Qualcomm Technologies International, Ltd.

# Audio Sink Application GATT

## User Guide

80-CT429-1 Rev. AE

October 31, 2017

Qualcomm Technologies International, Ltd. (formerly known as Cambridge Silicon Radio Limited) is a company registered in England and Wales with a registered office at: Churchill House, Cambridge Business Park, Cowley Road, Cambridge, CB4 0WZ, United Kingdom.
Registered Number: 3665875 | VAT number: GB787433096

# Revision history

| Revision | Date | Description |
|---|---|---|
| 1 | JUN 2015 | Initial release. Alternative document number CS-00329966-UG. |
| 2 | SEP 2015 | Editorial corrections |
| 3 | SEP 2016 | Updated for ADK 4.1. Updated to QTI standards. |
| 4 | APR 2017 | Final edits for release |
| AE | OCT 2017 | Added to the Content Management System. DRN updated to use Agile number. No change to the technical content. |

# Contents

# Tables

# Figures

# 1 Audio Sink application GATT overview

The Audio Sink application provides support for multiple GATT services. This support is provided using VM libraries that help the application manage complex GATT connections.

**NOTE** Read with the *Audio Sink Application User Guide*.

### GATT library

The GATT library manages multiple connections with GATT devices. It provides a high level API providing a simple interface for making use of GATT connections.

### GATT manager library

The GATT Manager is a slim layer that sits just above the GATT library and is designed to manage complex GATT connections by.

- Demultiplexing incoming GATT messages to different GATT Service libraries.
- Direct outgoing GATT requests and responses to particular remote devices.
- Allow GATT Service libraries to run without having to store knowledge about active GATT connections.

### GATT service libraries

The GATT Service libraries are designed to manage a particular GATT service and must be implemented to provide support for a GATT server or a GATT client instance of that particular GATT service.

### GATT server libraries

GATT Server libraries are libraries that manage a particular GATT service when a remote device is acting as the GATT client for that particular service. The available GATT Server libraries are:

- GATT Service
- GAP Service
- Battery Service
- Device Information Service
- Link Loss Service
- Transmit Power Service
- GAIA Service

- Immediate Alert Service
- Heart Rate Service

Each GATT Server library is responsible for managing part of the locally registered GATT database.

**NOTE** GATT Server libraries must not have overlapping handles within the locally registered GATT Database. For example, one server library cannot register to manage handles 3 to 7 while another server library has been registered to manage handle 1 to 5.

**GATT client libraries**

GATT Client libraries manage a particular GATT service when a remote device is acting as the GATT Server for that particular service. The available GATT Client libraries are:

- Scan Parameters
- GATT Service
- Battery Service
- Device Information Service
- HID Over GATT Service
- Immediate Alert Service
- Apple Notification Centre Service
- Heart Rate Profile

Each GATT Client library is responsible for managing part of the GATT database registered on a connected remote device.

**NOTE** GATT Client libraries must not have overlapping handles within the remote devices GATT Database. For example, one client library cannot register to manage handles 3 to 7 while another client library has been registered to manage handles 1 to 5.

# 2 GATT system architecture

The Audio Sink application provides GATT support using a layered architecture where functionality is split across multiple VM libraries, see Figure 2-1.

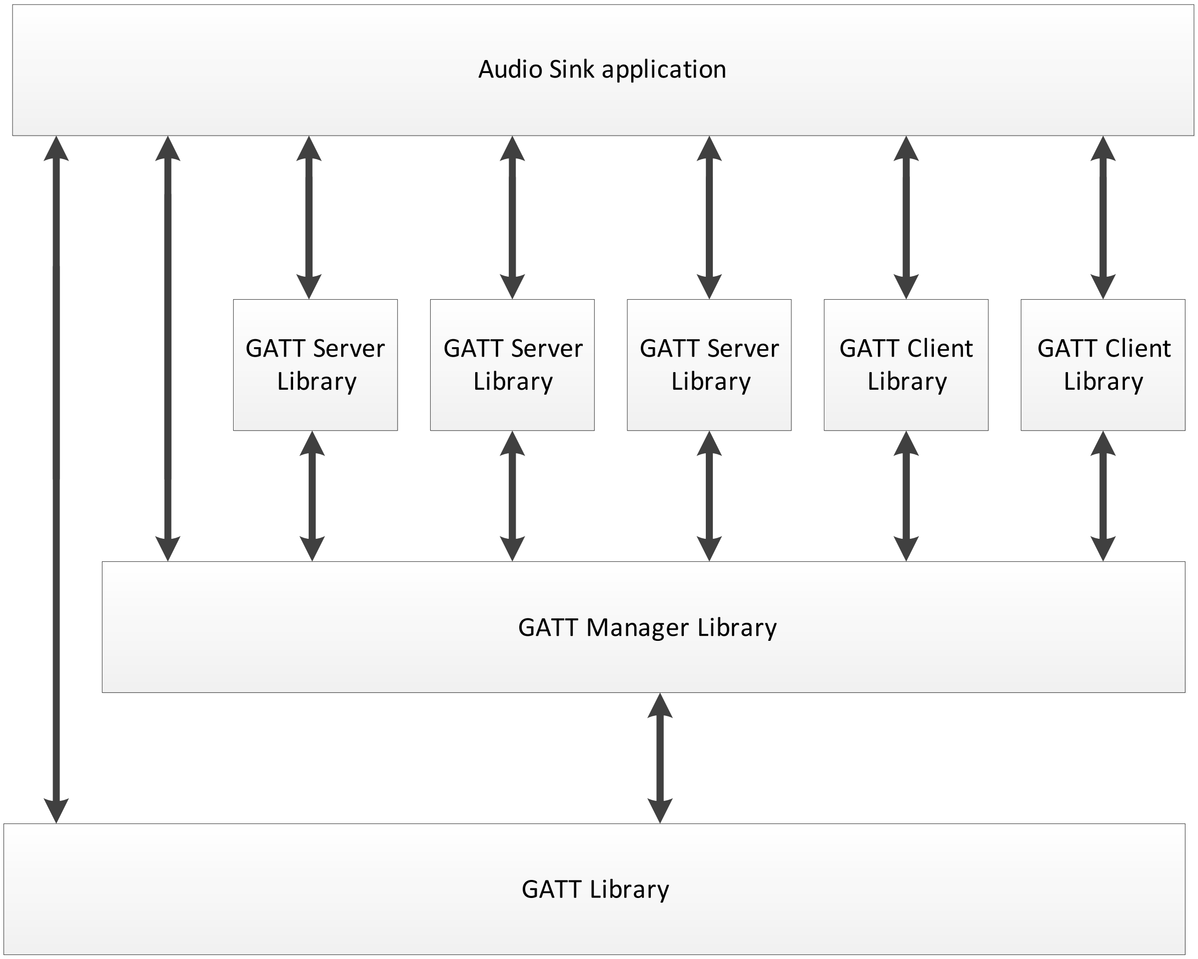

**Figure 2-1 GATT system architecture**

# 3 Bluetooth low energy operational modes

The Audio Sink application supports two modes of operation for Bluetooth low energy connections. These operational modes and how they can be configured are described in Audio Sink application Central mode and Audio Sink application Peripheral mode.

**NOTE** Bluetooth low energy scatternet scenario is supported, where the Audio sink application can be connected to both a central device as well as a peripheral device w.r.t modes of operation this would mean that Audio Sink application supports both modes being enabled at the same time.

## 3.1 Basic Bluetooth low energy scatternet

Figure 3-1 shows the application software flow for setting a basic Bluetooth low energy scatternet scenario with earlier bonded devices.

