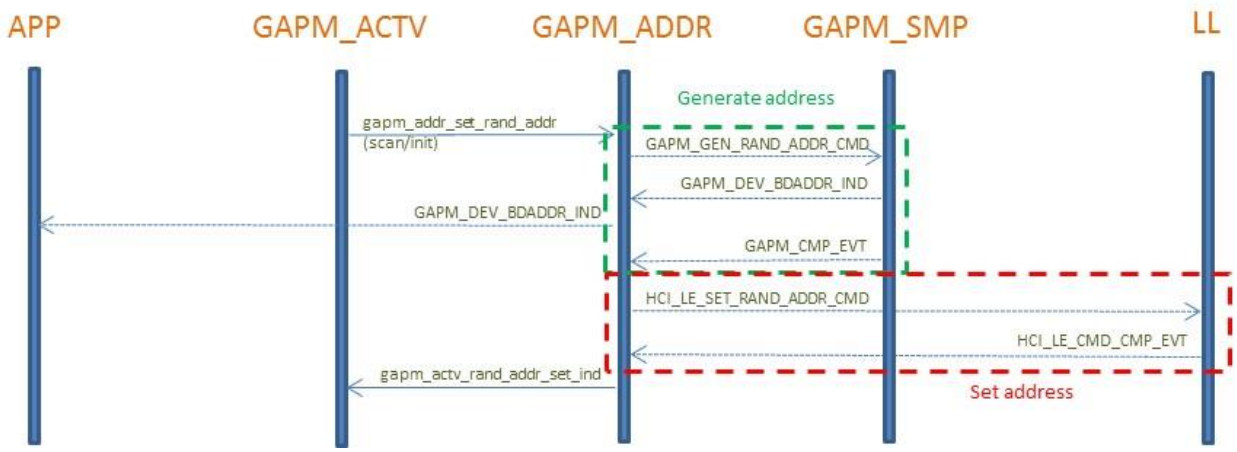

Figure 5-3: Set random address message sequence chart (scanning/initiating activity case)

Figure below describes the steps for address renewal procedure. Note that all private random addresses are updated at a same time in order to make the system easier. This means that the use duration of first generated address for a given activity can be smaller than renewal duration if a private random address is used by another activity.

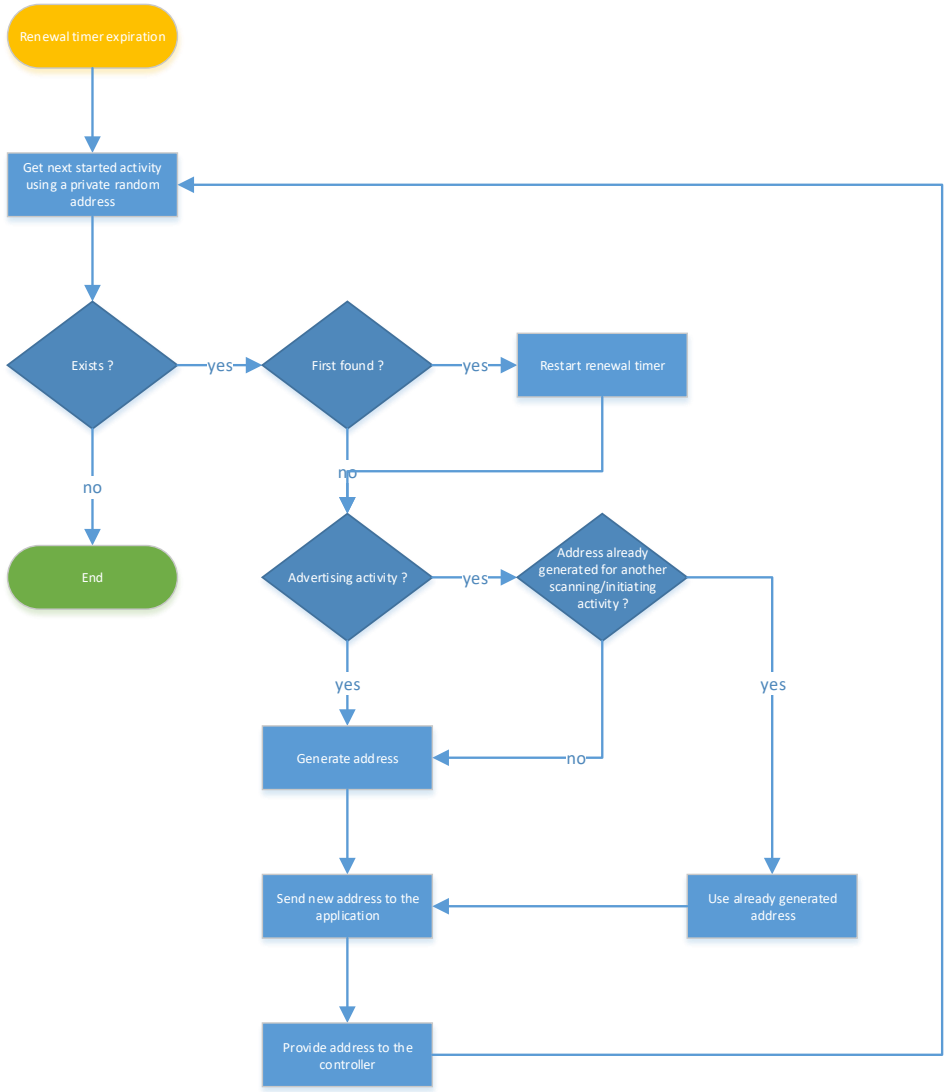

Figure 5-4: Random addresses renewal state machine

6 Connection management

6.1 Update connection Parameters

Operation that can be started over a LE link, Parameter update procedure is used to update link parameters.

If operation initiated by master on 4.0 (Legacy) devices (see Figure 6-1) there is no link negotiation and new link parameter are automatically applied.

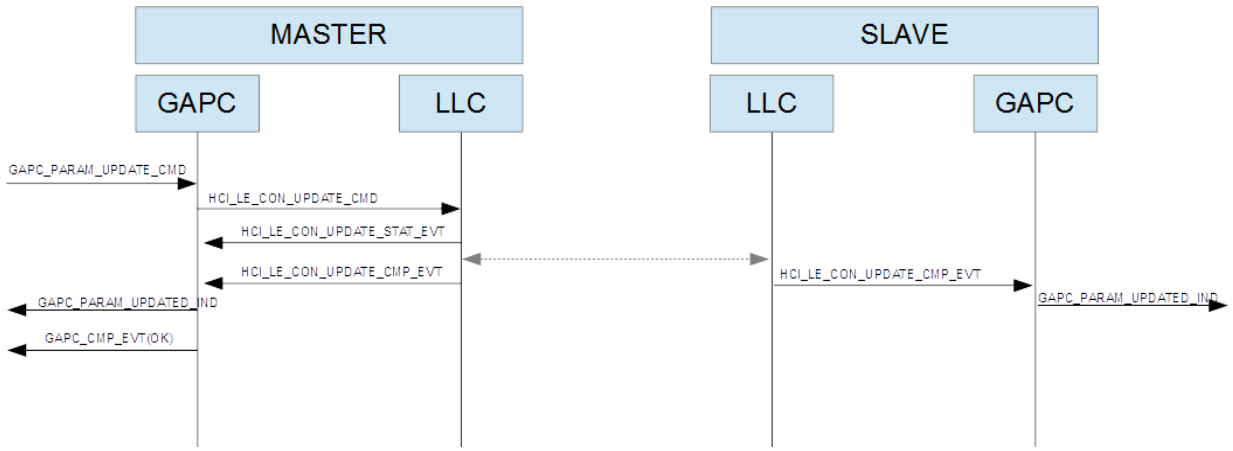

Figure 6-1: Parameter Update initiated by Master

When a slave initiate a connection parameter update without knowing remote features, first start a parameter update through HCI and if it fails due to following reason, use legacy negociation over L2CAP (see Figure 6-2):

- Unknown HCI Command
- Command Disallowed
- Unsupported Command
- Unknown LMP PDU
- Unsupported Remote feature
- LMP Pdu Not Allowed

Note: Operation completion message for a legacy parameter update initiated by slave (on slave device) have to be triggered before GAPM_PARAM_UPDATE_IND to ensure that master device did not use connection param request.

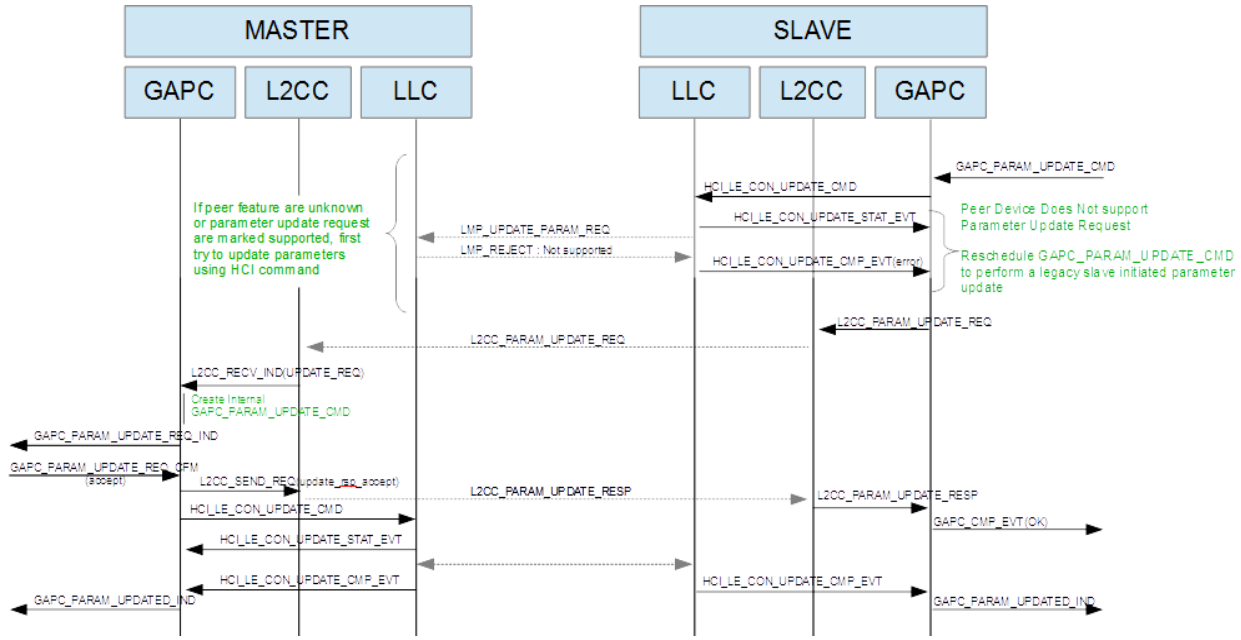

Figure 6-2: Legacy Parameter Update initiated by Slave

Figure 6-3 shows legacy parameter update negotiation initiated by slave and rejected by master.

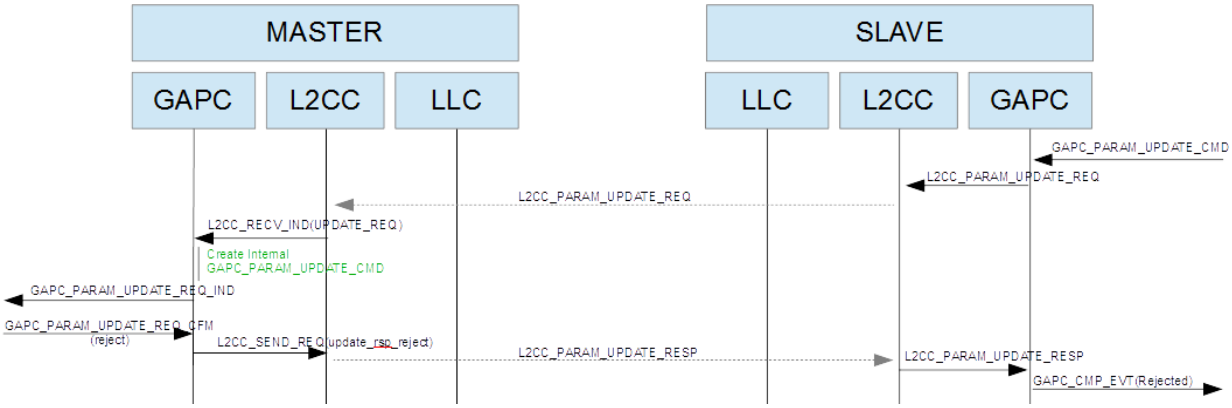

Figure 6-3: Legacy Parameter Update initiated by Slave, rejected by Master

If Parameter update request is supported by peer device, Connection parameter initiated by master or slave is a little bit different. It is requested for peer device to accept or reject new parameters. This Parameter update is only performed through LLCP even if it's initiated by slave or master of the link.

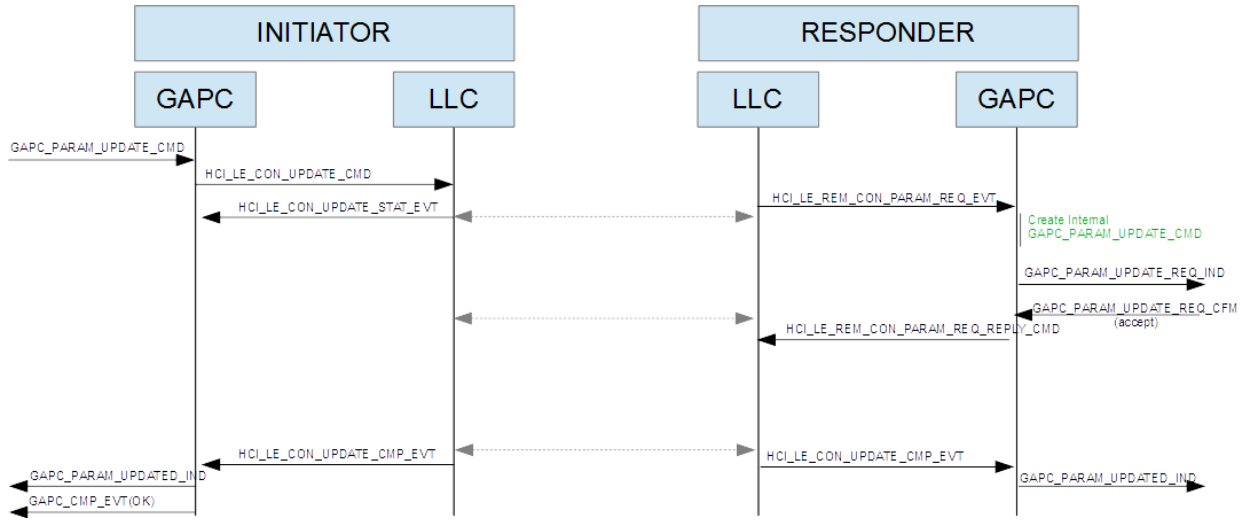

Figure 6-4: Parameter Update with remote update request support accepted by responder

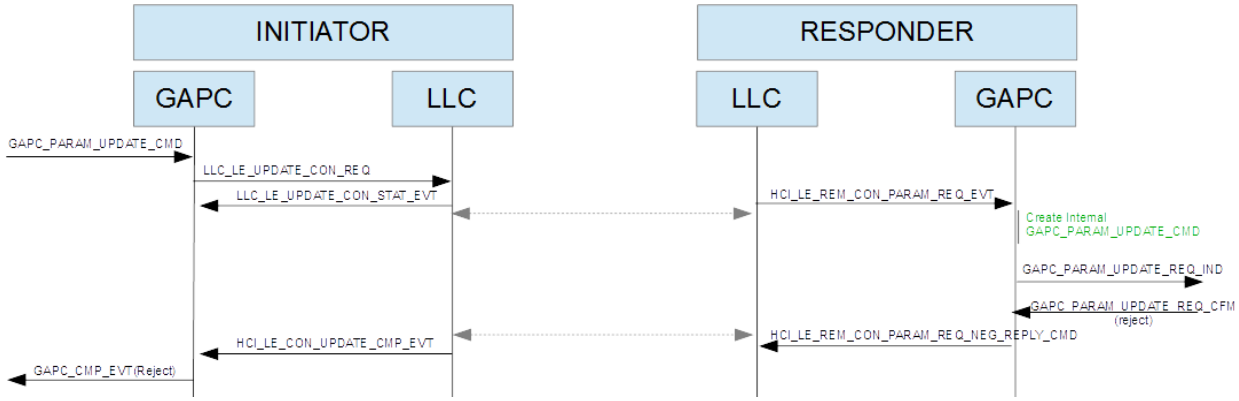

Figure 6-5: Parameter Update with remote update request support rejected by responder

6.2 Constant Tone Extension

Used for AoA/AoD feature, the HL implantation is straight forward. There is a simple mapping of HCI commands to HL API. Except for periodic advertising activity where the CTE transmission parameters can be set when activity is created.

6.3 Bonding

Bonding is the function where devices exchange and store security and identity information to create a secure relationship. It occurs at the first connection between devices or the first service that requires security or authorization.

Two types of bonding procedures are defined:

- Dedicated Bonding is when the user initiates SM pairing with the explicit purpose of creating a bond (i.e., a secure relationship) between two devices.
- General Bonding is when the user is requested to pair before accessing a service since the devices did not share a bond beforehand.