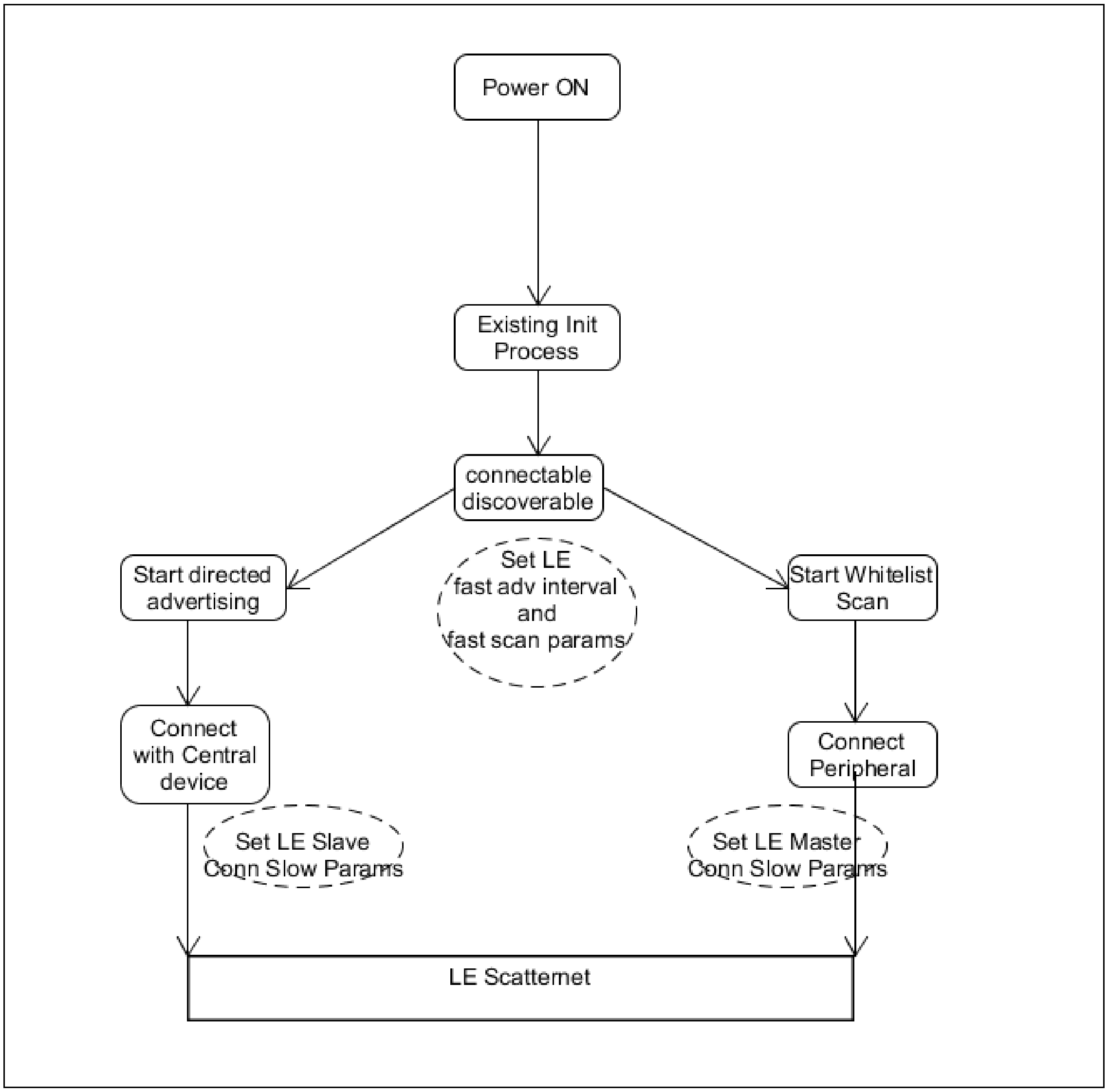
Power ON
Existing Init Process
connectable discoverable
Start directed advertising
Set LE fast adv interval and fast scan params
Start Whitelist Scan
Connect with Central device
Connect Peripheral
Set LE Slave Conn Slow Params
Set LE Master Conn Slow Params
LE Scatternet

**Figure 3-1 Bluetooth low energy scatternet while connecting**

**NOTE** To handle devices that support random address, after a timeout of a whitelist scan, the device searches all adverts and checks if one is already bonded (that is, has a resolvable address) and if one is found, a connection is initiated to the device.

## 3.2 Audio Sink application Central mode

When operating in Central mode, the Audio Sink application is able to find and connect to nearby Peripheral devices. Central mode is used when the Audio Sink application is required to make use of GATT services offered by Peripheral devices, for example a Bluetooth low energy remote controller.

## 3.3 Audio Sink application Peripheral mode

When operating in Peripheral mode, the Audio Sink application is able to advertise GATT services and be connected to by Central Devices. Peripheral mode is used when the Audio Sink application is required to offer GATT services to Central devices, for example to allow Central devices to read the local battery level.

# 4 Bluetooth low energy Bondable mode

The Audio Sink application has a Bluetooth low energy bondable mode that is used to bond with new devices. Bluetooth low energy bondable mode is described in Audio Sink application events for control of Bluetooth low energy Bondable mode and Audio Sink application bonding procedure.

- When Bluetooth low energy bondable mode is active, the Audio Sink application can bond with connected GATT devices.
- When Bluetooth low energy bondable mode is not active, the Audio Sink application cannot bond with connected GATT devices.
- The Audio Sink application never automatically starts Bluetooth low energy bondable mode.

## 4.1 Audio Sink application events for control of Bluetooth low energy Bondable mode

Table 4-1 lists the Audio Sink application events used to control Bluetooth low energy Bondable mode.

**Table 4-1 Audio sink application events for connecting with new Bluetooth low energy devices**

| Audio Sink Application Event | Description |
|---|---|
| `EventUsrBleStartBonding` | Start Bluetooth low energy bondable mode. |
| `EventSysBleBondableConnectionTimeout` | Indicates that Bluetooth low energy bondable mode has timed out before a device has connected. |
| `EventSysBleBondablePairingTimeout` | Indicates that Bluetooth low energy bondable mode has timed out while connected to an un-bonded device. |

## 4.2 Audio Sink application bonding procedure

To be able to scan/advertise and bond with a new Bluetooth low energy Peripheral/Central device, the Audio Sink application must first become bondable by starting Bluetooth low energy bondable mode. When Bluetooth low energy bondable mode has been started, the Audio Sink application can start scanning as well as advertising for new devices based on the configuration of peripherals and centrals from the Sink configuration tool.

The Audio Sink application can scan unknown devices. When a Peripheral device is discovered, a GATT connection request is actioned.

**NOTE** Bluetooth low energy bondable mode does not guarantee discovered peripheral devices are unknown. The discovered Peripheral device could already be bonded with the Audio

Sink application. Also while in bondable state the audio sink application supports only one role at a time depending on the remote device with which the GATT connection is first established (peripheral/central) after which the other role is disabled until bondable state is exited.

### 4.2.1 Audio Sink application bonding with a peripheral device

As soon as the GATT connection has been successfully established, the Audio Sink application parses the GATT database of the Peripheral device.

- If the Peripheral device does not support any GATT services that are supported by the Audio Sink application, the Audio Sink application disconnects the Peripheral device and a bonding request is not issued.
- If the Peripheral device supports at least one GATT service that is supported by the Audio Sink application, the Audio Sink application requests encryption and bonding:
    - If the bonding request fails, the Audio Sink application disconnects the Peripheral device.
    - If the bonding request succeeds, the Audio Sink application makes use of the Peripheral device’s supported GATT services.

### 4.2.2 Audio Sink application bonding with central device

When a remote central device establishes a GATT connection, it can discover all of the GATT services supported by the Audio Sink application and can initiate bonding procedure. However if the remote device does not initiate bonding until the expiry of the bondable timer the Audio Sink application as a peripheral initiates a security request.

# 5 GATT initialization

Example GATT initialization gives an example of the initialization process for the GATT subsystem.

**NOTE** To comply with Bluetooth specifications, the Audio Sink application must always register both the GATT and GAP server tasks. This must be done regardless of whether other server tasks are required or not.