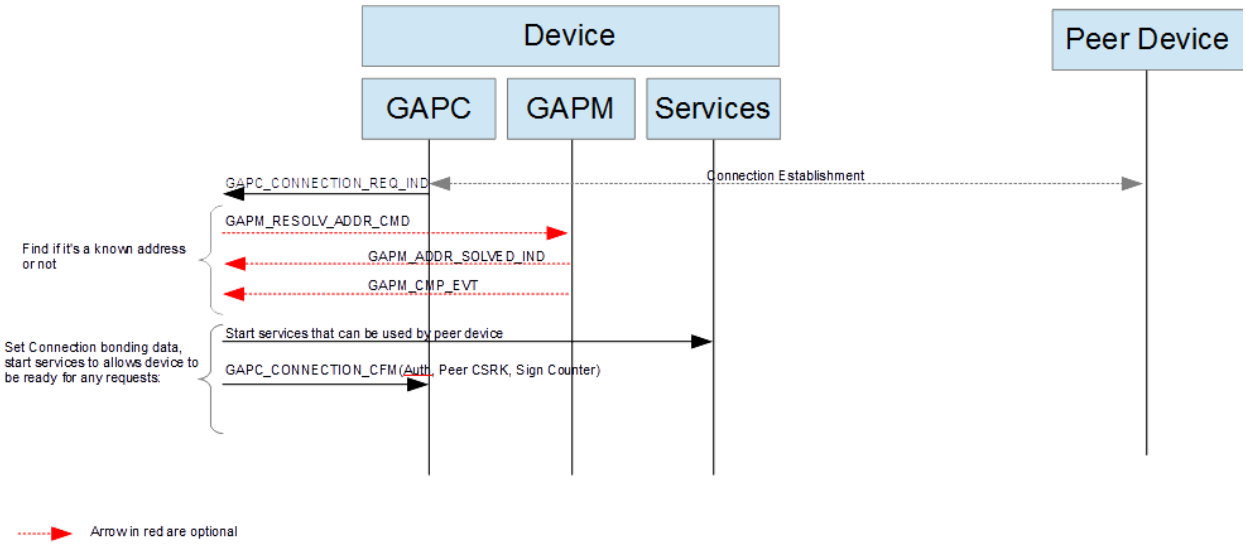

Figure 6-6: Connection establishment with a known device (recover bond data)

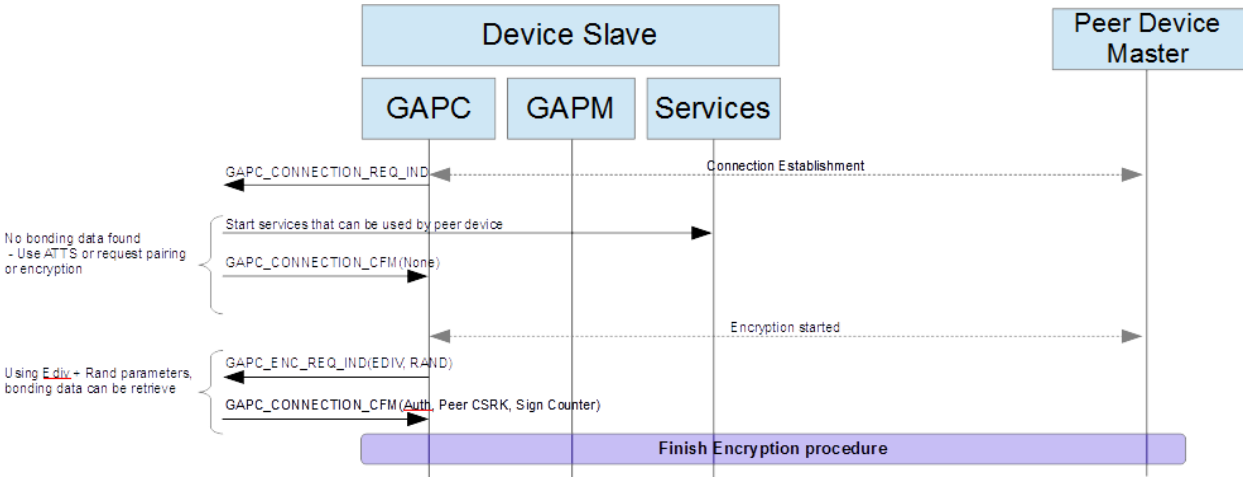

Figure 6-7: Recover bond data of a peer device with a random address.

7 Low Energy Security

Security mode and level defines the safety requirements of a device or access to services offered by the device.

7.1 Security Modes

The LE security is expressed in modes and levels. There are two security modes: Sec 1 (encryption) and Sec 2 (signing).

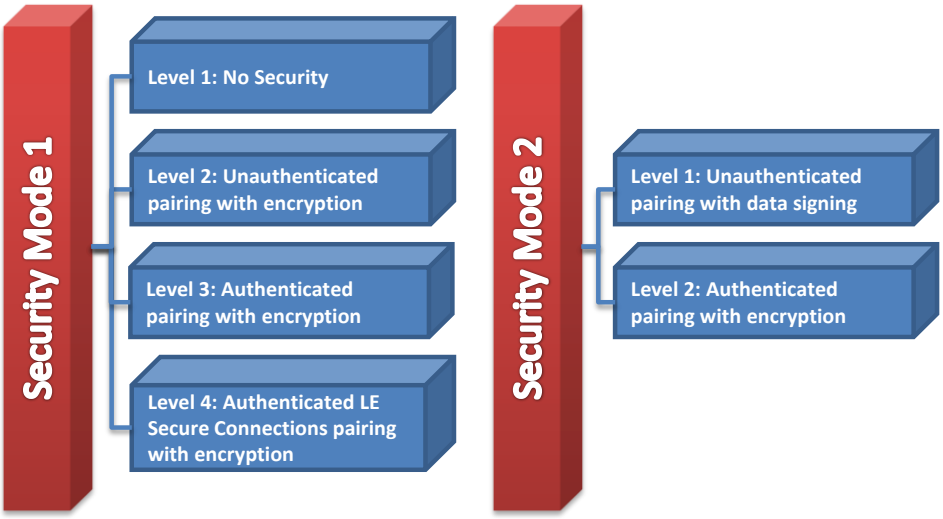

Figure 7-1: LE Security Modes

7.2 Authentication Procedure

The procedure pertains to satisfying the security requirements of the connecting devices when service request is initiated on either side. The authentication procedure is only valid after establishing LE link.
There are two types of pairing.

- Authenticated Pairing – Perform pairing procedure with authentication set to MITM protection
- Unauthenticated Pairing – Perform pairing procedure with authentication set to No MITM protection

7.3 Authorization Procedure

The procedure allows the continuation of service access by remote device. This is a confirmation by the user to continue with the procedure. Authorization may be granted after successful authentication.

7.4 Data Signing

The procedure is used to transfer authenticated data between two devices in an unencrypted connection. This is used by services that require fast connection setup and data transfer. If this is used, Security mode 2 is a must.

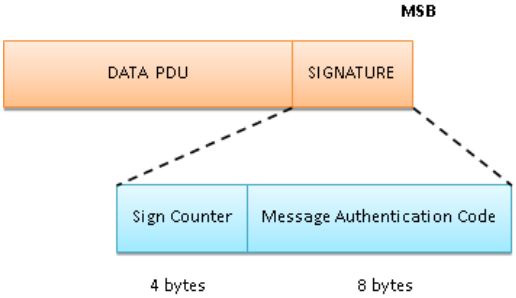

Figure 7-2: Packet Signature

7.5 Privacy

7.5.1 Host managed Privacy (1.1)

The host managed privacy feature provides specific level of security from an attacker to track an LE device over a certain period of time. This is an optional feature for all GAP roles.

	Broadcast	Observer	Central	Peripheral
Privacy Off	Public or Static	Public or Static	Public or Static	Public or Static
Privacy On - Connectable	N/A	N/A	Resolvable	Resolvable
Privacy On - Non Connectable	Resolvable or Non-Resolvable	Resolvable or Non-Resolvable	Resolvable or Non-Resolvable	Resolvable or Non-Resolvable

Table 7-1: Device address type according to privacy configuration

Note: For passive scan, Privacy feature is ignored

Note: If device has ***all roles***, it cannot use both Resolvable address for an air activity and Non-Resolvable address for another air activity. A privacy error will be triggered in that case.

7.5.2 Controller managed Privacy (1.2)

With controller managed privacy, the application should set the resolving address list (RAL) (see [1] Vol. 1 Part. A Chapter 5.4.5) using GAPM_RAL_MGMT_CMD command. This resolving address list can be managed like a white list and is used in complement of the white list.

When the controller managed privacy is enabled, scan, advertise and initiating parameters are set to use the resolving list.

If this feature is enabled when setting device configuration (see 15.2) then Central Address Resolution characteristic becomes present in the GAP service (see 13)

7.5.3 LE Address

There are two types of Bluetooth LE addresses

- Static Address
  - Two most significant bits are equal to 1
  - All the other bits are not “all 0s” nor “all 1s”
- Private Address
  - Non-resolvable Address
    - Two most significant bits are equal to 0
    - All the other bits are not “all 0s” nor “all 1s”
  - Resolvable Address
    - Two most significant bits are equal to 01
    - 22 remaining bits of prand are not “all 0s” nor “all 1s”
    - 24 bits hash part is derived from IRK, prand and ah func

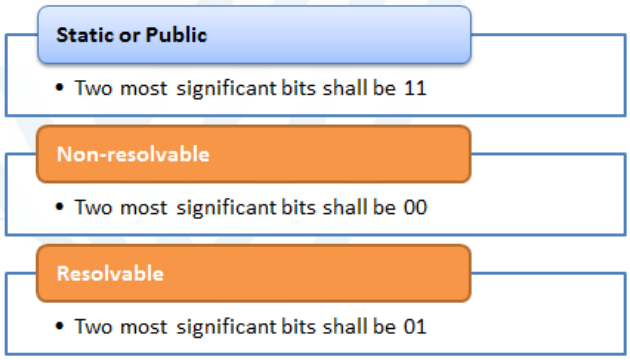

Figure 7-3: LE Address

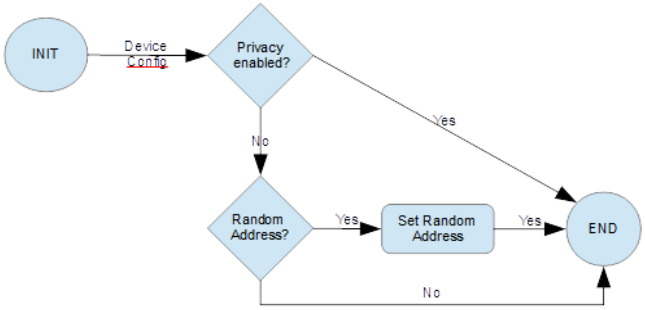

Figure 7-4: Initialize device address FW state machine

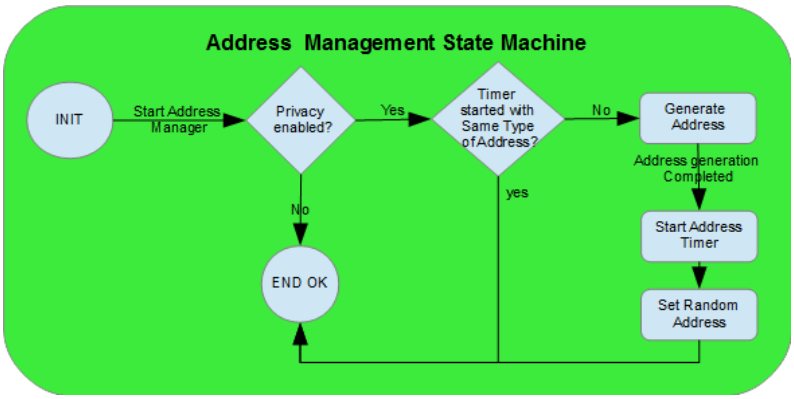

Figure 7-5: Air operation address management FW state machine

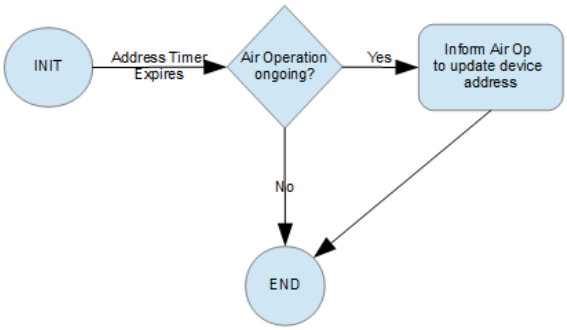

Figure 7-6: Privacy address management FW state machine

8 Security Manager Toolbox

The Bluetooth Low Energy Security Manager allows two devices to setup a secure relationship either by encrypting a link, by bonding (exchanging information about each other) or signature use over a plain link.

Please refer to the Bluetooth Core 4.0 specification [1] for the SM requirements and protocol methods.

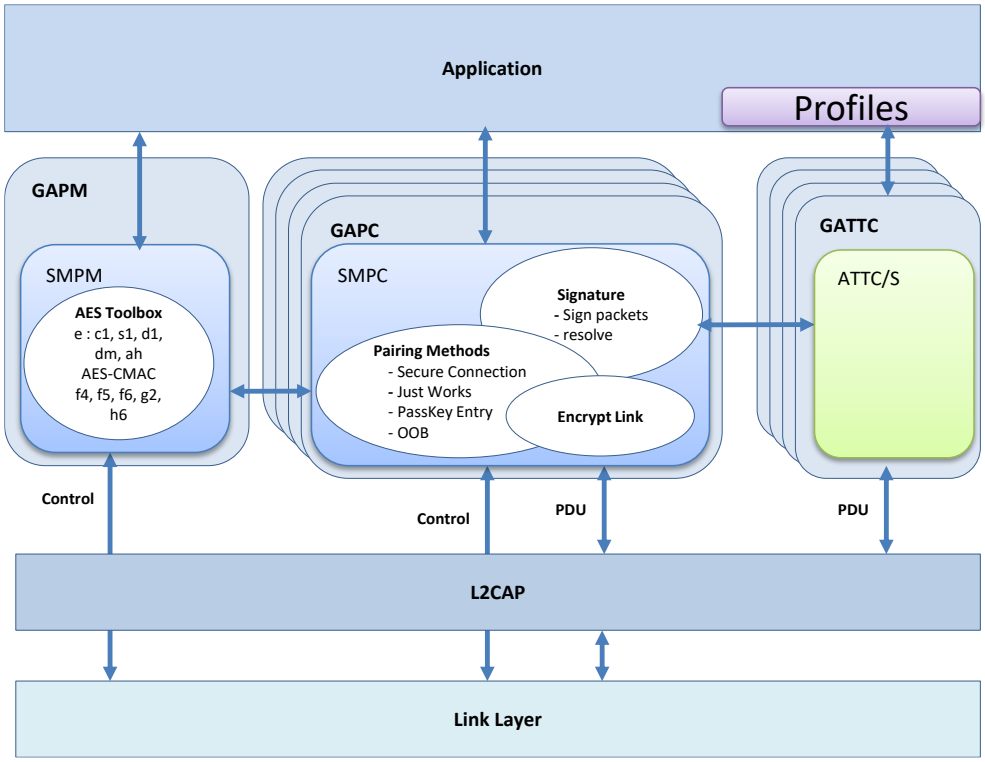

Figure 8-1: SMP Block Overview

A few key concepts must be presented for a clearer understanding of the SM:

- **Pairing**: this procedure allows two devices to agree upon features that will allow them to establish a certain level of security.
- **Bonding**: this procedure involves at least one device sending some sort of identification or security information to the other device, to be used in future connections. This may be an encryption key, signature key, identification resolution key. If both devices are bondable, the Transport Key Distribution Phase following pairing will occur. Otherwise, no bonding information will be exchanged and if any is sent, it is a violation of protocol. Pairing may occur without necessarily bonding, but the features exchanged during pairing are essential to the existence of a bonding stage. If not both devices are bondable, no information about the peer should be stored (not even BD address or other kind of non-security related information).
- **Unauthentication/Authentication**: Unauthentication is NOT lack of any security, but an intermediary level between no security and authenticated security level. The relationship between two devices is said to be (un)authenticated when the key(s) being used for their link encryption/signing/etc. have a security property that confirms (un)authentication. This security property is bestowed on a key during pairing, in function of the STK method generation used. For Either Passkey Entry and OOB Methods, all keys generated and exchanged afterwards have Authenticated (MITM) property (a pin key/ larger OOB key was used, this enforces security). If the Just Works method was used, all keys will have Unauthenticated (No MITM) property. There may also be the No security property, which applies when the link is plain.

- **LE Secure Connection**: This pairing method allows having a greater security level than the normal pairing method. It uses Private/Public keys (P-256 elliptic curve) security algorithm in order to prevent any man in the middle attack. This secure connection is a fully new pairing method that can be used for just work pairing, OOB or pin code entry. With this method, except just work pairing, the security level of the link is considered as "Secure Connection Authenticated" link.