## 5.1 Example GATT initialization

The following describe an example of how to initialize the GATT subsystem for the Audio Sink application when a local GATT database is required:

Local GATT database

GATT server libraries

Example GATT initialization procedure

In the example, the Audio Sink application registers two GATT primary services, GATT Service A and GATT Service B.

### 5.1.1 Local GATT database

Table 5-1 describes the example GATT database to be registered locally.

**Table 5-1 Example GATT database**

| GATT Primary Service ID | Service Characteristics | |
|---|---|---|
| | Handle | Characteristic ID |
| GATT Service A | 1 | A1 |
| | 2 | A2 |
| GATT Service B | 3 | B1 |
| | 4 | B2 |
| | 5 | B3 |

### 5.1.2 GATT server libraries

The locally registered database (defined in Table 5-1) contains two GATT Services. To manage each service within the database, GATT Server libraries supporting these services should be used. Table 5-2 describes the example GATT Server libraries required.

**Table 5-2 Example GATT server libraries**

| GATT Server Library | Description |
|---|---|
| GATT Server Service A | GATT Server library to manage the GATT Service A portion of the locally registered GATT database. |
| GATT Server Service B | GATT Server library to manage the GATT Service B portion of the locally registered GATT database. |

### 5.1.3 Example GATT initialization procedure

The local database and the GATT Server libraries to manage the local GATT database must be registered. This section describes the procedure required to do this, a few points to note are:

- The local GATT Database must be registered first.
- Each required GATT Server library must be individually registered:
  - Only GATT Server Libraries that manage a section of the registered GATT database should be registered.
  - The handle range for each GATT Server library must not overlap.
- When the local GATT database and all the required GATT Server libraries have been registered, the device can be registered with GATT (and the ATT firmware layer):
  - If a local GATT database is registered, but no GATT Server libraries have been registered, this request fails.

Figure 5-1 describes the API interactions required to register and initialize the local GATT Database in the example described.

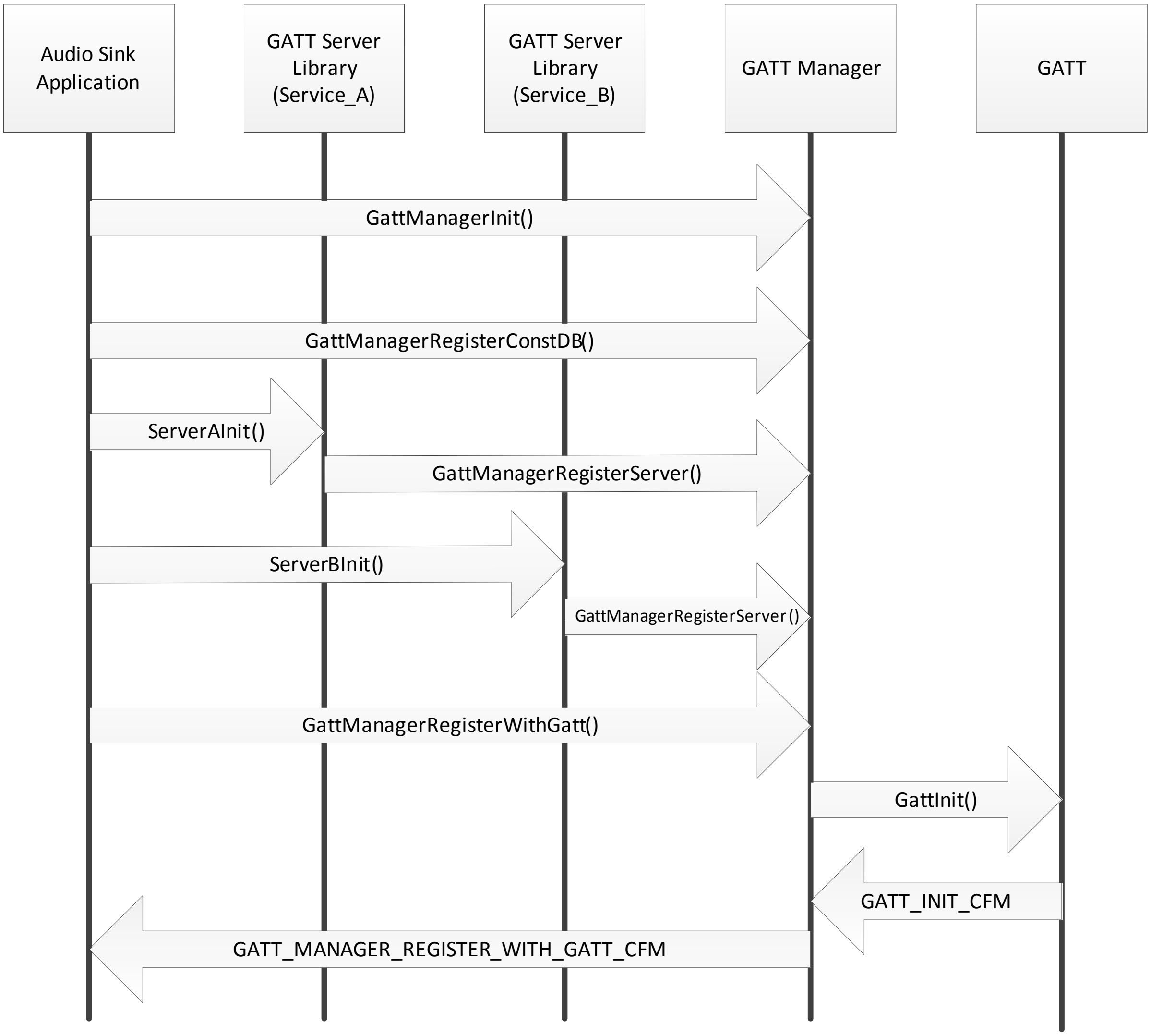
Audio Sink Application
GATT Server Library (Service_A)
GATT Server Library (Service_B)
GATT Manager
GATT
GattManagerInit()
GattManagerRegisterConstDB()
ServerAInit()
GattManagerRegisterServer()
ServerBInit()
GattManagerRegisterServer()
GattManagerRegisterWithGatt()
GattInit()
GATT_INIT_CFM
GATT_MANAGER_REGISTER_WITH_GATT_CFM

**Figure 5-1 GATT Initialization process when GATT server role is required**

# 6 GATT device connection

## 6.1 Remote device issues GATT connection

Waiting for a remote GATT connection and Canceling wait for a GATT remote connection describe how the Audio Sink application manages connections when the remote device initiates the connection. In this case, the Audio Sink application is in the Peripheral role, and the remote device is in the Central role.

### 6.1.1 Waiting for a remote GATT connection

After the Audio Sink application has been initialized with GATT Server libraries (see ), remote devices can be connected. Figure 6-1 shows the procedure for starting advertising and allowing a remote device to connect.

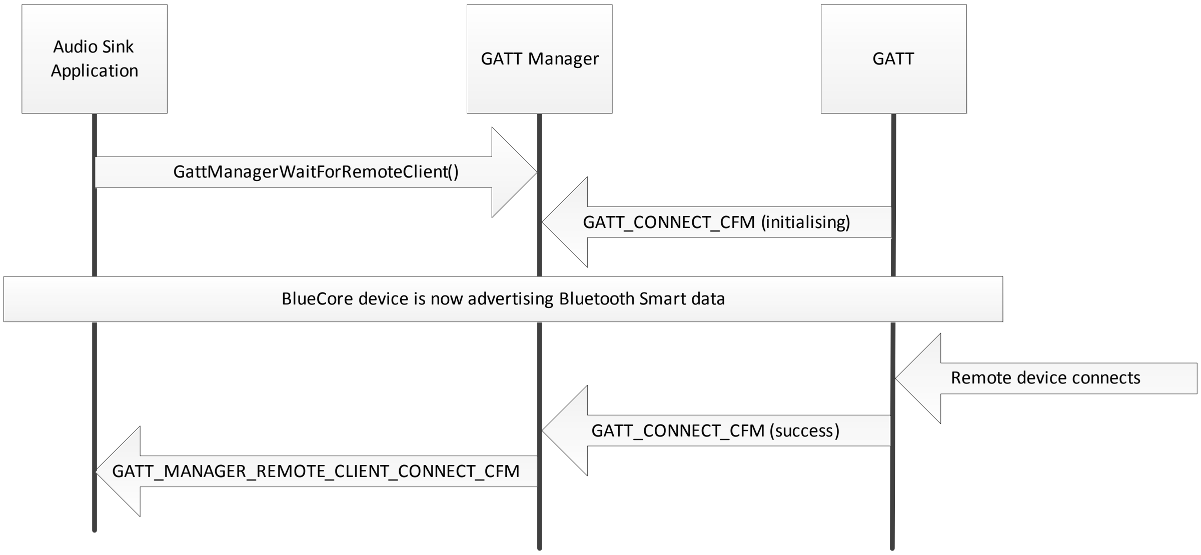
Audio Sink Application
GATT Manager
GATT
GattManagerWaitForRemoteClient()
GATT_CONNECT_CFM (initialising)
BlueCore device is now advertising Bluetooth Smart data
Remote device connects
GATT_CONNECT_CFM (success)
GATT_MANAGER_REMOTE_CLIENT_CONNECT_CFM