The Security Manager (SM) Toolbox is in charge of BLE secure communication issues: encrypted links, identity or private addresses resolution and signed unencrypted messages. The functionalities of the SM are enforced by clearly specified pairing and key distribution methods, and the protocol that is to be respected for their correct implementation. An additional cryptographic toolbox of functions based on the AES-128 algorithm supports key generation, private address generation and resolution and message signing and signature resolution.

The architecture decided for the implementation of the Security Manager is visible in Core Spec 4.1 Vol. 3 Part C, Chap. 10 [1]. Since the different functionalities may be required simultaneously for several connections a device may have, those functionalities have been implemented in the toolbox called SMPC: the Security Manager Protocol Controller. SMPC toolbox is available only using GAPC API.

However, certain higher and lower layer modules have an unique instance, handled By GAPM task through its API, SMPM Toolbox – Security Manager Protocol Manager – which will monitor SMPC's requests and responses without overloading those modules.

The dialogue between SMPM and SMPC's through GAPM and GAPC's API is limited to a few basic requests and responses. The communication between SMPCs, Higher and lower layers is much richer and also allows a device to proceed with link encrypting procedures at different stages with the different peers it possesses.

8.1 Keys Definition

There are several important types of keys in BLE security.

Key Type	Description
Identity Root (IR)	✓ 128-bit key generated for LE device ✓ Only for devices that support encryption or use random addresses ✓ Device can have multiple IR keys, but will only use one per connection ✓ Used to generate IRK and DHK
Encryption Root (ER)	✓ 128-bit random generated ✓ Used to generate CSRK and LTK.
Identity Resolving Key (IRK)	✓ 128-bit key ✓ Used to resolve random addresses
Diversifier Hiding Key (DHK)	✓ 128-bit key ✓ Used to encrypt DIV during encryption connection setup
Connection Signature Resolving Key (CSRK)	✓ 128-bit key ✓ Used to sign and verify signatures on the receiving device
Long Term Key (LTK)	✓ 128-bit key, used partially depending on agreed key size ✓ Used to generate contributory session key for an encrypted connection
Diversifier (DIV)	✓ 128-bit stored value, used to calculate LTK ✓ A new DIV is generated each time a unique LTK is distributed. ✓ The DIV value is masked to the 2 octet EDIV distributed value.
Short Term Key (STK)	✓ Generated at the end of Phase 2 using TK

	✓ Used to encrypt link after Phase 2 (according to agreed key size)
Temporary Key (TK)	✓ Either 0, Pass Key or OOB depending on STK generation method ✓ Used to calculate STK

Table 8-1: BLE Keys

8.2 Cryptographic Algorithms Overview

This part gives a fast overview about the cryptographic algorithms used in the SMP layer.

The understanding of theses algorithms is required to better understand what is done in the SMP layer.

Notations:

- 0^s is the bit string that consists of s '0' bits.
- The concatenation on bit strings is denoted ||.
- The function $LSB_s(X)$ returns the s least significant bits of X.

8.2.1 Encryption Function e

The encryption function generates a 128-bit encrypted data from a 128-bit key and a 128-bit data using the AES-128-bit block implemented in the controller.

encryptedData = e(key, data)

Each time this function is supposed to be used in the SMP, a LLM_LE_ENC_CMD message is sent to the controller. The implementation of the stack implies that the answer LLM_LE_ENC_CMP_EVT is sent to SMPM.

8.2.2 Keys Generation

The keys used in SMP are distributory, which means that the keys for distribution are made by the devices that dispense them.

The following security functions are used to generate SMP keys:

- Encryption function ***e*** generates 128-bit *encryptedData* from a 128-bit key and 128-bit *plaintextData* using the AES-128 bit block cipher as defined in NIST Publication FIPS-197. This function is not implemented in the Host stack, as it makes use of the Controller feature allowing the encryption of a block of data by a specified key (HCI LE Encrypt command)

  encryptedData = e (*key*, *plaintextData*)

- The diversification function d1 generates diversified keys (IRK, CSRK, LTK, DHK) from base keys such as IR and ER and makes use of the encryption function e:

  d1 (k, d, r) = ***e*** *(k, d')* with d'= padding||r||d, r and d 16 bits long.

- Random addresses are generated by combining a random number and a hash of that random number. The random address hash function *ah* is used to generate the hash.

  ah (k, r) = ***e****(k, r') mod* 2^{24}, r'= padding || r with r 24 bits long

- DIV is masked during encryption session setup using the output of the DIV mask generation function *dm* to generate the distributed EDIV value.

  dm (k, r) = ***e****(k, r') mod* 2^{16}, r'= padding || r, with r 64 bits long

- During the pairing process confirm values are exchange. This confirm value generation function *c1* is used to generate the confirm values.

c1 (k, r, preq, pres, iat, rat, ia, ra) = ***e****(k,* ***e****(k, r XOR pres||preq||rat'||iat') XOR pad||ia||ra)*

with :

- *pres* = pairing result PDU
- *preq* = pairing request pdu
- *rat'* = responder address type padded to 8 bits
- *iat'* = initiator address type padded to 8 bits
- *ra* = responder address
- *ia* = initiator address

- The key generation function *s1* is used to generate the STK during the pairing process.

s1 (k, r1, r2) = ***e****(k, r')* , r1 = Srand, r2 = Mrand and r'= r1[127:64] || r2[127:64]

8.2.3 Resolvable Private Address Hash Part Generation

A resolvable private address is the concatenation of two 24-bit parts: the hash part and the random part. The hash allows an address to be resolved using an Identity Resolution Key (IRK) and the random part.

resolv_priv_addr = hash || prand

The function allowing to generate the hash part is called ah.

hash = ah(IRK, prand) = LSB_{24}*(e(IRK, prand'))*

with prand' = 0^{104} *|| prand*

8.2.4 Confirm Value Generation

The confirm value is exchanged during a pairing procedure.

The function used to generate a confirm value is called c1. This function takes values as parameters:

- Temporary Key (TK), 128 bits
- A random number (R), 128 bits
- The pairing request command (PREQ), 56 bits
- The pairing response command (PRES), 56 bits
- The initiating device address type (IAT), 1 bit
- The initiating device address (IA), 48 bits
- The responding device address type (RAT), 1 bit
- The responding device address (RA), 48 bits

The following steps have to be followed within the c1 function:

- *Let* p_1 *= PRES || PREQ ||* 0^7 *|| RAT ||* 0^7 *|| IAT*
- *Let* $p_2 = 0^{32}$ *|| IA || RA*
- *Let* $e_1 = e(TK, R \text{ XOR } p_1)$
- *confirm* $= e(TK, e_1 \text{ XOR } p_2)$

8.2.5 Short Term Key (STK) Generation

The STK is used during the pairing procedure to encrypt the link before the device keys are exchanged. It is generated through the s1 function.

The following steps have to be followed within the s1 function:

- *Let Srand' = LSB_{64}(Srand)*
- *Let Mrand' = LSB_{64}(Mrand)*
- *Let r = Srand' || Mrand'*
- *STK = e(TK, r)*

8.2.6 AES-CMAC Algorithm

RFC-4493 1 defines the Cipher-based Message Authentication Code (CMAC) that uses AES-128 as the block cipher function, also known as AES-CMAC.

The inputs to AES-CMAC are:

- M is the variable length data to be authenticated
- K is the 128-bit key

The AES-CMAC requires generation of two subkeys which are deduced from input key K.

Generation of two subkeys is made using the following algorithm:

- *Let L = e(CSRK, 0^{128})*
- *If MSB(L) == 0, K_1 = (L << 1) else K_1 = (L << 1) XOR R_{128}, with R_{128} = 0^{120}10000111*
- *If MSB(K_1) == 0, K_2 = K_1 << 1 else K_2 = (K_1 << 1) XOR R_{128}*

AES-CMAC is performed using the algorithm describes below:

- *Produce K_1, K_2 = subkey_gen(K) (see 0)*
- *Let n = ceil(M_{len}/128), M_{len} = len(M), M = Data PDU*
- *Let M = M_1 || M_2 || … || M_n* where M_1, …, M_{n-1} are blocks of size 128*
- *If len(M_n*) == 128, M_n = K_1 XOR M_n* else M_n = K_2 XOR (M_n* || **1**0^j), with j = 128 – len(M_n*) - 1*
- *Let C_0 = 0^{128}*
- *For i = 1 to n, let C_i = AES_128(K, C_{i-1} XOR M_i)*

8.2.7 Data Signing (MAC Generation)

The Data Signing procedure is used to authenticate a data PDU sent over a non-encrypted link.

The algorithm used for data signing in presented in [2].

The inputs to MAC are:

- *M is the variable length data to be authenticated*
- *CSRK is the 128-bit key*

The MAC generation uses AES-CMAC algorithm like described below:

- *MAC = MSB_{64}(AES-CMAC $_{CSRK}$(M))*

8.3 LE Secure Connections Confirm Value Generation Function f4

During the LE Secure Connections pairing process, confirm values are exchanged. These confirm values are computed using the confirm value generation function f4 (see [1] Vol 3 Part H Chapter 2.2.6).

This confirm value generation function makes use of the MAC function AES-CMAC X , with a 128-bit key X.

The inputs to function f4 are:

- *U is 256 bits*
- *V is 256 bits*
- *X is 128 bits*
- *Z is 8 bits*

Algorithm:

- *f4 (U, V, X, Z) = AES-CMAC $_{X}$ (U || V || Z)*

Numeric Comparison / Just Works	Out-Of-Band	Passkey Entry
Ca = f4 (PKax, PKbx, Na, 0)	Ca = f4 (PKax, PKax, Rara, 0)	Cai = f4 (PKax, PKbx, Nai, rai)
Cb = f4 (PKbx, PKax, Nb, 0)	Cb = f4 (PKbx, PKbx, Rbrb, 0)	Cbi = f4 (PKbx, PKax, Nbi, rbi)

Table 8-2 Inputs to f4 for the different protocols

8.4 LE Secure Connections Key Generation Function f5

The LE Secure Connections key generation function f5 is used to generate derived keying material in order to create the LTK and keys for the commitment function f6 during the LE Secure Connections pairing process (see [1] Vol 3 Part H Chapter 2.2.7).

The inputs to function f5 are:

- *W is 256 bits*
- *N1 is 128 bits*
- *N2 is 128 bits*
- *A1 is 56 bits*
- *A2 is 56 bits*

The key (T) is computed as follows:

- *SALT = 0x6C88 8391 AAF5 A538 6037 0BDB 5A60 83BE*
- *T = AES-CMAC$_{SALT}$(W)*

The string “btle” is mapped into keyID using extended ASCII as follows:

- *keyID = 0x62746c65*

The output of the key generation function f5 is as follows:

- *f5(W, N1, N2, A1, A2) = AES-CMAC$_T$(0x00||keyID||N1||N2||A1||A2||0x0100)*
  || AES-CMAC$_T$(0x01||keyID||N1||N2||A1||A2||0x0100)

The LTK and MacKey are calculated as:

- *MacKey || LTK = f5(DHKey, N_master, N_slave, BD_ADDR_master, BD_ADDR_slave)*

8.5 LE Secure Connections Check Value Generation Function f6

The LE Secure Connections check value generation function f6 is used to generate check values during authentication stage 2 of the LE Secure Connections pairing process (see [1] Vol 3 Part H Chapter 2.2.8).

The inputs to function f6 are:

- *W is 128 bits*
- *N1 is 128 bits*
- *N2 is 128 bits*
- *R is 128 bits*
- *IOcap is 24 bits*
- *A1 is 56 bits*
- *A2 is 56 bits*

The output of the check value generation function f6 is as follows:

- *f6(W, N1, N2, R, IOcap, A1, A2) = AES-CMAC$_W$(N1||N2||R||IOcap||A1||A2)*

The inputs to f6 are different depending on different association models:

Numeric Comparison / Just Works	Out-Of-Band	Passkey Entry
Ea = f6 (MacKey, Na, Nb, 0,IOcapA, A, B)	Ea = f6 (MacKey, Na, Nb, rb, IOcapA, A, B)	Ea = f6 (MacKey, Na20, Nb20, rb, IOcapA, A, B)
Eb = f6 (MacKey, Nb, Na, 0, IOcapB, B, A)	Eb = f6 (MacKey, Nb, Na, ra, IOcapB, B, A)	Eb = f6 (MacKey, Nb20, Na20, ra, IOcapB, B, A)

Table 8-3 Inputs to f6 for the different protocols

In Passkey Entry, ra and rb are 6-digit passkey values expressed as a 128-bit integer.

8.6 LE Secure Connections Numeric Comparison Value Generation Function g2

The LE Secure Connections numeric comparison value generation function g2 is used to generate the numeric comparison values during authentication stage 1 of the LE Secure Connections pairing process (see [1] Vol 3 Part H Chapter 2.2.9).

The inputs to function g2 are:

- *U is 256 bits*
- *V is 256 bits*
- *X is 128 bits*
- *Y is 128 bits*

The output of the numeric comparison value generation function g2 is as follows:

- *g2 (U, V, X, Y) = AES-CMAC$_X$(U||V||Y) mod 2^{32}*

8.7 Link Key Conversion Function h6

The function h6 is used to convert keys of a given size from one key type to another key type with equivalent strength (see [1] Vol 3 Part H Chapter 2.2.10).

The inputs to function h6 are:

- *W is 128 bits*
- *keyID is 32 bits*

The output of h6 is as follows:

- *h6 (W, keyID) = AES-CMAC$_W$(keyID)*

8.8 Identity Root Generation

The Identity Root (IR) can be created in two ways. It can be assigned by a value or generated in random. If by arbitrary creation, it will follow the requirements of random generation defined in Volume 2, Part H Section 2 of Bluetooth Core Specification.

8.8.1 Identity Resolving Key Generation

The Identity Resolving Key (IRK) is used for random address construction and resolution. It is created through the diversification function d1, using the IR as parameter k and 0x0001 as parameter d. In case a hierarchy method is not used, IRK may be directly assigned as a random 16 octet value to the device (per connection).

8.8.2 Diversifier Hiding Key Generation

The Diversifier Hiding Key (DHK) is used to mask DIV during the encrypted session setup. It is created through the diversification function d1, using the IR as parameter k and 0x0002 as parameter d. If the hierarchy method is not used, it can also be randomly generated.

8.8.3 Connection Signature Resolving Key Generation

The Connection Signature Resolving Key (CSRK) is used to sign data and resolve signature of received messages. It can be assigned or randomly generated. If by arbitrary creation, it will follow the requirements of random generation defined in Volume 2, Part H Section 2 of Bluetooth Core Specification.

8.8.4 Long Term Key and Diversifier Generation

Devices supporting encrypted links in the slave role are capable of generating unique LTK and DIV values.

The DIV is used by the slave device to regenerate a previously shared LTK in order to start an encrypted connection with a previously paired master device.

Any method of generation of LTK can be used as it is not visible outside the slave device. New values of LTK and DIV are generated each time they are distributed.

8.8.5 Encrypted Session Setup

Establishing an encrypted link requires that both devices use the same Key, which has either been generated on both devices using the same base parameters (reference to STK) or previously distributed. Both devices always use the Slave distributed LTK if the link is to be encrypted using LTK.

The host of the master provides the Link Layer with the Long Term Key to use when setting up the encrypted session, together with the EDIV and RAND number that correspond to it. The EDIV and RAND are two 'identifiers' for the LTK and they allow retrieval of the same key on both devices without actually exchanging it. During the encryption session setup the master device sends the *EDIV* and the random number to the slave device. The host of the slave receives the *EDIV* and *Rand* values and provides the corresponding Long Term Key to the slave's Link Layer to use when setting up the encrypted link.

The encrypted session can be setup either by using STK or LTK. The procedure is the same, the only difference being that when using STK, EDIV=RAND=0.

8.8.6 Signing Algorithm

An LE device can send signed data without having to establish an encrypted session with a peer device. Data is signed using *CSRK*.

The signing algorithm is used in two situations:

- ✓ Signing own data with own CSRK in view of transmission to peer which is supposed to have received the CSRK during phase 3, and would thus interpret the received message.
- ✓ Verification of received signed messages, using CSRK received from peer during previous Phase 3. The same algorithm is used to generate the signature of the received message and check it against the received signature.

8.8.7 Slave Initiated Security

There are three manners in which the master handles the security request from slave.

- ✓ No LTK is available for this connection, or the existing security information does not have the requested security properties => pairing must be initiated.
- ✓ An LTK is available for this connection, with security properties matching the request => start encrypting link directly without pairing.
- ✓ Send slave Pairing Failed PDU that Master does not support pairing at that moment.

8.9 Procedure Details

This part presents the messages that are exchanged between the layers of the RW-BLE stack during the different procedure that are supported by the SMP. The SMP API messages are described in the next part.

8.9.1 Random Address Generation

A device may use a random address. This random address may be of either of the following types:

- Static address
- Private address

A private address may be either of the following types:

- Non-resolvable private address

- Resolvable private address

The three figures below give the structure of each kind of private address.