**Figure 6-1 Connection initiated from a remote device**

**NOTE** Bluetooth specifications state that Peripheral devices must not connect to a Central device. They must advertise and wait for a Central device to discover them and then connect.

### 6.1.2 Canceling wait for a GATT remote connection

In some cases, the Audio Sink application may need to cancel the request to wait for a remote device to connect (which puts the Audio Sink application into Bluetooth low energy advertising mode). Figure 6-2 shows the procedure that must be followed to cancel the advertising.

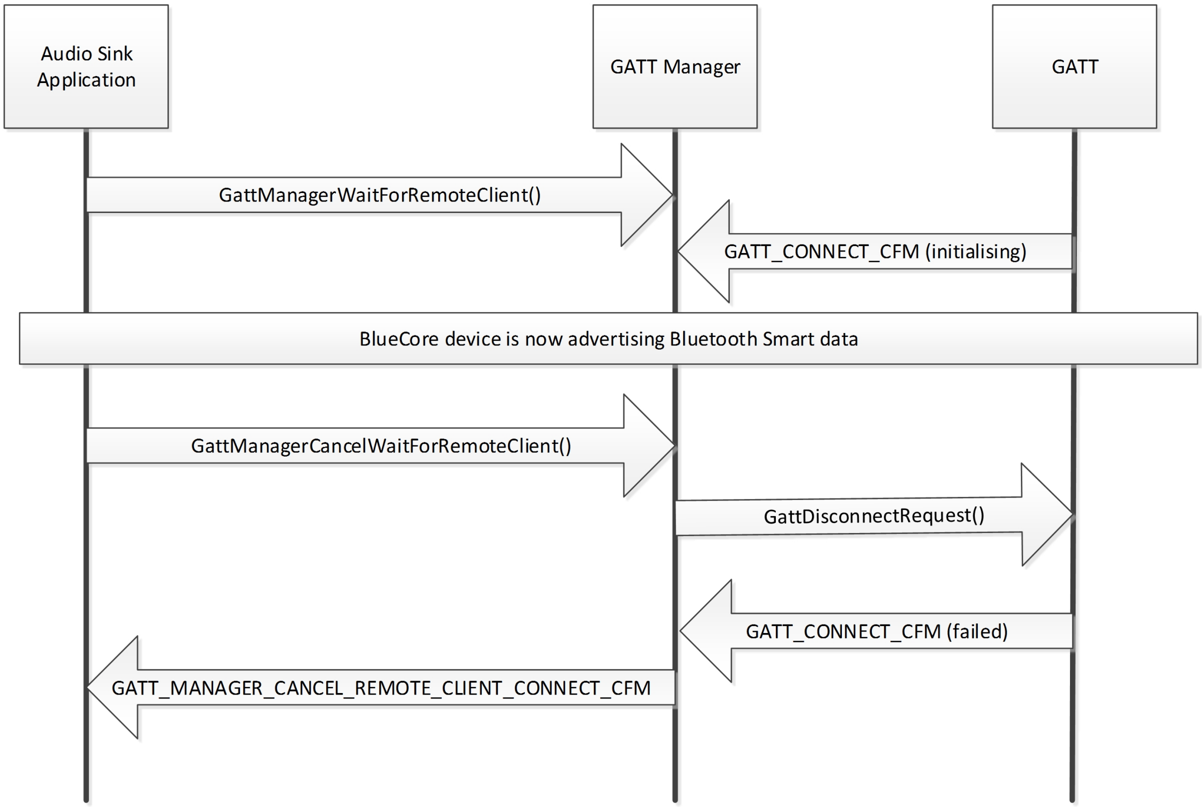
Audio Sink Application
GATT Manager
GATT
GattManagerWaitForRemoteClient()
GATT_CONNECT_CFM (initialising)
BlueCore device is now advertising Bluetooth Smart data
GattManagerCancelWaitForRemoteClient()
GattDisconnectRequest()
GATT_CONNECT_CFM (failed)
GATT_MANAGER_CANCEL_REMOTE_CLIENT_CONNECT_CFM

**Figure 6-2 Canceling wait for a remote connection**

## 6.2 Audio sink device initiates a GATT connection

The Audio Sink application manages connections when it initiates the connection as described below. In this case, the Audio Sink Application is in the Central role, and the remote device is in the Peripheral role.

## GATT connecting to a remote device

Figure 6-3 describes the process of initiating a connection from the application.

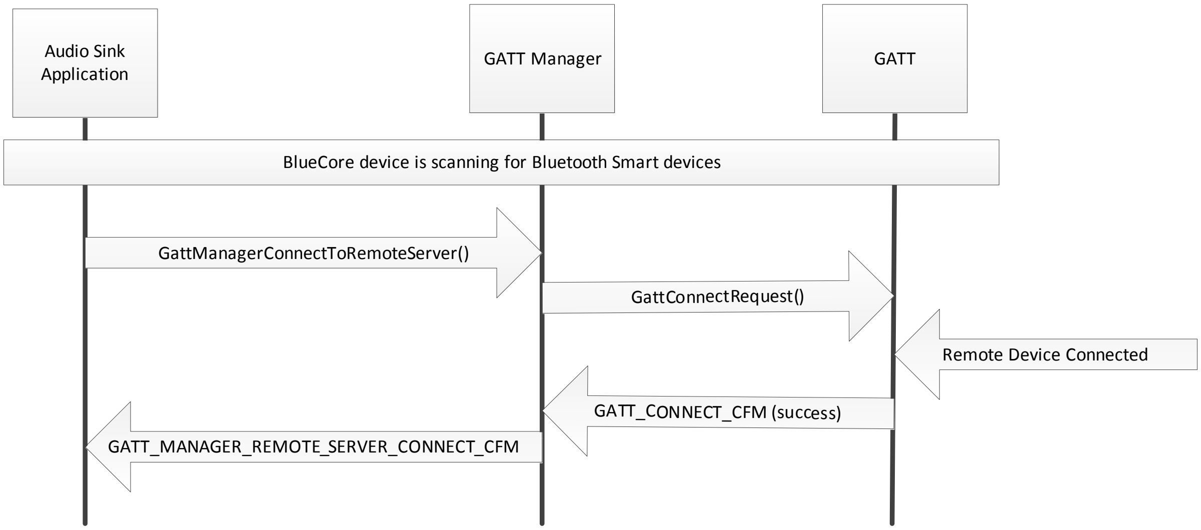
Audio Sink Application
GATT Manager
GATT
BlueCore device is scanning for Bluetooth Smart devices
GattManagerConnectToRemoteServer()
GattConnectRequest()
Remote Device Connected
GATT_CONNECT_CFM (success)
GATT_MANAGER_REMOTE_SERVER_CONNECT_CFM

**Figure 6-3 Connection initiated from the local application**

# 7 GATT server procedures

When a connection has been established, GATT transactions can begin within the Audio Sink application. This section details the case when the Audio Sink Application is acting as a GATT Server, and the remote device is acting as a GATT Client.

## 7.1 GATT server library handling a request

The GATT Manager library routes each incoming GATT request from a connected remote device to the appropriate GATT Server library. The library must then respond to the incoming GATT Request using the GATT Manager library:

- In a minority of situations, the GATT Server library is able to respond to the GATT request immediately. For example, if the request is to retrieve constant data, for example, the software version.
- In most cases, the GATT Server library passes the incoming GATT request on to the application layer. The GATT Server library should provide a downstream API for the application to respond to the request.

Figure 7-1 shows an example transaction where the GATT Server library handles an incoming GATT request.

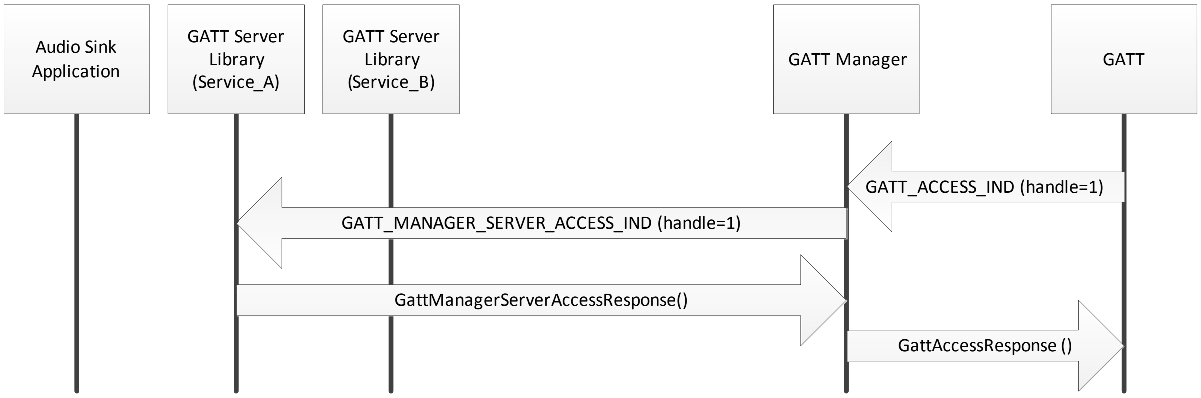
Audio Sink Application
GATT Server Library (Service_A)
GATT Server Library (Service_B)
GATT Manager
GATT
GATT_ACCESS_IND (handle=1)
GATT_MANAGER_SERVER_ACCESS_IND (handle=1)
GattManagerServerAccessResponse()
GattAccessResponse ()

**Figure 7-1 GATT requests handled by the GATT server library**

**Description**

- The GATT Manager passes the request to the GATT Server library registered to handle that item of the database.
- The GATT Server library parses the request and immediately responds using the GATT Manager API.

**NOTE** The incoming request is made to handle 1 of the local GATT database. This handle is managed by GATT Server Library Service A so the GATT Manager forwards the request to this library.

## 7.2 Audio sink application handling an incoming GATT request

Figure 7-2 shows an example transaction where the application handles an incoming GATT request.

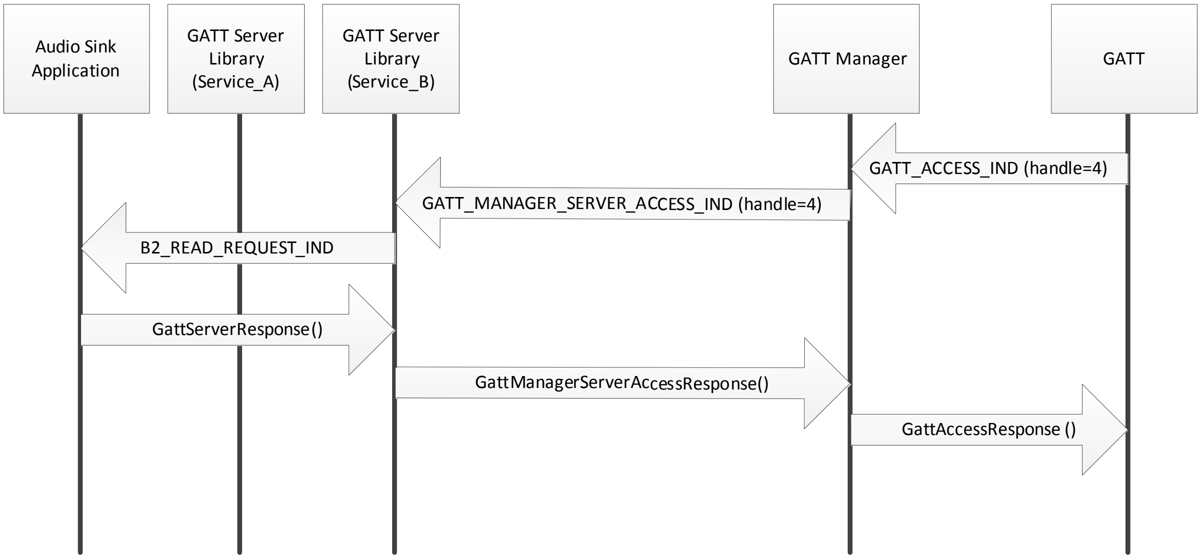
Audio Sink Application
GATT Server Library (Service_A)
GATT Server Library (Service_B)
GATT Manager
GATT
GATT_ACCESS_IND (handle=4)
GATT_MANAGER_SERVER_ACCESS_IND (handle=4)
B2_READ_REQUEST_IND
GattServerResponse()
GattManagerServerAccessResponse()
GattAccessResponse ()

**Figure 7-2 GATT requests handled by the application layer**

**Description**

- The GATT Manager passes the request to the GATT Server library registered to handle that item of the database.
- The GATT Server library parses the request. If possible, the library responds to the request immediately using the GATT Manager API. If the library is not able to respond to the request, it passes the request to the application layer in the form of an API message:
  - The application layer responds to the request by calling a library API of the GATT Server library.
  - The GATT Server library responds to the request using the GATT Manager API.

**NOTE** The incoming request is made to handle 4 of the local GATT database. This handle is managed by GATT Server Library Service B so the GATT Manager forwards the request to this library.

## 7.3 GATT server initiated procedures

When a device has been connected, the remote GATT Server can initiate the Notification and Indication procedures. The GATT Manager VM library provides an API for the GATT Server initiated procedures:

- Send GATT Notification
- Send GATT Indication

For more details on the API, see the GATT Manager library code documentation. Figure 7-3 describes an example GATT transaction for sending data to the remote GATT Server device. The message exchanges shown are typical for most GATT Manager APIs.