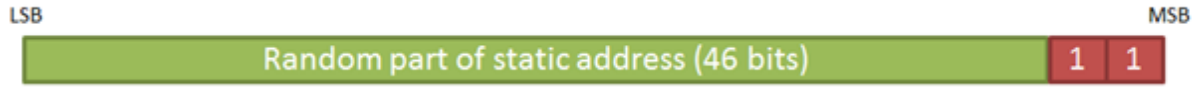

Figure 8-2: Static random address structure

Figure 8-3: Private non-resolvable random address structure

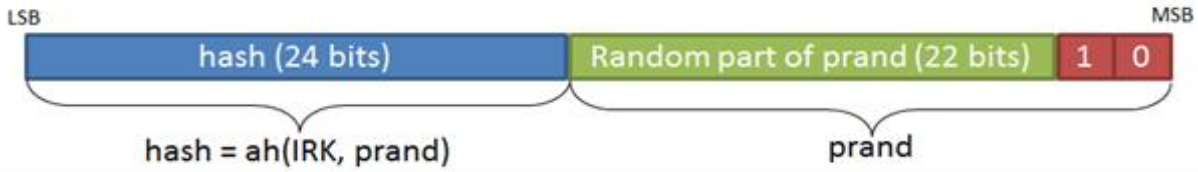

Figure 8-4: Private resolvable random address structure

The random address generation procedure will be the same whatever the kind of random address which is requested. However in the case of a resolvable private address, the IRK used to generate the address shall be kept by the higher layers so that it can be distributed to a peer device.

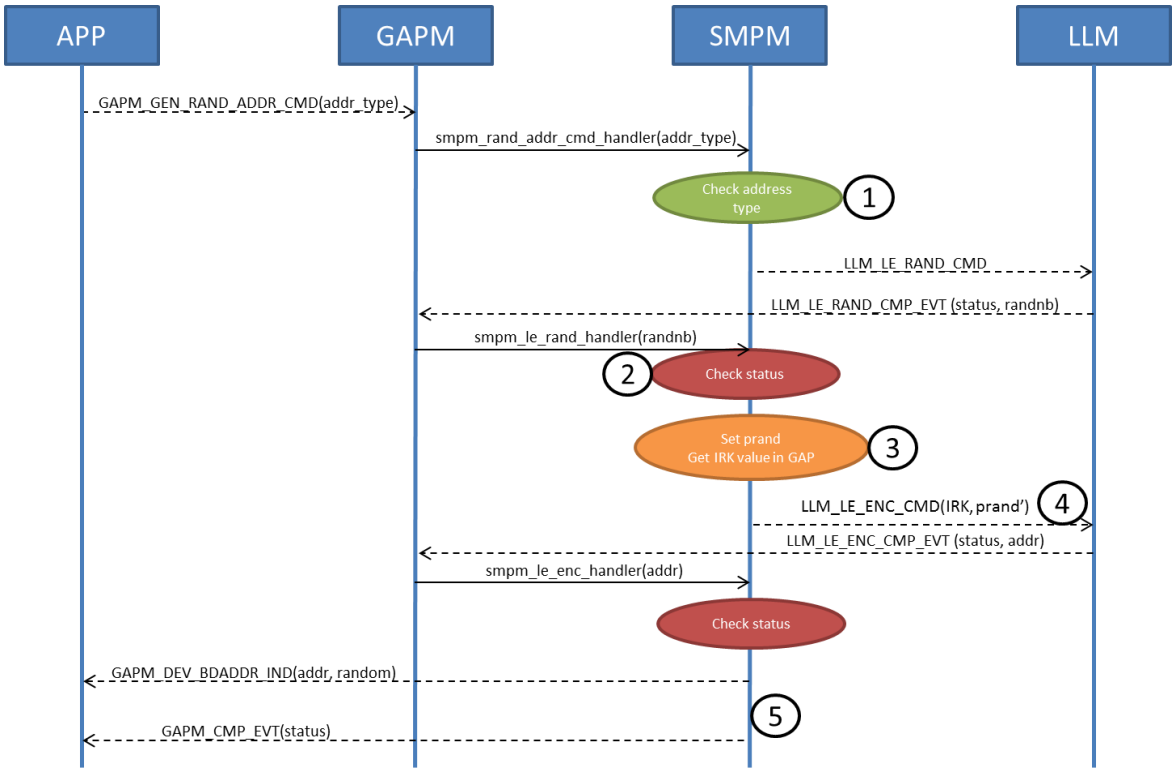

Figure 8-5: Random Address Generation Procedure

- If the address type is not valid, a GAPM_CMP_EVT message with a GAP_ERR_INVALID_PARAM status error is sent.
- If an error status is returned by the controller, a GAPM_CMP_EVT message with a GAP_ERR_LL_ERROR status error is sent.
- prand = LSB_{22}(randnb) || LSB_{2}(addr_type)
- prand' = 0^{104} || prand
- If an error status is returned by the controller, a GAPM_CMP_EVT message with a GAP_ERR_LL_ERROR status error is sent else the status will be GAP_ERR_NO_ERROR.

8.9.2 Address Resolution

The address resolution procedure is used to identify a device which would use a resolvable private random address. The structure of this kind of address is defined in 8.9.1.

The GAP provides several IRK for a same address. The hash part of this address is regenerated using the IRK and the prand part of the address. If the generated hash part is the same as the hash part of the provided address, the address is considered as resolved, else another IRK shall be sent.

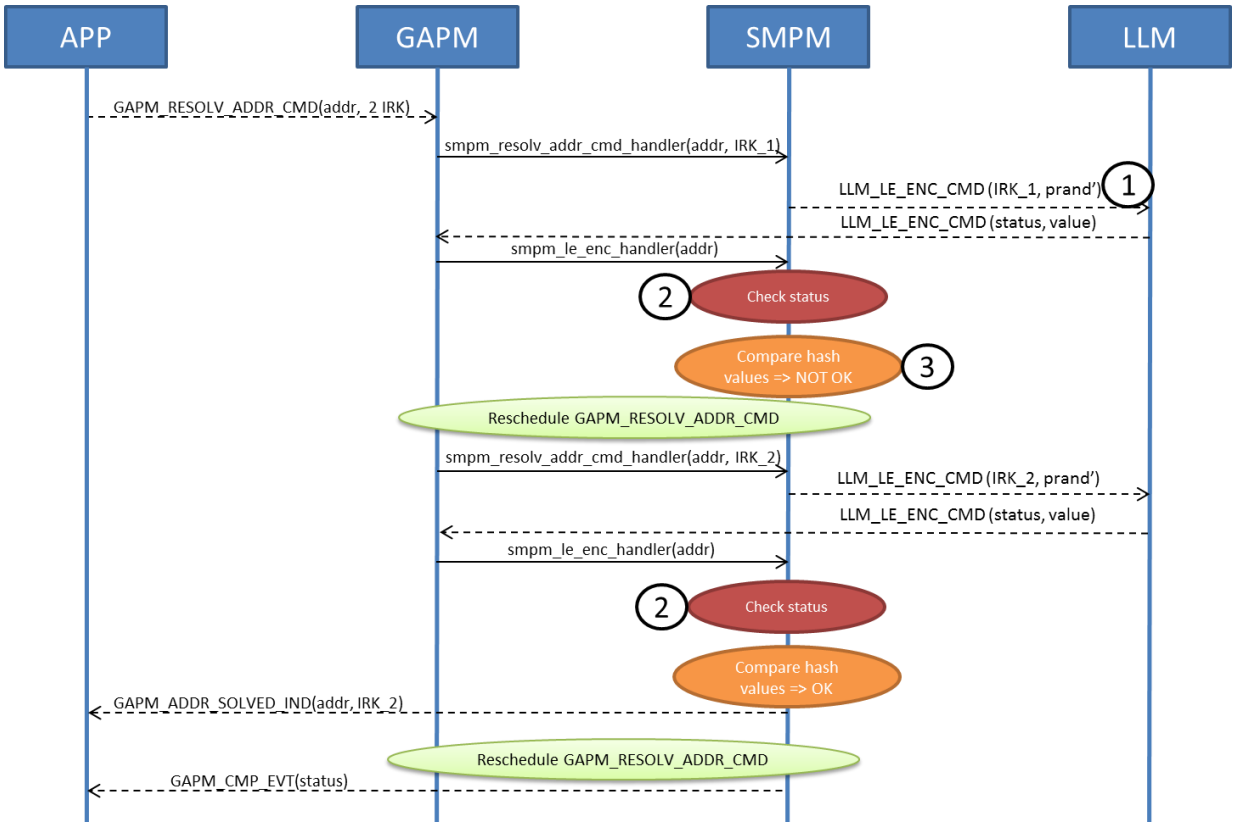

Figure 8-6: Address Resolution Procedure

- prand' = 0^{104} || prand = 0^{104} || addr[0:23]
- If an error status is returned by the controller, a GAPM_CMP_EVT message with a GAP_ERR_LL_ERROR status error is sent.
- hash = value[0:23]

8.9.3 Encryption Toolbox Access

The encryption toolbox access provides a way a host layer to use the hardware encryption block. This block can be access using the LLM API.

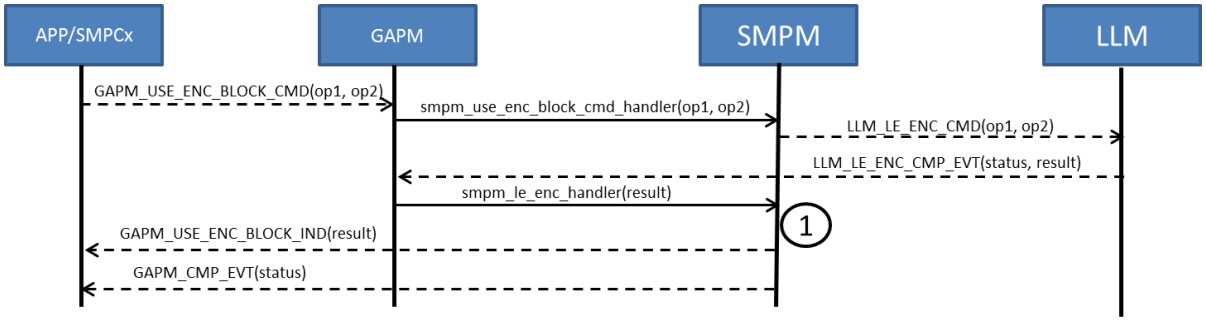

Figure 8-7: Encryption Toolbox Access

- If an error status is received from the controller, the GAPM_CMP_EVT message with a GAP_ERR_LL_ERROR status is directly sent to the requested layer.

8.9.4 Pairing

The pairing procedure between two devices can be divided into several phases:

- Phase 1 – Pairing Feature Exchange: It is used to exchange IO capabilities, OOB authentication data, authentication requirements and which keys to distribute.
- Legacy Phase 2 – Authentication and Encryption: Information exchanged during the phase 1 are used to determine which method will be used to encrypt the link (Just Works, Passkey Entry, Out Of Band).
- LE Secure Connections Phase 2: – Authentication and Encryption: Information exchanged during the phase 1 are used to determine which method will be used to encrypt the link (Just Works, Numeric Comparison, Passkey Entry, Out Of Band). The outcome of this pairing is Long Term Key (LTK) Generation
- Phase 3 – Transport Keys Distribution: This phase is optional and depends on the key distribution features shared during phase 1.

8.9.4.1 Phase 1: Pairing Feature Exchange (Initiated by Master)

The pairing is always iniated by the master device by sending a pairing request message.

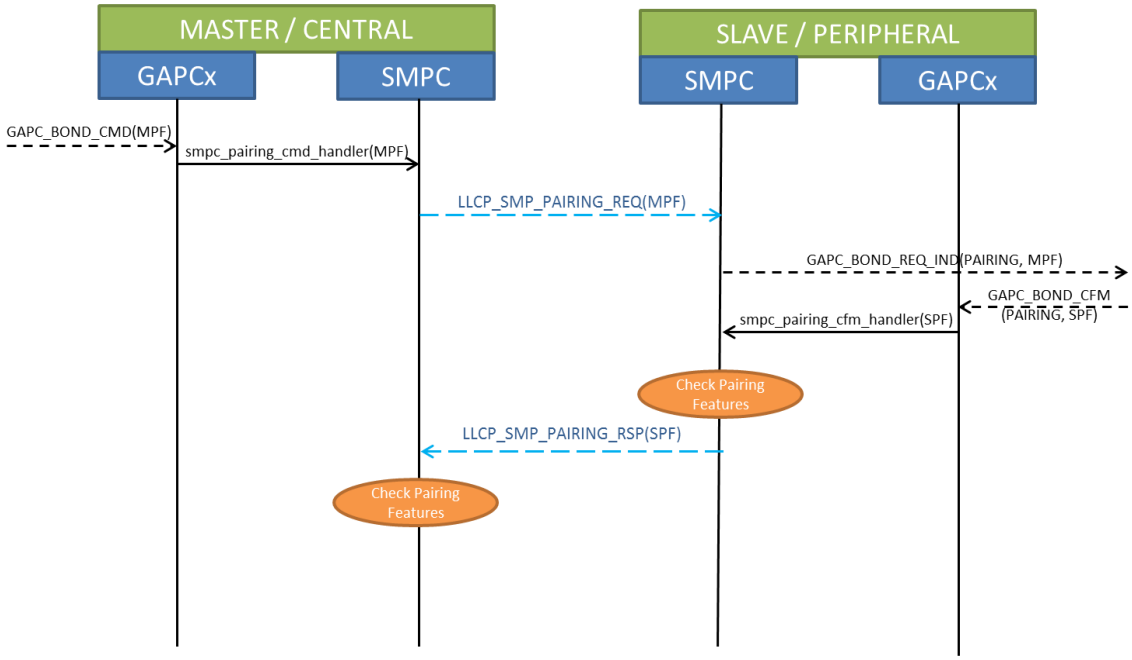

Figure 8-8: Pairing Phase 1: Pairing Features Exchange (Initiated by Master)

If the slave device doesn't support pairing, it responds using the Pairing Failed message with the error code "Pairing Not Supported" upon reception of a Pairing Request message.

If a device receives a command with invalid parameters, it responds with a Pairing Failed command with the error code "Invalid Parameters".

The minimum and maximum encryption key length parameters shall be between 7 bytes and 16 bytes in 1 byte step. The smaller value of the initiating and the responding devices maximum encryption key length will be used as the encryption key size. If the resultant encryption key size is smaller than the minimum key size parameter, the device responds with Pairing Failed command with the error code "Encryption Key Size".

8.9.4.2 Phase 1: Pairing Feature Exchange (Initiated by Slave)

A slave device may require that the master initiates a pairing produre by sending a security request.

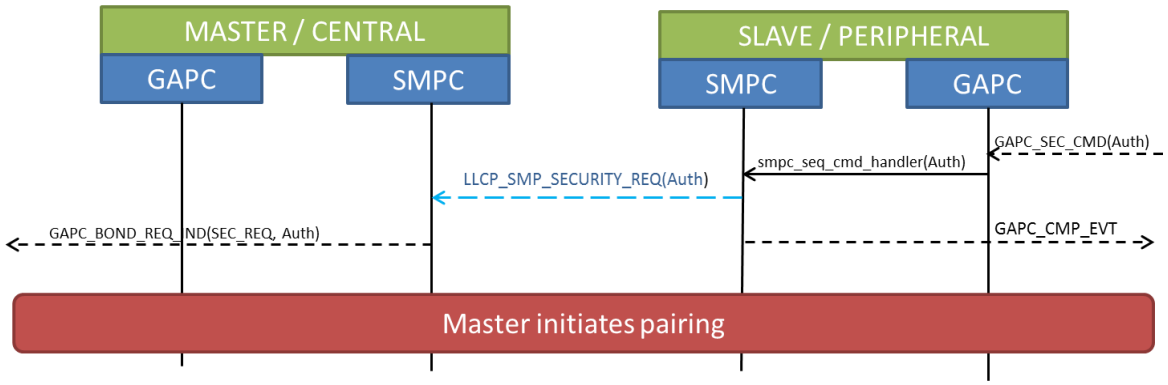

Figure 8-9: Pairing Phase 1: Pairing Features Exchange (Initiated by Slave)

8.9.4.3 Legacy Phase 2: Authentication and Encryption

The information exchanged in Phase 1 is used to select which STK generation method is used in Phase 2.

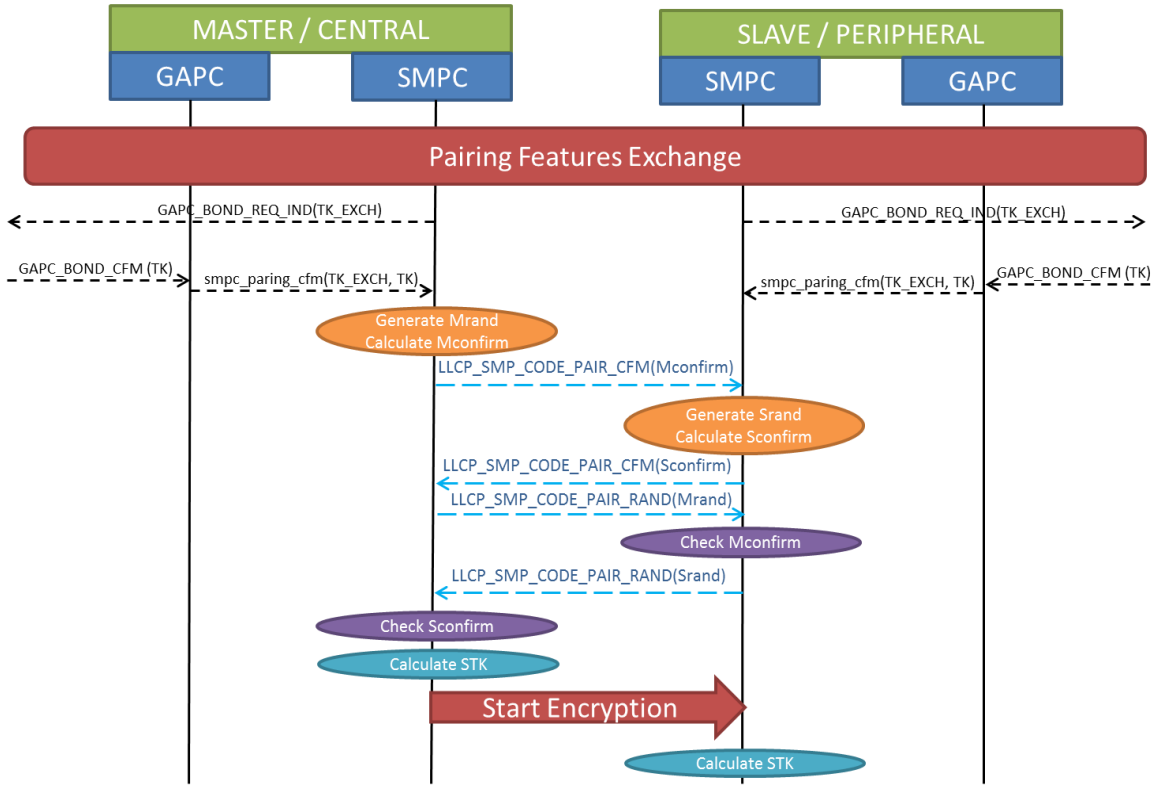

Figure 8-10: Phase 2: Authentication and Encryption

If the just works method is used, no TK will be required (0 is used) from the application.

If a generated Sconfirm or Mconfirm value doesn't match with the received confirm value from the peer device, the device aborts the the pairing procedure by sending a Pairing Failed message with a "Confirm Value Failed" error code.

8.9.4.4 LE Secure Connection Phase 2: Authentication and Encryption

8.9.4.4.1 Authentication Stage 1: Just Work Method

If not possible to enter passkey or do a numeric comparison, this method applies:

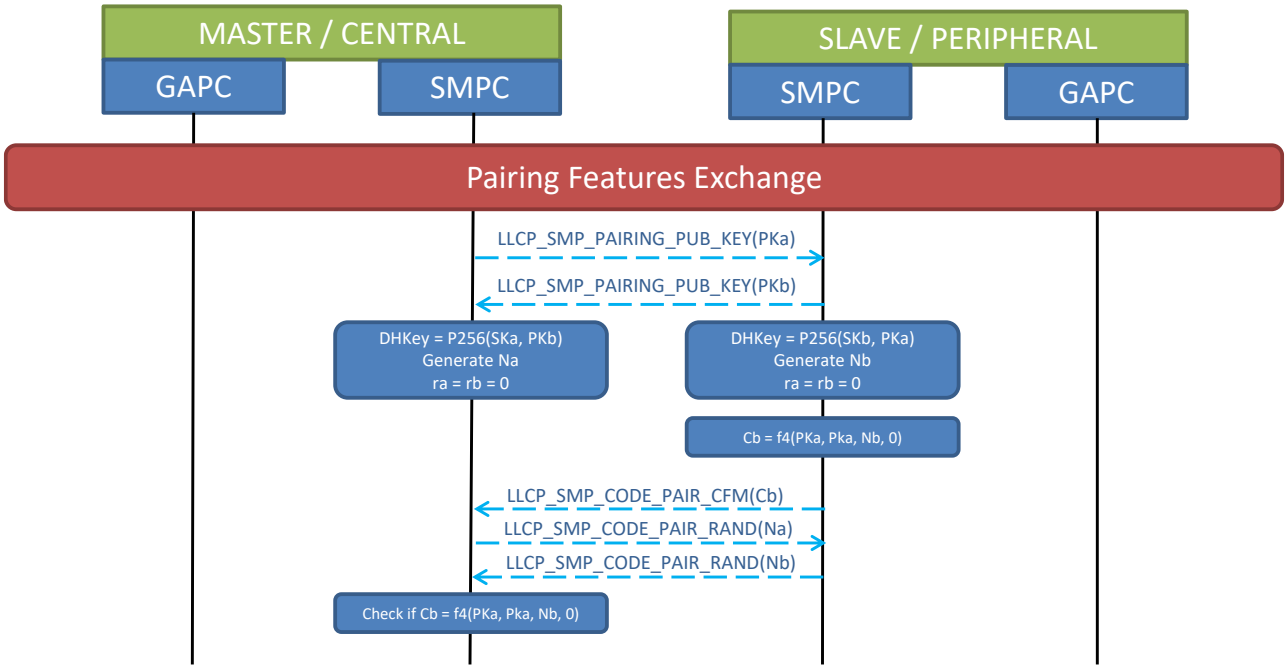

Figure 8-11: Phase 2: LE Secure Connection Just Work Pairing

At the end of the pairing, link is considered Unauthenticated.

8.9.4.4.2 Authentication Stage 1: Numeric Comparison Method

If both devices have display capability, Numeric comparison should be chosen

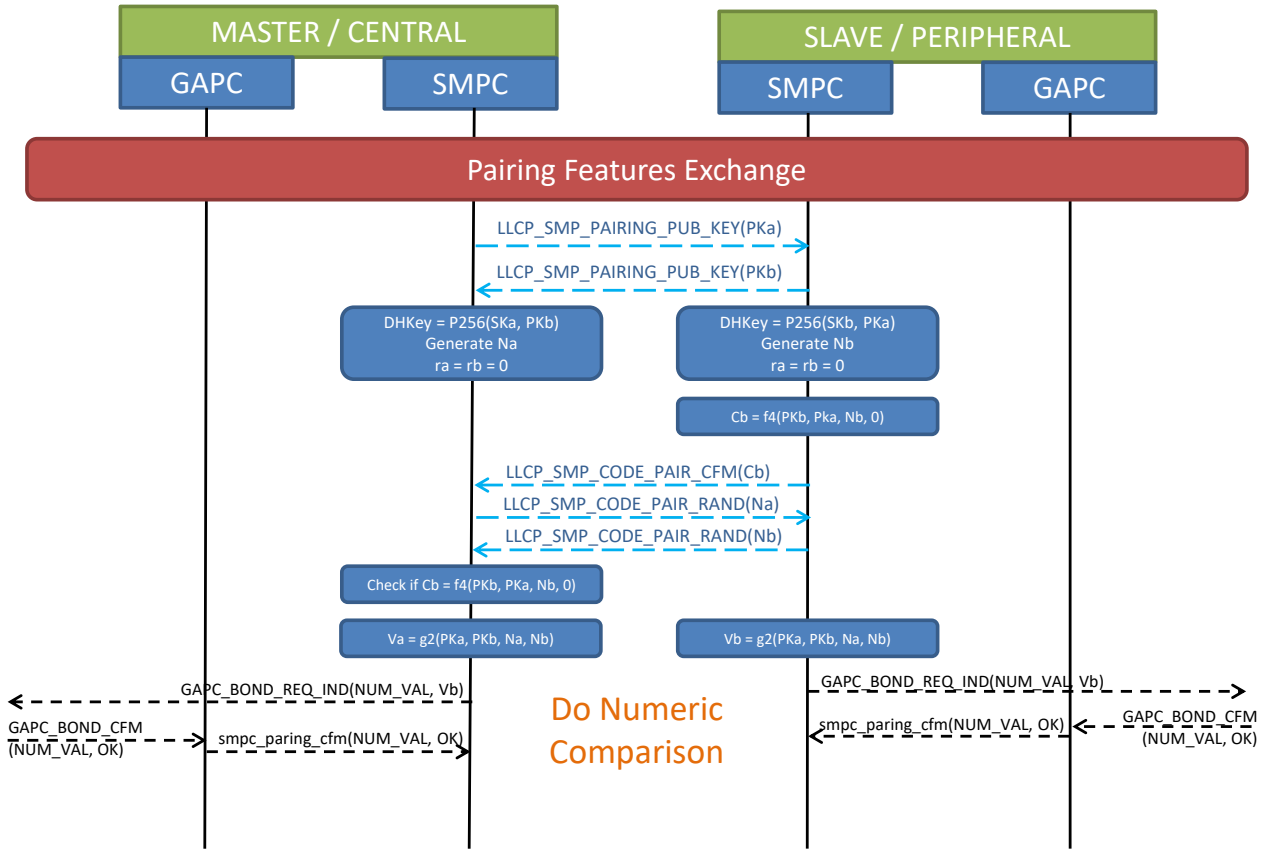

Figure 8-12: Phase 2: LE Secure Connection Numeric Comparison Pairing

At the end of the pairing, link is considered Secure Connection Authenticated.

8.9.4.4.3 Authentication Stage 1: Passkey Entry Method

If both devices pin code entry is possible, passkey entry is chosen:

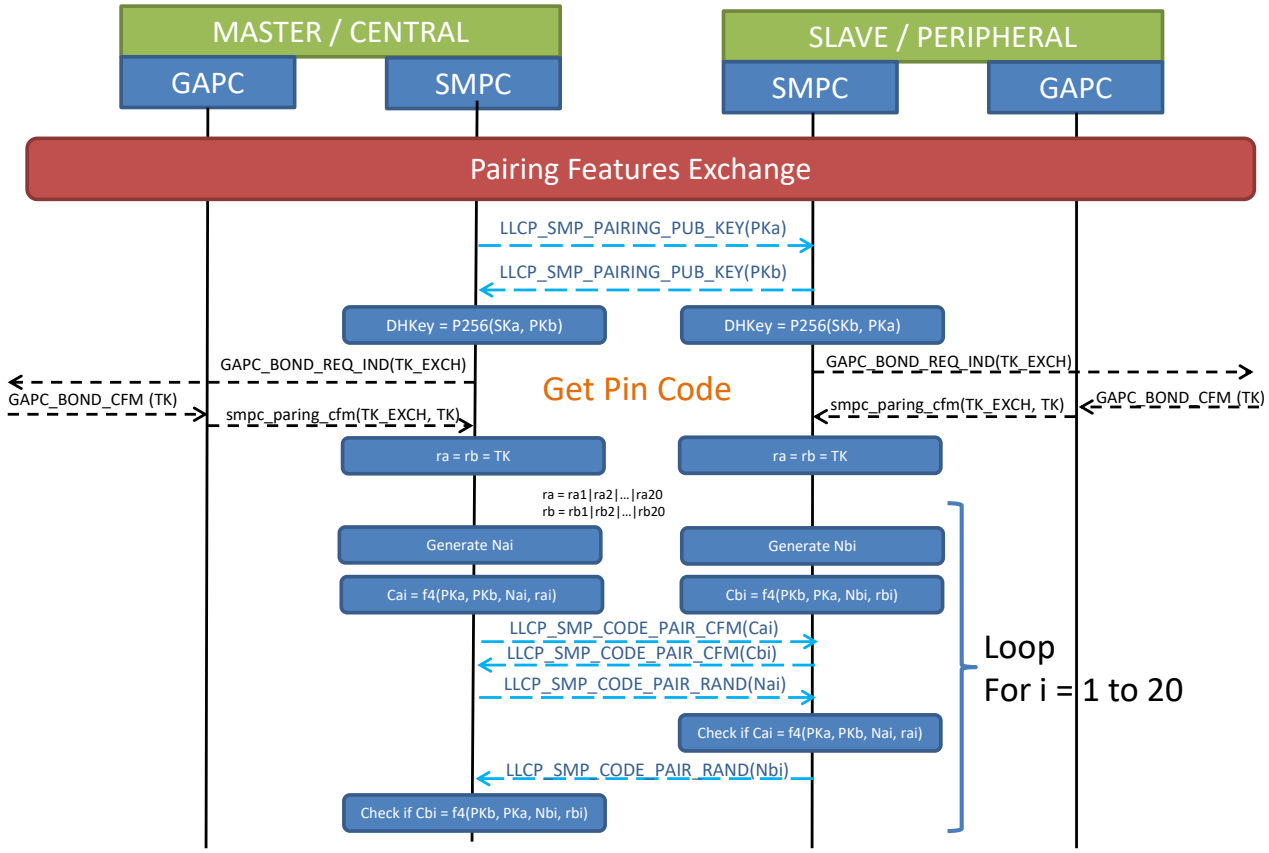

Figure 8-13: Phase 2: LE Secure Connection Passkey Entry pairing

During Passkey entry, *LLCP_SMP_PASS_KEY_ENTRY* message can be sent to peer device using *GAPC_BOND_CFM(PASSKEY_ENTRY)* message in order to inform peer device that user is entering password.

At the end of the pairing, link is considered Secure Connection Authenticated.

8.9.4.4.4 Authentication Stage 1: Out of Band Method

If OOB Data can be sent by one or both devices, Out Of Band pairing method is chosen:

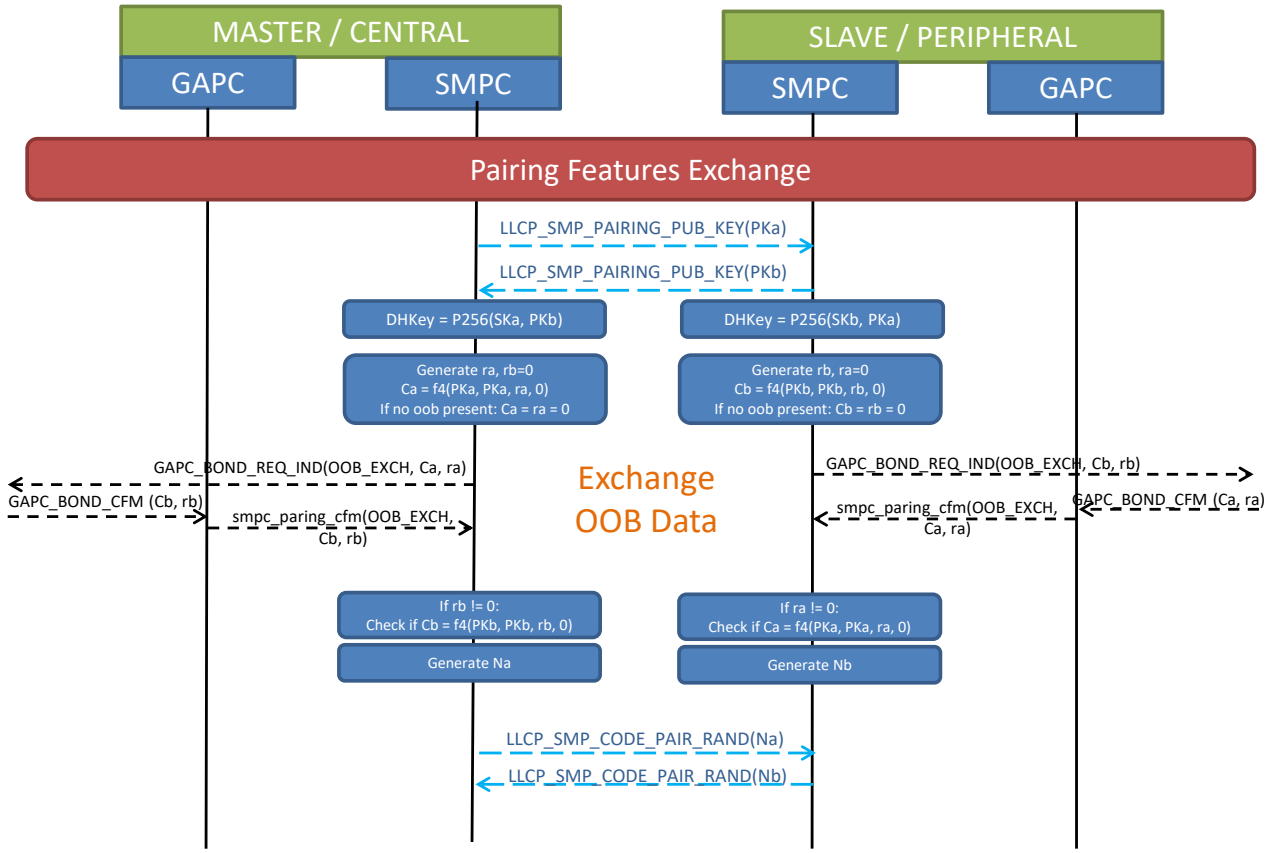

Figure 8-14: Phase 2: LE Secure Connection Out Of Band pairing

At the end of the pairing, link is considered Secure Connection Authenticated.

8.9.4.4.5 Authentication Stage 2: Generation of LTK

After the LE Secure Connection Authentication Stage2, LTK is generated according to pairing information. Then link is encrypted.

If encryption succeeds and BOND bit present in pairing feature exchange, the generated LTK is provided to upper application.