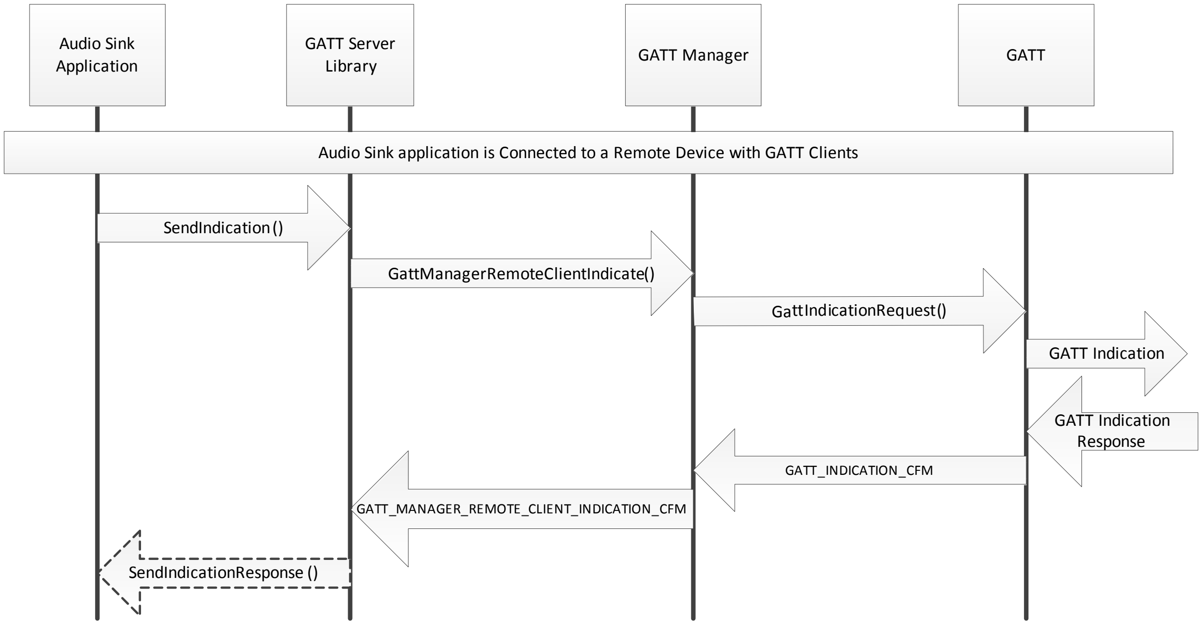
Audio Sink Application
GATT Server Library
GATT Manager
GATT
Audio Sink application is Connected to a Remote Device with GATT Clients
SendIndication()
GattManagerRemoteClientIndicate()
GattIndicationRequest()
GATT Indication
GATT Indication Response
GATT_INDICATION_CFM
GATT_MANAGER_REMOTE_CLIENT_INDICATION_CFM
SendIndicationResponse ()

**Figure 7-3 Sending indications using the GATT server library**

# 8 GATT client procedures

When a connection has been established, GATT transactions can begin within the Audio Sink application. This section details the case when the Audio Sink Application is acting as a GATT Client, and the remote device is acting as a GATT Server.

## 8.1 GATT service discovery

When a device has been successfully connected and the Audio Sink application is acting as a GATT Client, the Audio Sink application starts discovering which primary services are supported by the connected device. The procedure for discovering the GATT Primary Services of a remote GATT Server is:

- The Audio Sink application discovers all primary services defined on the remote GATT Server:
  - Discovered primary services that are supported by the Audio Sink application are temporarily stored in a list.
  - Discovered primary services that are not supported by the Audio Sink application are ignored.
- When all primary services have been discovered:
  - For each supported primary service that has been discovered, the Audio Sink application initializes the GATT client library used to manage the service. Library initialization is done sequentially not simultaneously.
  - If no supported primary services have been discovered, the device is disconnected.

### 8.1.1 Discovering supported GATT primary services

In the example exchange shown in Figure 8-1, the Audio Sink application supports one GATT service, Service One. The Remote GATT Server supports two primary services, Service One and Service Two.

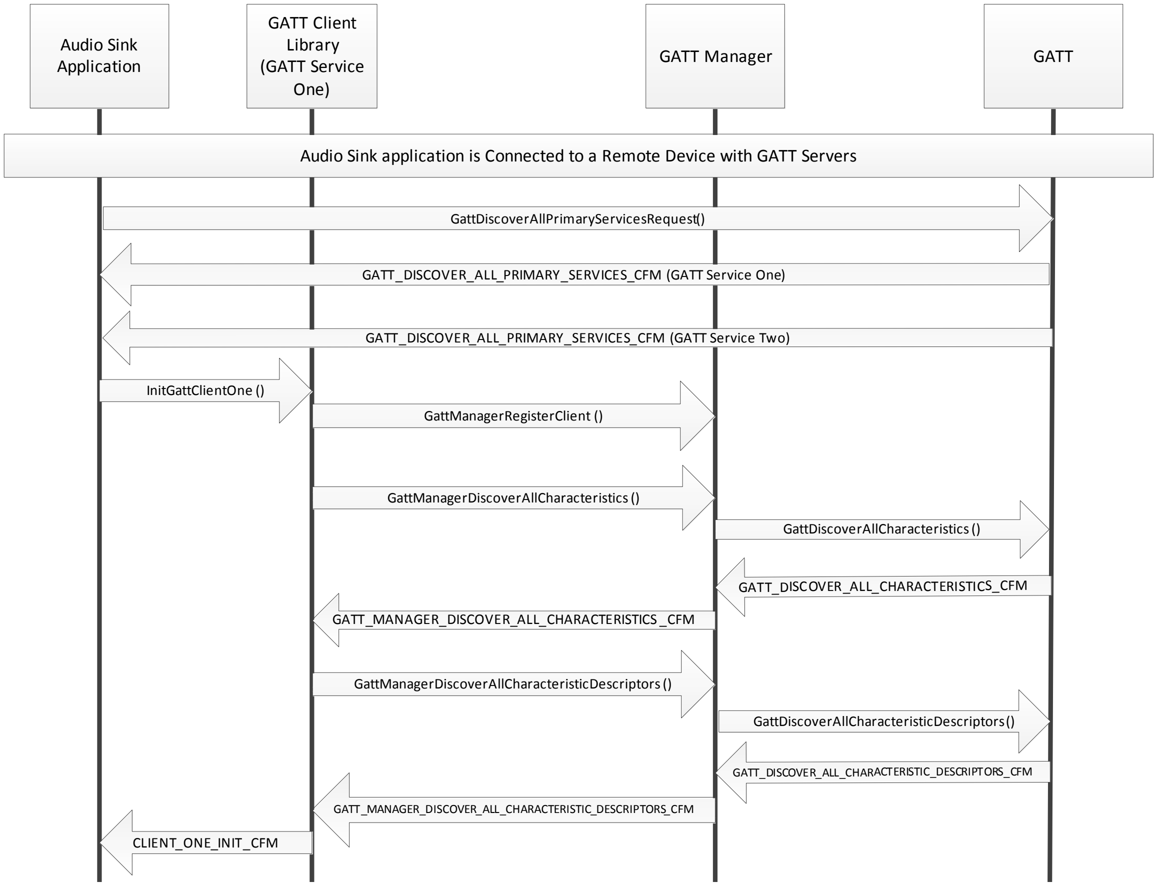
Audio Sink Application
GATT Client Library (GATT Service One)
GATT Manager
GATT
Audio Sink application is Connected to a Remote Device with GATT Servers
GattDiscoverAllPrimaryServicesRequest()
GATT_DISCOVER_ALL_PRIMARY_SERVICES_CFM (GATT Service One)
GATT_DISCOVER_ALL_PRIMARY_SERVICES_CFM (GATT Service Two)
InitGattClientOne ()
GattManagerRegisterClient ()
GattManagerDiscoverAllCharacteristics ()
GattDiscoverAllCharacteristics ()
GATT_DISCOVER_ALL_CHARACTERISTICS_CFM
GATT_MANAGER_DISCOVER_ALL_CHARACTERISTICS_CFM
GattManagerDiscoverAllCharacteristicDescriptors ()
GattDiscoverAllCharacteristicDescriptors()
GATT_DISCOVER_ALL_CHARACTERISTIC_DESCRIPTORS_CFM
GATT_MANAGER_DISCOVER_ALL_CHARACTERISTIC_DESCRIPTORS_CFM
CLIENT_ONE_INIT_CFM

**Figure 8-1 Discovering primary services on a remote GATT server**

**Description**

- The Audio Sink application discovers all primary services on the remote GATT Server.
- The Audio Sink application discovers the remote GATT Server supports Service One and temporarily stores that information.
- The Audio Sink application ignores the discovered primary service (Service Two) that is not supported.
- When all primary services have been discovered, the Audio Sink application initializes the GATT Client library to manage Service One.

### 8.1.2 No supported GATT primary services discovered

In the example exchange shown in Figure 8-2, the Audio Sink application supports one GATT service, Service One. The Remote GATT Server supports two primary services, Service Two and Service Three.