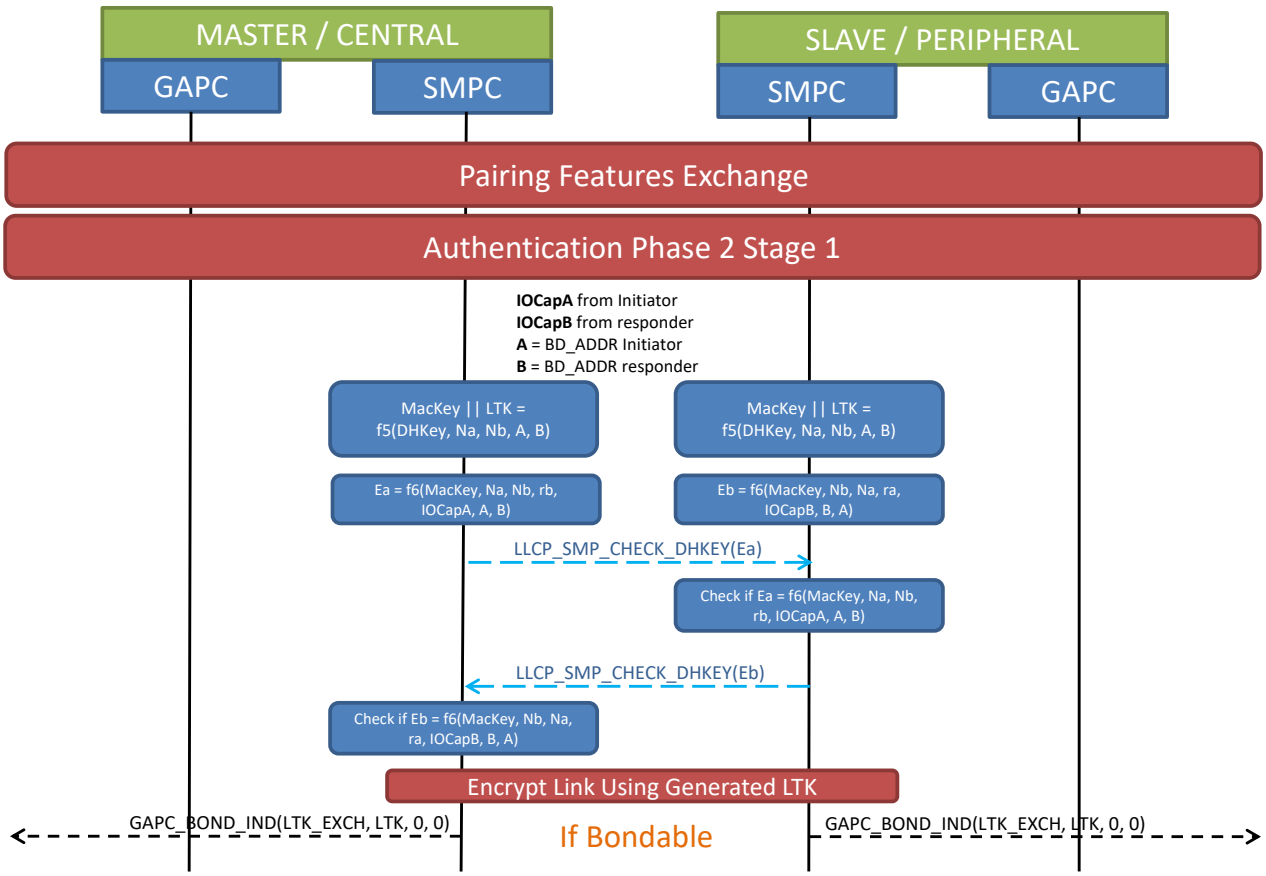

Figure 8-15: Phase 2: LE Secure Connection LTK Generation

8.9.4.5 Phase 3: Transport Keys Distribution

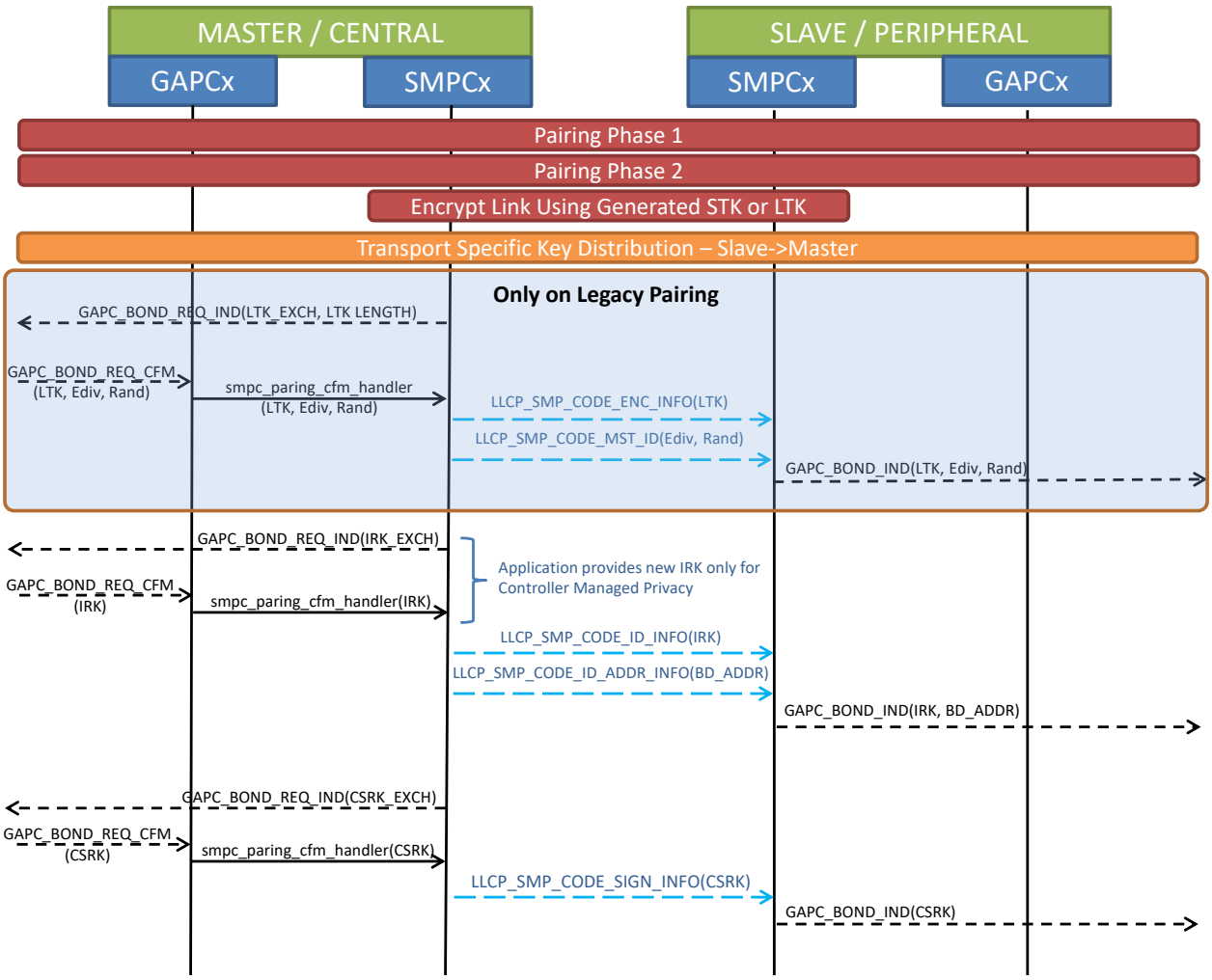

Figure 8-16: Phase 3: Transport Keys Distribution

When Privacy is managed by host (privacy 1.1), IRK value is already set in the GAP environment. But for a controller managed privacy (privacy 1.2), IRK should be unique for each bonded device, and so the new IRK should be generated and retrieved from the application.

The LTK and the CSRK needs to be retrieved from the application.

On legacy paring, application is responsible for generating the transport keys (CSRK, IRK, LTK, Ediv, Rand) by any means. The GAPM_USE_ENC_BLOCK_CMD message can be used through the GAP API.

On secure Connection pairing, application is responsible for generating only CSRK, IRK, the LTK is generated by the pairing algorithm

8.9.4.6 End of Pairing Procedure

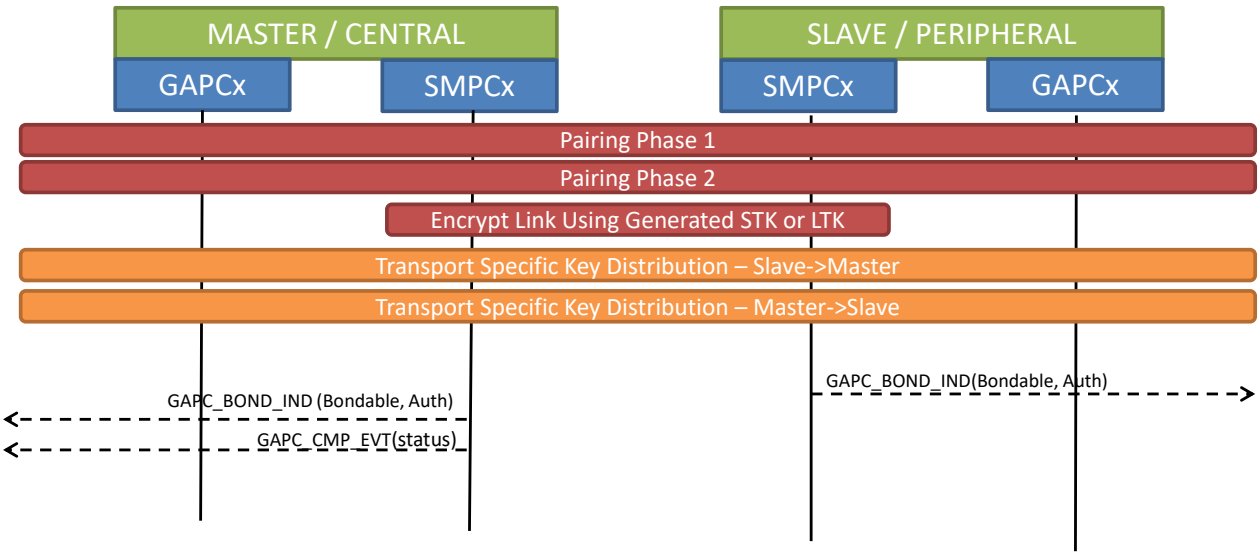

Figure 8-17: End of Pairing Procedure

The pairing procedure is considered as over in the following case:

- A Pairing Failed message has been received or generated.
- Phase 2 is over and no keys need to be distributed.
- All required keys have been distributed during Phase 3.

8.9.5 Encryption

The master device must have the security information (LTK, EDIV, and Rand) distributed by the slave device to setup an encrypted session. An encrypted session is always initiated by the master.

8.9.5.1 Case 1: Both devices have LTK

If a master already knows the encryption keys of the slave device it is connected with, it can initiate the creation of an encrypted link.

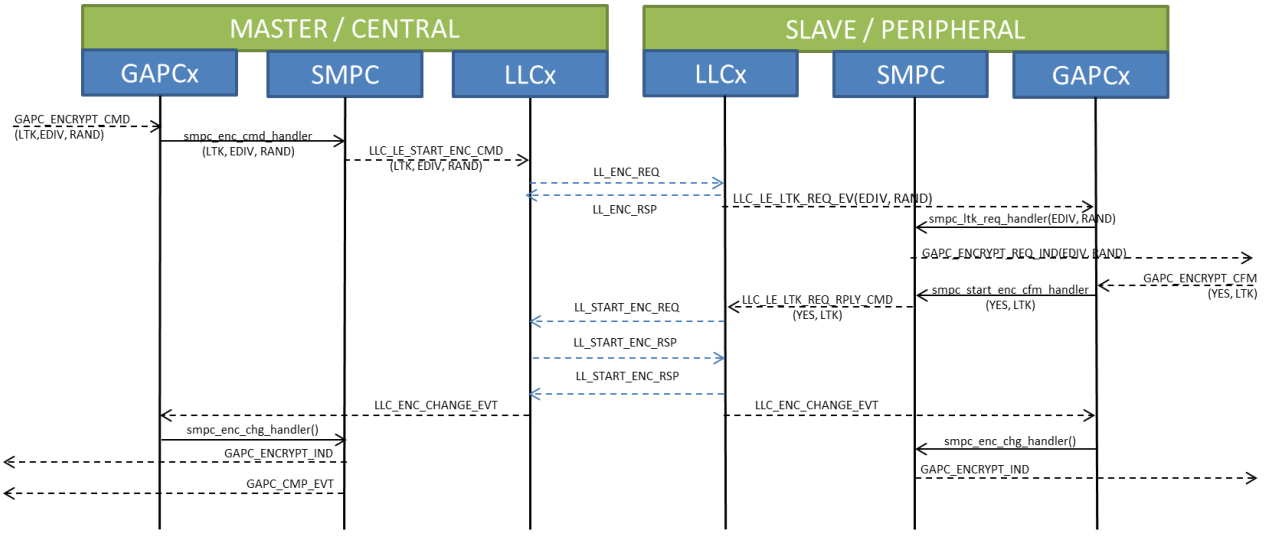

Figure 8-18: Start Encryption Procedure (Both devices have keys)

The slave may require establishment of an encrypted session by sending a security request. Upon reception of this request, the master device will check if it can retrieve the LTK distributed by the device. If a key is found, the master will start the encryption procedure else it will start a pairing procedure.

8.9.5.2 Case 2: Slave forgot the LTK

If the slave forgot the LTK distributed by the master device during a previous bonding procedure, it will reject the encryption request with a “PIN KEY missing” error.

Upon reception of this error, the master can initiate a new pairing procedure with the slave device.

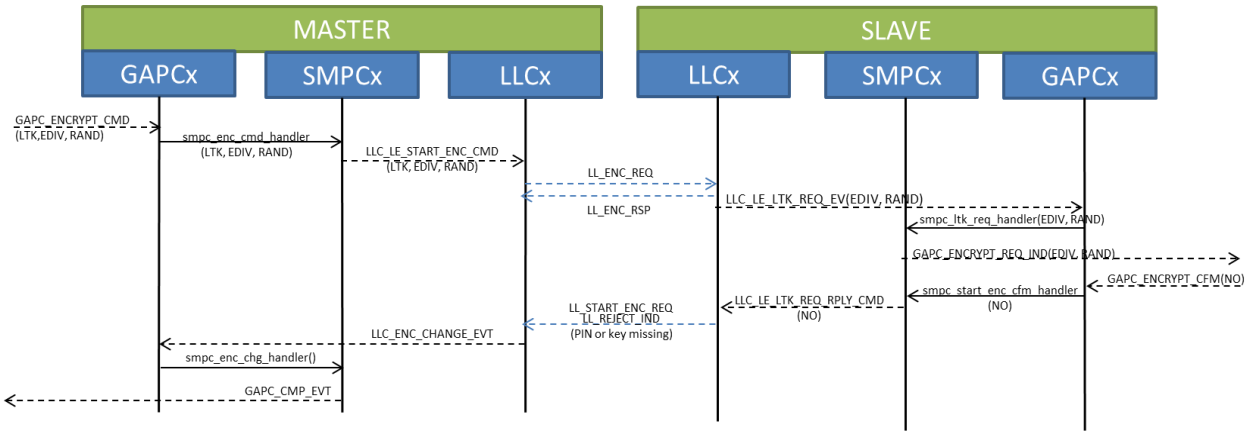

Figure 8-19: Start Encryption Procedure (Slave forgot keys)

8.9.5.3 Case 3: Slave doesn’t support encryption

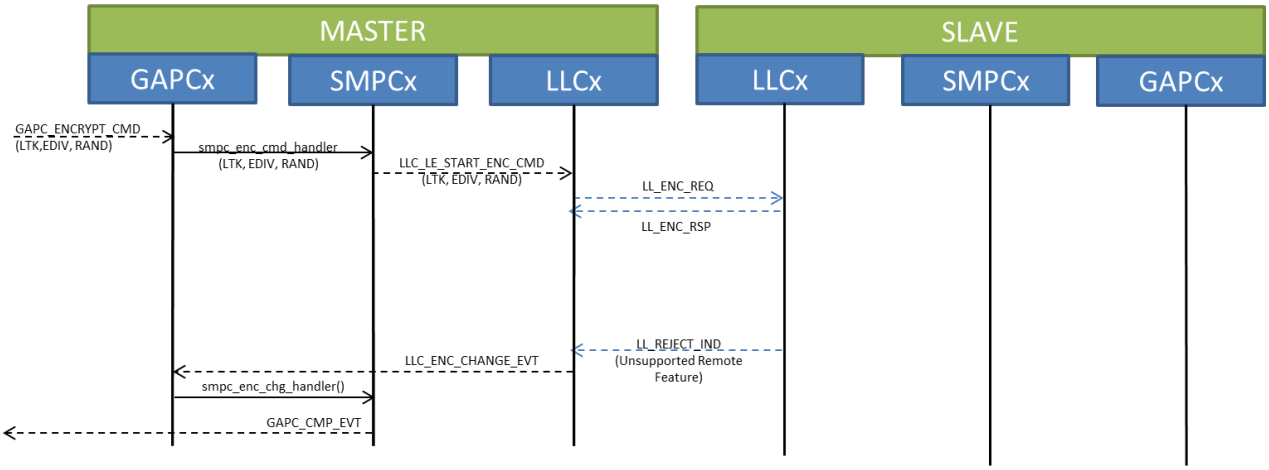

Figure 8-20: Start Encryption Procedure (Slave doesn’t support encryption)

8.9.6 Data Signing

The Data Signing procedure is used to authenticate a data PDU sent over a non-encrypted link.