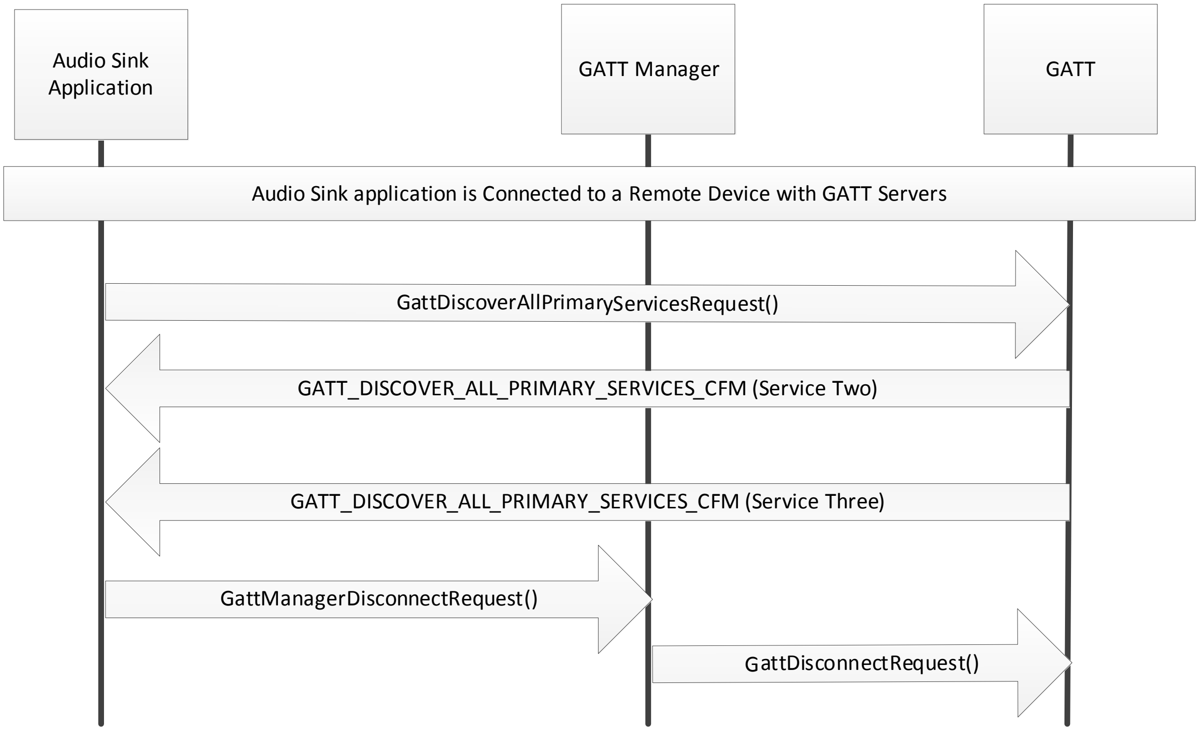
Audio Sink Application
GATT Manager
GATT
Audio Sink application is Connected to a Remote Device with GATT Servers
GattDiscoverAllPrimaryServicesRequest()
GATT_DISCOVER_ALL_PRIMARY_SERVICES_CFM (Service Two)
GATT_DISCOVER_ALL_PRIMARY_SERVICES_CFM (Service Three)
GattManagerDisconnectRequest()
GattDisconnectRequest()

**Figure 8-2 Failing to discover any supported GATT primary services**

**Description**

- The Audio Sink application discovers all primary services on the remote GATT Server.
- The Audio Sink application ignores the discovered primary services (Service Two and Service Three) that are not supported.
- When all primary services have been discovered and the Audio Sink application has not found any supported primary services, there is nothing more that can be done so the remote GATT Server is disconnected.

## 8.2 GATT client initiated procedures

When a connection has been made, the remote GATT Server services can be used by the Audio Sink application and client libraries. Typically a GATT Server offers characteristics that can be read or written. The GATT Manager VM library provides an API for the GATT Client initiated procedures:

- Read characteristic value
- Read using characteristic UUID
- Read long characteristic value

- Read multiple characteristic values
- Write without response
- Signed write without response
- Write characteristic value
- Write long characteristic value
- Reliable write prepare
- Reliable write execute
- Indication response

For more details on the API, refer to the GATT Manager library code documentation. Figure 8-3 describes an example GATT transaction for reading data from the remote GATT Server device. The message exchanges shown are typical for most GATT Manager APIs.

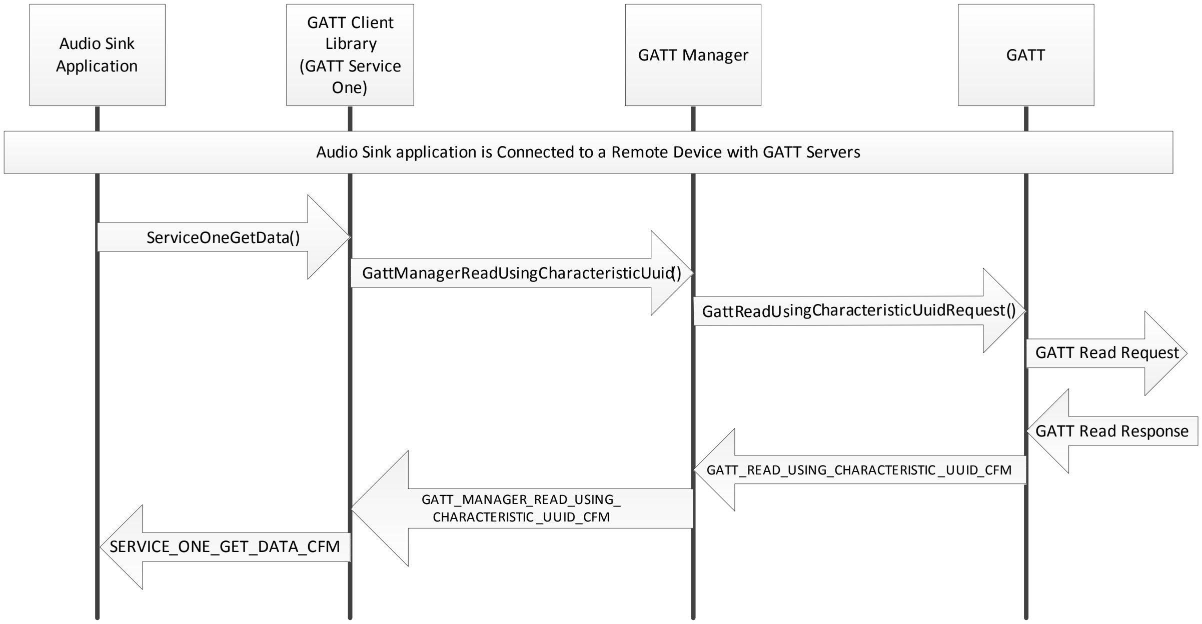
Audio Sink Application
GATT Client Library (GATT Service One)
GATT Manager
GATT
Audio Sink application is Connected to a Remote Device with GATT Servers
ServiceOneGetData()
GattManagerReadUsingCharacteristicUuid()
GattReadUsingCharacteristicUuidRequest()
GATT Read Request
GATT Read Response
GATT_READ_USING_CHARACTERISTIC_UUID_CFM
GATT_MANAGER_READ_USING_CHARACTERISTIC_UUID_CFM
SERVICE_ONE_GET_DATA_CFM

**Figure 8-3 Example GATT transaction for reading data**

# 9 GATT device disconnection

Disconnections with GATT devices can be issued by the Audio Sink application or the remote device.

## 9.1 Audio sink device issues disconnection

Figure 9-1 shows the message sequence when the Audio Sink application initiates a disconnection with a connected remote device.