More details about the generation of the signature can be found in 8.2.6.

8.9.6.1 Subkeys Generation

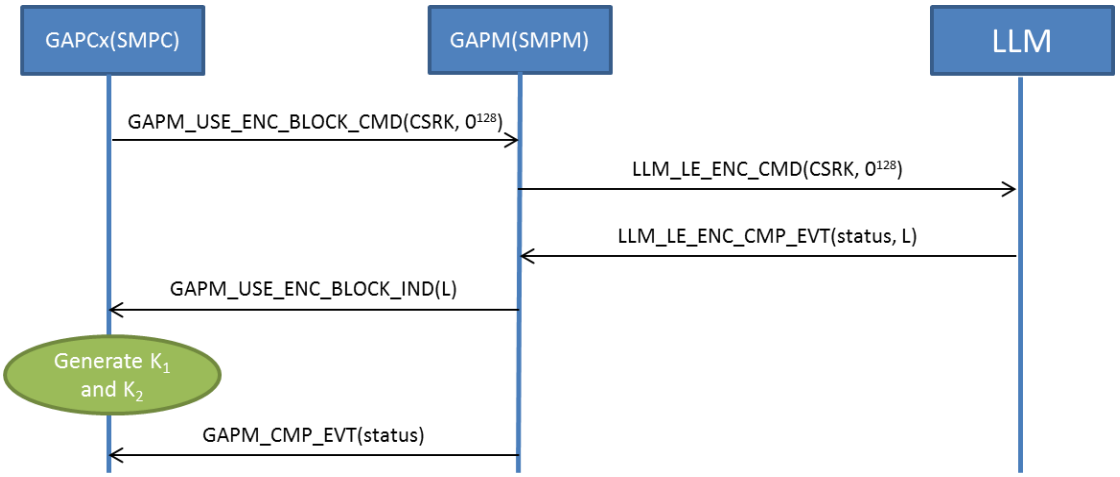

Figure 8-21: Data Signing: Subkeys Generation

8.9.6.2 MAC Generation

The data to be signed in the concatenation of the data PDU and the SignCounter value.

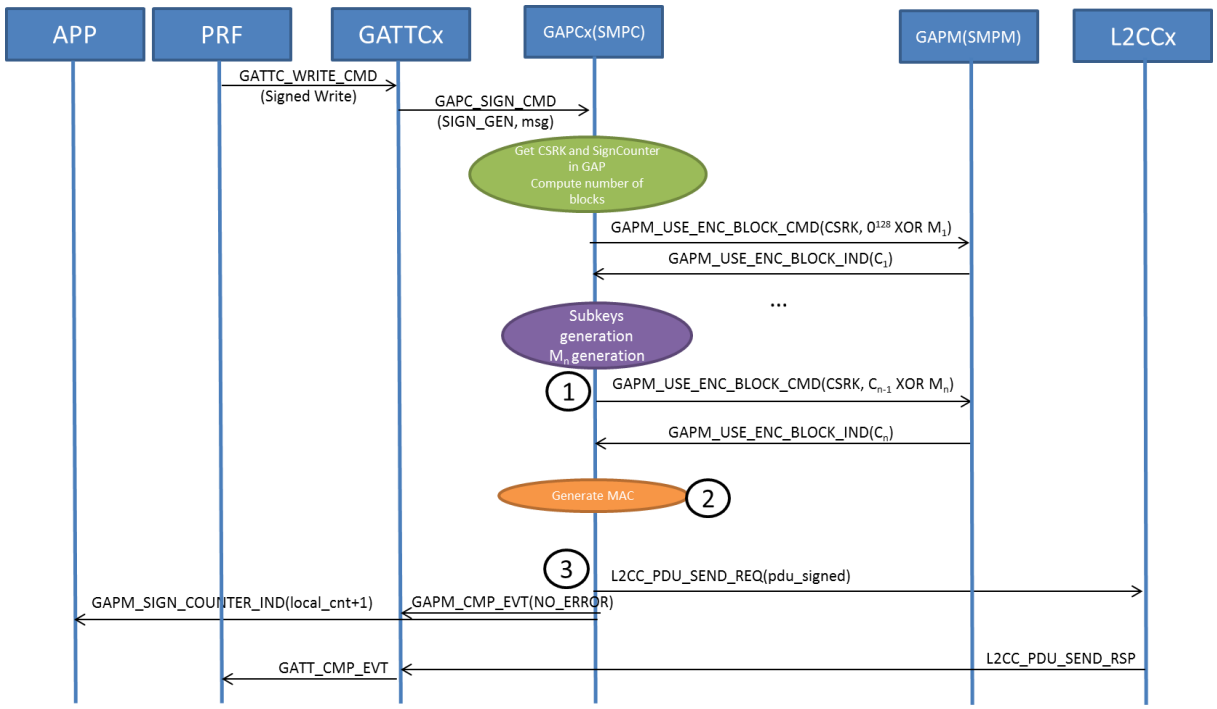

Figure 8-22: Data Signing: MAC Generation

- SMPC module receive a PDU message to sign from GATTC, it uses SMPM Encryption block through GAPM API.
- After using several times encryption block, it generate MAC signature and append it to the PDU.
- The signed PDU message is conveyed to L2CC with GATTC task as source ID in order to prevent a kernel reschedule. Application is also informed that sign counter has been increased.

8.9.6.3 MAC Verification

The verification of the received MAC is done by generating a MAC value based on the received data PDU and SignCounter values. If the generated MAC value matches with the received one, the signature is accepted.

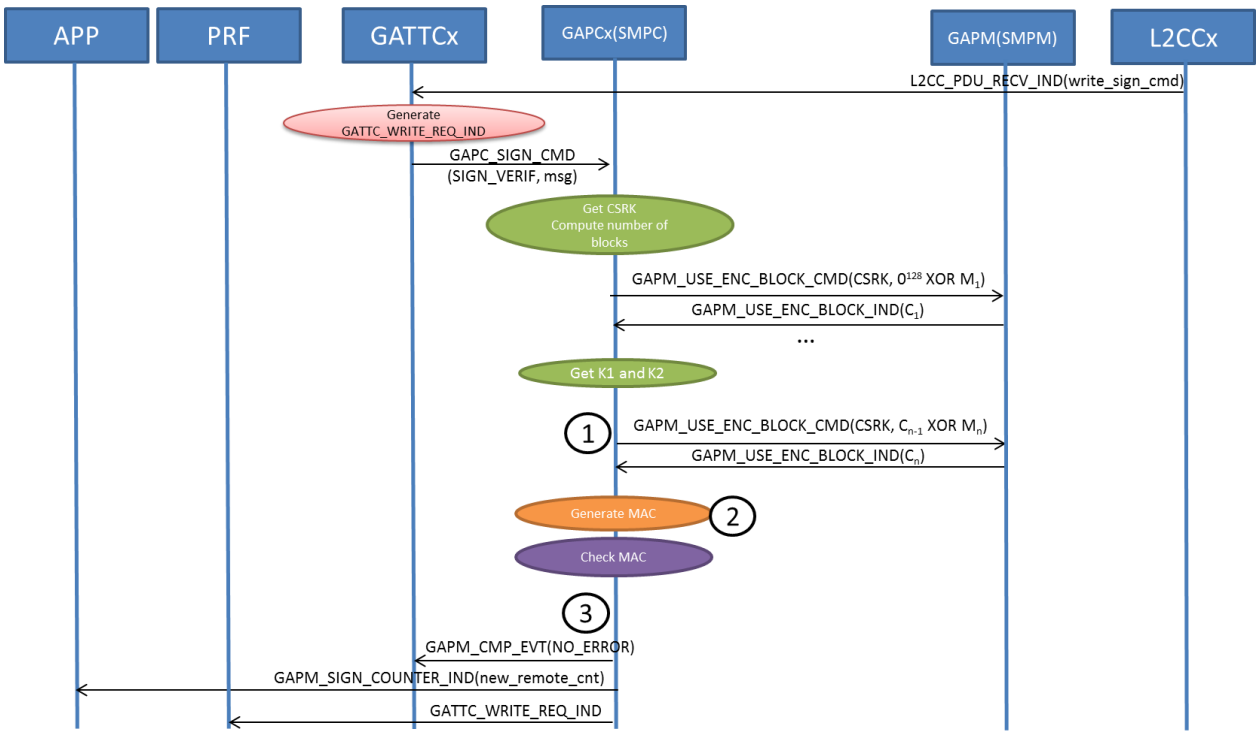

Figure 8-23: Data Signing: MAC Verification

- SMPC module receives a GATTC_WRITE_REQ_IND message with as signature to verify from GATTC, it uses SMPM Encryption block through GAPM API.
- After using several times encryption block, it generate MAC signature and compare to the provided signature.
- The GATTC_WRITE_REQ_IND message is sent to the targeted profile GATTC task as source ID in order to prevent a kernel reschedule. Application is also informed that remote sign counter has been increased. If an error occurs during signature, GATTC_WRITE_REQ_IND message is dropped and GATTC is inform that signature verification has failed.

8.9.7 Pairing Repeated Attempts

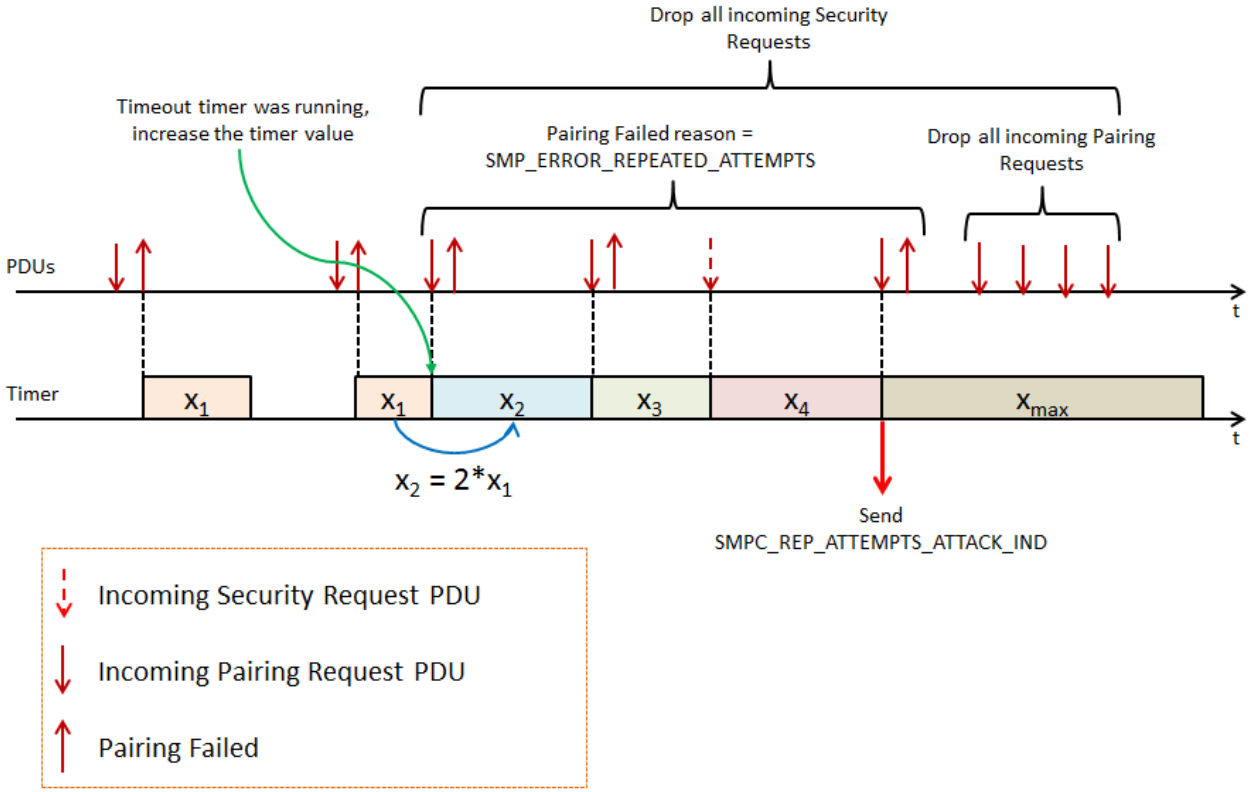

Figure 8-24: Repeated Attempts Protection

The Bluetooth specification [1] required the implementation of a mechanism

"When a pairing procedure fails a waiting interval shall pass before the verifier will initiate a new Pairing Request command or Security Request command to the same claimant, or before it will respond to a Pairing Request command or Security Request command initiated by a device claiming the same identity as the failed device. For each subsequent failure, the waiting interval shall be increased exponentially."

Figure 8-24 presents the mechanism implemented to rapidly detect an attack from a malicious device.

The minimal interval value is set to 2s and the maximal interval value is set to 30s. Thus, according to the procedure described in the Figure 8-24, a repeated attempt attack will be detected after 5 attempts.

8.10 Security Manager Protocol Data Unit Format

All SMP commands are transmitted over L2CAP using fixed channel with CID 0x0006 in Basic L2CAP mode. SMP has a fixed L2CAP MTU size of 23 octets. Only a single SMP command is sent per L2CAP frame.

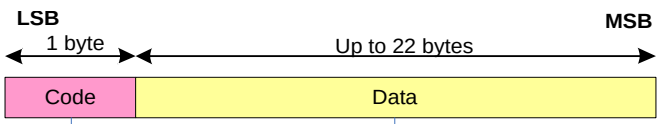

Figure 8-25: SMP Command PDU

8.10.1 SMP PDU Codes

The table below specifies the SMP codes. A packet with a code not included in the list below is ignored.

Code	Description
0x00	Reserved
0x01	Pairing Request
0x02	Pairing Response
0x03	Pairing Confirm
0x04	Pairing Random
0x05	Pairing Failed
0x06	Encryption Information
0x07	Master Identification
0x08	Identity Information
0x09	Identity Address Information
0x0A	Signing Information
0x0B	Security Request
0x0C	Public Key
0x0D	DHKey Check
0x0E	Keypress Notification

Table 8-4: SMP Codes

To ensure there is no lag during the procedure, an SM Timer is implemented allowing maximum 30s of delay between PDU transmissions on a device. This timer is reset and started upon transmission or reception of Pairing Request command. It is reset every time a command is queued for transmission. If the timer expires, failure is indicating to the Host and no more SMP exchanges are allowed. A new SM procedure may start once the physical link has been re-established.

9 LE Ping

LE Ping Feature is handled by Lower Layers and is described into BLE SW FS (see [9] Chapter 9.6.9). Application can configure or retrieve the Authenticated payload timeout (10ms step) per through GAP interface.

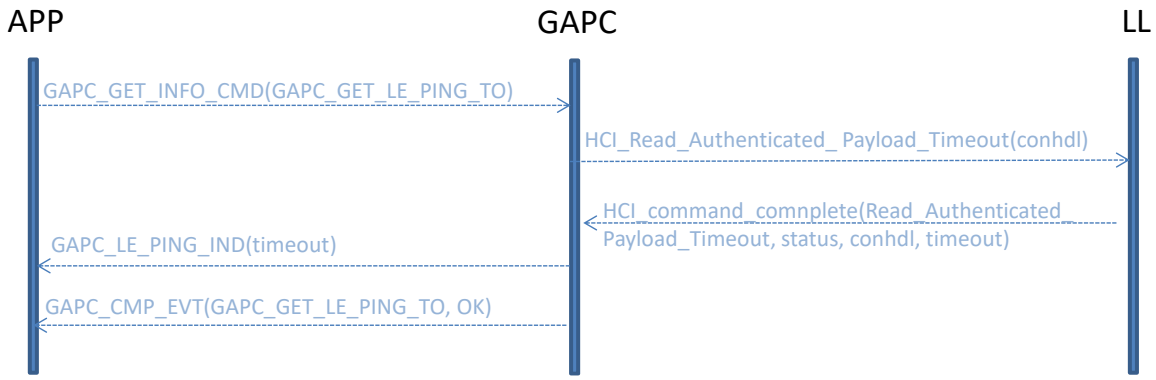

Figure 9-1: Retrieve LE Ping Authenticated payload timeout from LL.

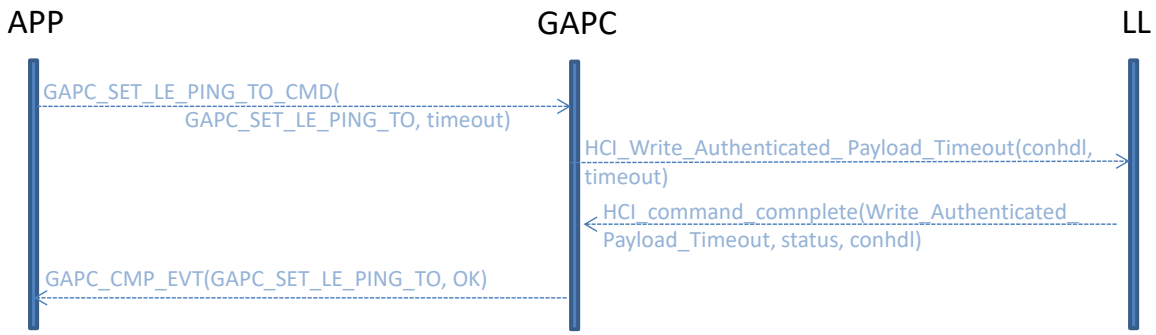

Figure 9-2: Modify LE Ping Authenticated payload timeout used by LL.