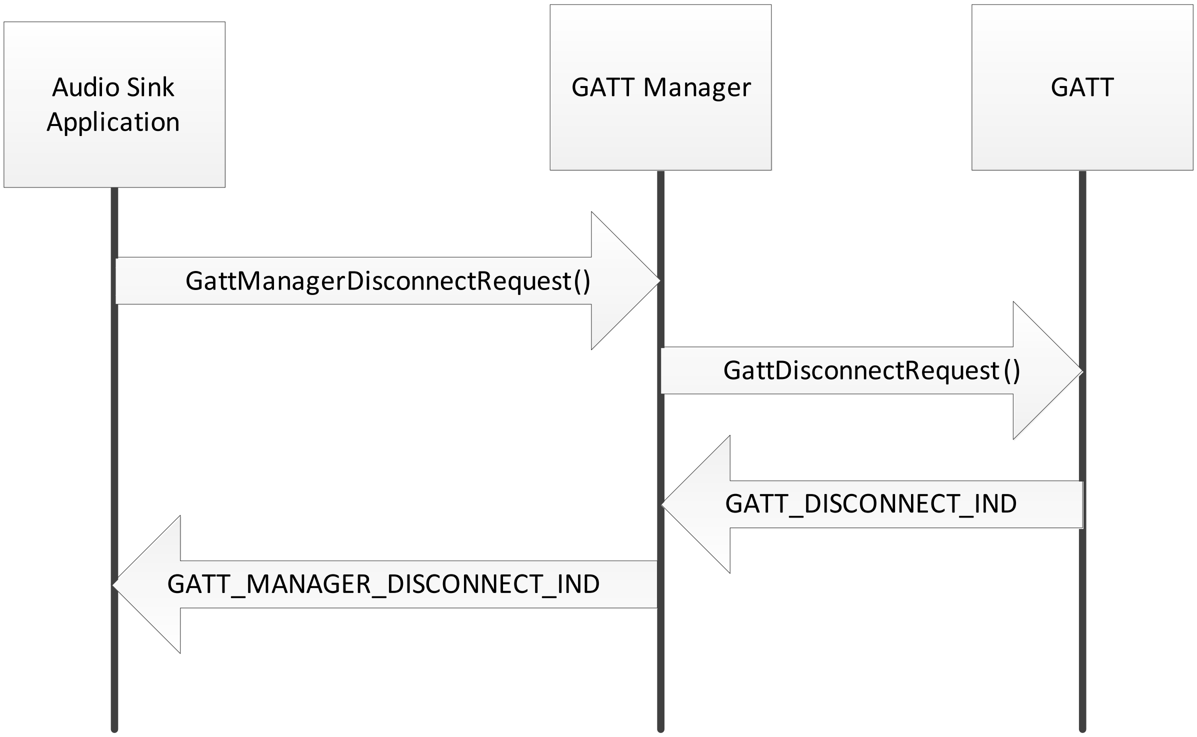

**Figure 9-1 Audio sink application disconnecting a remote device**

## 9.2 Remote device issues a GATT disconnection

Figure 9-2 shows the message sequence when the remote device initiates a disconnection with a connected Audio Sink application.

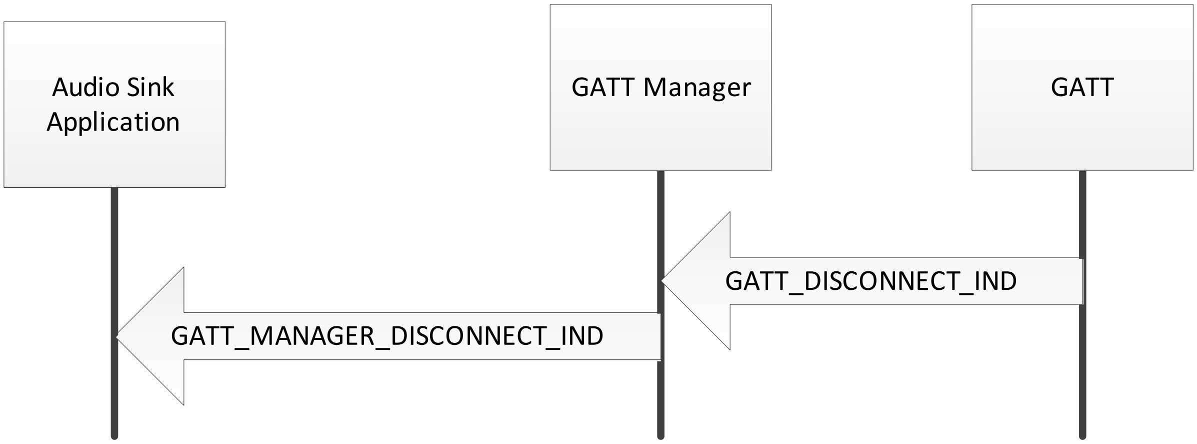

**Figure 9-2 Remote device disconnecting from the audio sink application**

## 9.3 Disconnection in GATT client role

While a remote device is connected to the Audio Sink application, GATT Client libraries may be initialized. When the connection has been terminated, these libraries are no longer required and memory used by them must be tidied up. The GATT Client libraries must also be unregistered from the GATT Manager. Figure 9-3 shows an example for tidying up the GATT Client library managing Service One of the remote device that has disconnected.

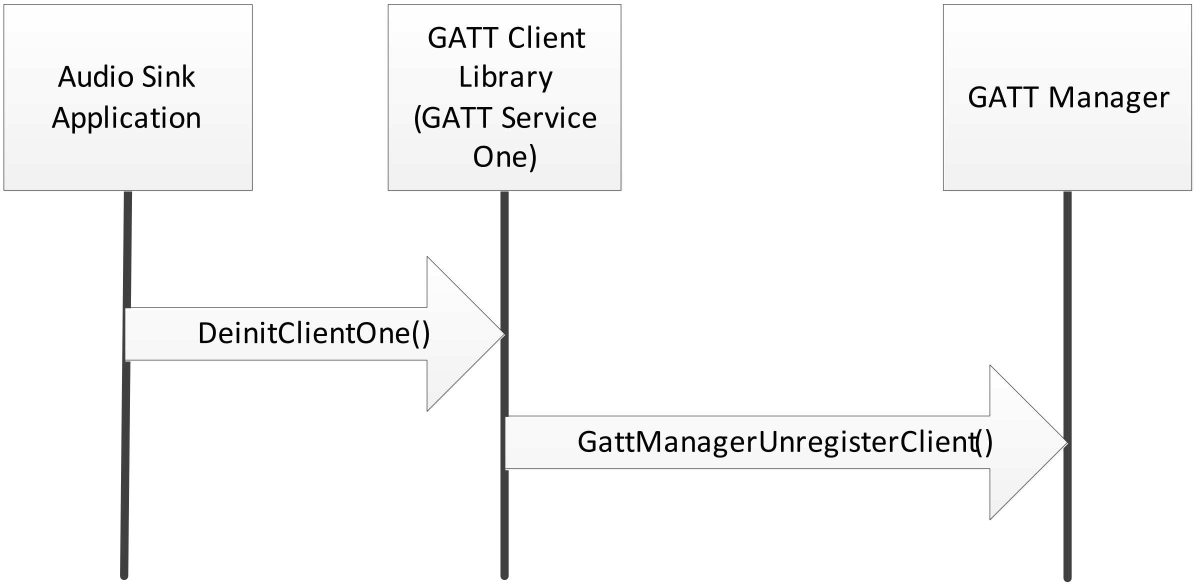
Audio Sink Application
GATT Client Library (GATT Service One)
GATT Manager
DeinitClientOne()
GattManagerUnregisterClient()

**Figure 9-3 Tidying up GATT client libraries after disconnection**

# Document references

| Document | Reference |
|---|---|
| *ADK Audio Sink Application User Guide* | 80-CT439-1/CS-00236868-UG |

# Terms and definitions

| Term | Definition |
|---|---|
| ADK | Audio Development Kit |
| API | Application Programming Interface |
| Bluetooth | Set of technologies providing audio and data transfer over short-range radio connections |
| GAIA | Generic Application Interface Architecture |
| GAP | Generic Access Profile |
| GATT | Generic Attribute Profile |
| HID | Human Interface Device |
| QTIL | Qualcomm technologies International, Ltd. |
| UUID | Universally Unique Identifier |
| VM | Virtual Machine |