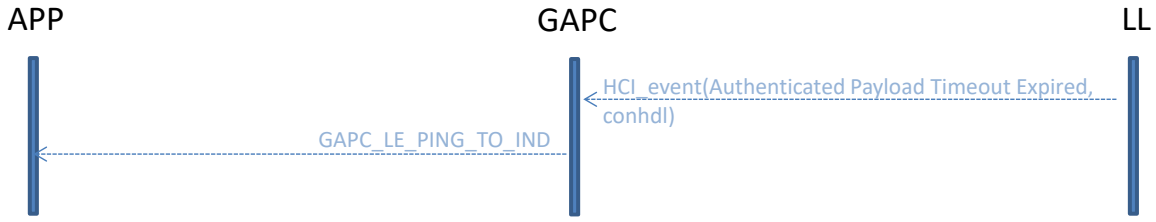

Figure 9-3: Inform Application about LE Ping Authenticated payload timeout expiration.

10 LE Data Packet Length Extension

Size of LE data packets can be negotiated over BLE Link. The preferred LE data packet size is set by application when setting device configuration (see 15.2).

Although, when link is established, application can try to (re)negotiate LE data packet size using GAPC_SET_LE_PKT_SIZE_CMD.

When Link size is updated, GAPC_LE_PKT_SIZE_IND is trigged. It doesn't change the fragmentation mechanism in L2CAP since it will use as much as possible fragmentation mechanism provided by lower layers.

11 LE Data Rate negotiation

Used for LE 2Mb/s or future rate required by features such as Long Range, the preferred LE Data Rates are set-up by application when setting device configuration (see 15.2).

Although, when link is established, application can try to (re)negotiate LE data rate using GAPC_SET_PHY_CMD.

When Link rate is updated, GAPC_LE_PHY_IND is triggered. It doesn't change the fragmentation mechanism in L2CAP since it will use as much as possible fragmentation mechanism provided by lower layers.

Finally it's possible to read the current link data rate using GAPC_GET_PHY operation within GAPC_GET_INFO_CMD message.

12 Profile Management

Our stack implementation supports a large amount of profiles; for each profiles, a minimum of two tasks is implemented, one for the profile, one for the client. Those tasks should support multiple connections.

In a normal use case, an application should not support all profile and services in same time; number of profile should be limited to a certain amount of profile tasks. To do so, an Array in Generic Access Profile environment variable is used to manage profile tasks. This array contains the task descriptor and a pointer to environment heap.

At start-up application decides profiles that can be started (both client and services tasks). For services task, it means that corresponding attribute database will be loaded, and a minimum authentication level is selected:

- No Authentication required
- Unauthenticated link required
- Authenticated link required
- Secure Connection link required

Profile manage allocation of its task state array, and its environment memory (static and for each links).

Number of profile tasks managed by Generic Access Profile is managed by a compilation flag.

Note: For integration purpose, the customer should allow this to be runtime configurable.

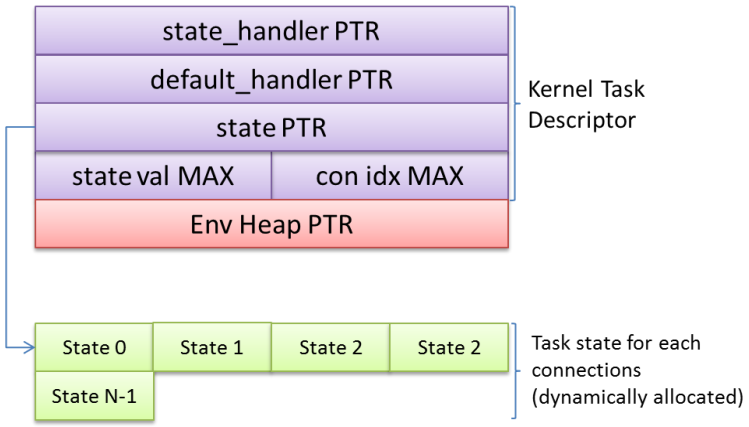

Figure 12-1: Overview of a Profile Task descriptor in GAP profile task array.

Note: When all profile tasks has been affected, an application requesting to use another profile will receive an OUT of Memory error.

In order to fix profile API, instead of using a task number, a profile id (statically set) is used. This ID should be unique and not be used by another task.

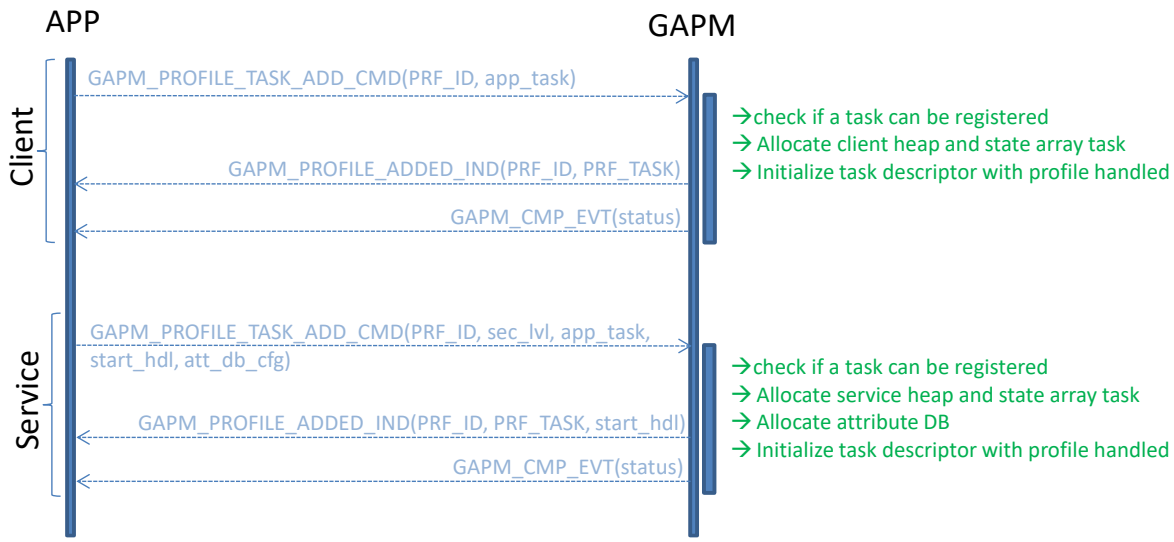

Figure 12-2: Profile Task registration.

For the GTL, in GAP environment, a specific array is used to retrieve correspondence between profile identifier and corresponding task id. GAP also provides a native API to retrieve profile id from task id or task id from profile id.

When a profile is registered, it is natively informed about link establishment (to allocate environment) and termination.

Note: When system is reset, all registered task are remove and profiles are cleaned-up.

By default Profile task descriptors are initialized without any handler and without any task id.

This ensures that when task is not registered, any message kernel to this task will be ignored.

Note: if GTL receive a message on a non-registered profile identifier, it should answer with a generic error message.

When an application has to communicate with a profile task, it has to request its task identifier to gap through its native API.

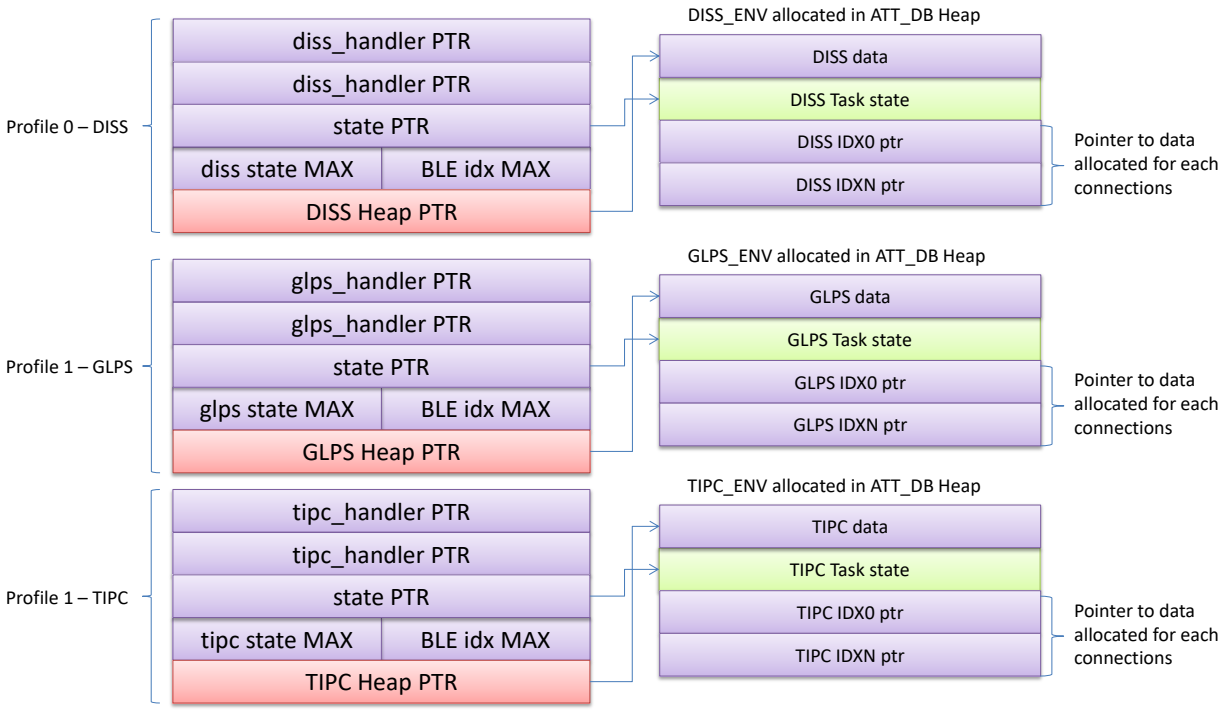

Figure 12-3: Example of Profile Task registration.

13 GAP service database

GAP service (UUID = 0x1800) will be represented as an attribute service in the Attribute database. Depending on the role of the device, certain attribute characteristics are required in the service definition.

CHARACTERISTICS	GAP ROLE			DESCRIPTION
	CT	PH	BC/OB	
(0x2A00) Device name	**m**	**m**	**x**	Name of the device in UTF-8 format (write optional)
(0x2A01) Appearance	**m**	**m**	**x**	Representation of the LE device (write optional)
(0x2A04) Preferred conn par	**x**	**o**	**x**	Set of conn parameters preferred by the device
(0x2A06) Central Addr Resolution	**o**	**o**	**x**	Central Address Resolution characteristic defines whether the device supports privacy with address resolution

Table 13-1: GAP Characteristics

Those characteristics value are not present into the database. If peer device try to read or write those values, a request will be sent to application. It allows application to manage memory position of those fields.

14 GAP Environment Variables

14.1 GAP Manager Environment

Type	Value	Comment
ke_msg*	CFG operation	Operation used to configure System, use encryption block, get system information
ke_msg*	AIR operation	Operation used to perform advertising, scanning or connection init activity
gap_bdaddr*	scan_filter	Scan filtering Array
co_list	reg_le_psm	Registered list of LE Protocol/Service Multiplexer for LE Credit Based Connection
uint16_t	svc_start_hdl	Gap Service start handle
uint16_t	renew_dur	Duration before regenerate device address when privacy is enabled.
gap_sec_key	IRK	IRK used for resolvable random BD address generation
bd_addr	addr	Current BD address (private or public)
uint8_t	role	Current device role
uint8_t	connections	Number of connections
uint8_t	cfg_flags	Flag field for: - Addr is private or public - Host Privacy Enabled - Controller Privacy Enabled - Use resolvable/non resolvable address - Slave preferred param present Address Renew timer started
uint8_t	pairing_mode	Pairing mode authorized
uint16_t	max_mtu	Maximum device MTU size
uint16_t	max_mps	Maximum device MPS size
t_public_key	public_key	Local device Public Key
uint16_t	audio_cfg	Audio configuration flag
bool	embedded_host	In Full mode, by default the AHI API is used, but if an HCI Reset is received, TL is switched to HCI and embedded host is disabled
uint8_t	max_nb_lecb	Maximum number of allowed LE Credit Based channels
uint8_t	nb_lecb	Current number of LE Credit Based channel connections established

Table 14-1: GAPM Environment variables

14.2 GAP Controller Environment

Type	Value	Comment
ke_msg*	Link Info operation	Operation used to manage Link info (get link and peer info)
ke_msg*	Link Param operation	Operation used to manage Link parameters (update parameters)
ke_msg*	SMP operation	Operation used to manage SMP
ke_msg*	LECBC operation	Operation used for LE Credit Based Connection
ke_task_id_t	disc_requester	Task id requested disconnection
uint16_t	conhdl	Connection handle
gap_sec_key[]	csrk	CSRK values (Local and remote)
uint32_t[]	sign_counter	signature counter values (Local and remote)
uint8_t	key_size	Encryption key size
gap_bdaddr[]	src	BD Address used for the link that should be kept
smpc_pair_info/ smpc_sign_info	pair_info/ sign_info	Pairing Information or sign info according to ongoing SMP procedure
uint8_t	SMP state	State of the current SMP procedure
uint8_t	fields	Configuration fields: - Link Authorization level - Encrypted Link - Role - Is SMP Timeout Timer running - Is Repeated Attempt Timer running Has task reached a SMP Timeout

Table 14-2: GAPC Environment variables

14.3 GAP Profiles Environment

Type	Value	Comment
prf_tasks_env[]	prf	Array of Profile tasks environment descriptor

Table 14-3: GAP Profiles Environment variables

15 Device initialization

15.1 Software Reset

At system start-up, to initialize Software state machines, a SW reset command shall be sent.

This command also initialize attribute database, after a SW reset, device attribute database is empty, Device configuration and Profile configuration shall be performed

15.2 Device Configuration

At system start-up, after sending software reset command, the device shall be set-up using the Set device configuration command.

Configuration of device can be updated only if there is no on-going connection.

- **Role**: Five role allowed

Note: The device can support all roles simultaneously, it means that it can be both master and slave of connection, and start scan and advertising activity in same time.

Roles	Scan	Advertise	Master Connect	Slave Connect	Periodic ADV	Periodic Sync Estab
Observer	X	O	O	O	O	X
Broadcaster	O	X	O	O	X	O
Peripheral	O	X	O	X	X	O
Central	X	O	X	O	O	X
All	X	X	X	X	X	X

Table 15-1: Device roles

- **Device Privacy:**
  - Device IRK: Used to generate random address (only valid for Host Privacy 1.1)
  - Privacy managed by host (privacy 1.1), by controller (privacy 1.2) or disabled
  - Renew address timer duration
- **Device Address**: (if privacy disabled or managed by controller)
  - Device address type
  - Device static address (if address type is random)
- **Packet Management**
  - Packet Size**:** Maximum MTU allowed by device (mini = 23 bytes, max = 2048)
  - Preferred Transmit size for Data length extension
  - Preferred LE Rate
- **GAP DB configuration:**
  - GAP DB start handle (0x0000 – dynamically allocated)
  - Appearance Write Permissions
  - Device Name Write Permissions + Device name max length
  - Peripheral Preferred Connection Parameters present + Read Permissions
- **GATT DB configuration:**

- GATT DB start handle (0x0000 – dynamically allocated)
- Service changed characteristic present.

- **L2CAP channel per connection**:
  - Maximum number of L2CAP Channel per connection
  - Maximum MTU and MPS size authorized on local device. It also limits maximum MTU and MPS size that can be transmitted to peer device.

Note: Set device configuration command recreate GAP and GATT database.

References

[1]	**Title**	Specification of the Bluetooth
	Reference	Bluetooth Specification
	Source	Bluetooth SIG

[2]	**Title**	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
	Reference	SP_800-38B
	Source	National Institute of Standards and Technology

[3]	**Title**	RivieraWaves Kernel
	Reference	RW-BT-KERNEL-SW-FS
	Source	RivieraWaves

[4]	**Title**	Host Software Functional Specification
	Reference	RW-BLE-HOST-SW-FS
	Source	RivieraWaves

[5]	**Title**	L2CAP Software Functional Specification
	Reference	RW-BLE-L2CAP-SW-FS
	Source	RivieraWaves

[6]	**Title**	Generic Attribute Software Functional Specification
	Reference	RW-BLE-GATT-SW-FS
	Source	RivieraWaves

[7]	**Title**	RW HCI Software
	Reference	RW-HCI-SW-FS
	Source	RivieraWaves

[8]	**Title**	GAP Interface Specification
	Reference	RW-BLE-GAP-IS
	Source	RivieraWaves

[9]	**Title**	RW-BLE Link Layer Software
	Reference	RW-BLE-LL-SW-FS
	Source	RivieraWaves