Qualcomm Technologies International, Ltd.

Qualcomm GAIA Ecosystem

Reference

80-CT408-1 Rev. AR

October 19, 2017

Revision history

Revision	Date	Description
1	MAR 2011	Initial release. Alternative document number CS-00211724-DC.
2	MAR 2011	Product name update.
3	MAR 2011	Corrections to Configuration Commands descriptions.
4	APR 2011	Updated packet format
5	AUG 2011	Minor editorial changes.
6	OCT 2011	A note that a license key is required and should be tested on the production line was added to the Introduction.
7	JAN 2012	Updated to latest style guidelines.
8	FEB 2013	Updated for ADK 2.5.
9	DEC 2013	Corrections. GAIA v2.2 commands added. Updated to New CSR™ style.
10	MAY 2014	Remove requirement for license key
11	SEP 2016	Updated for ADK 4.1 GAIA enhancements. Updated to conform to QTI standards.
12	NOV 2016	Updated for ADK 6.0
13	DEC 2016	Issued for ADK 6.0
14	APR 2017	Editorial update for Vendor ID and copyright years.
AR	OCT 2017	Added to Content Management System. DRN updated to use Agile number. No technical changes.

Contents

Tables

Figures

1 Introduction to QTIL GAIA

QTIL Generic Application Interface Architecture (GAIA) implements an end-to-end, host-agnostic ecosystem supporting host application access to device functionality.

This document describes the low-level packet structure of the QTIL GAIA protocol and explains the concept of *notifications*, which contribute to the timely delivery of status information without energy-intensive polling. It also gives some examples of how a host application can use QTIL GAIA over RFCOMM to interact with a Bluetooth device.

2 QTIL GAIA wire protocol

The underlying protocol data unit is a packet composed of octets with framing to permit transmission over stream-based connections such as RFCOMM and RS-232. Numeric fields longer than 8 bits are packed with the most significant octet first. Textual strings are encoded using UTF-8.

In this document:

- The Host is the controlling party, for example, an application running on a smartphone.
- The Device is the controlled party, for example, a Bluetooth Headset.
- Commands may be sent from the Host to the Device or from the Device to the Host.

Where a connection needs to be established at a lower protocol level, for a Bluetooth RFCOMM for example, the QTIL GAIA protocol does not dictate which party is the initiator. That is the Host may be initiated as the RFCOMM Client or the Server.

2.1 QTIL GAIA command format

For packet-based transports (for example, connections with Bluetooth low energy devices using GATT).

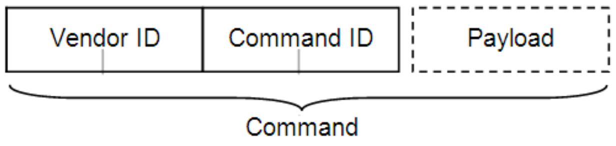


Figure 2-1 Command format

Description of Packet Fields:

- Vendor ID: This 16-bit field qualifies the command ID. All commands in this document have the Vendor ID assigned to QTIL by the Bluetooth SIG, `0x000a`.
- Command ID: This 16-bit field identifies the individual command.
- Payload: The payload contains any information required to be passed by a specific command. It consists of zero or more octets depending on the command.

2.2 QTIL GAIA framing

For stream-based transports, for example connections with Classic Bluetooth devices using RFCOMM.

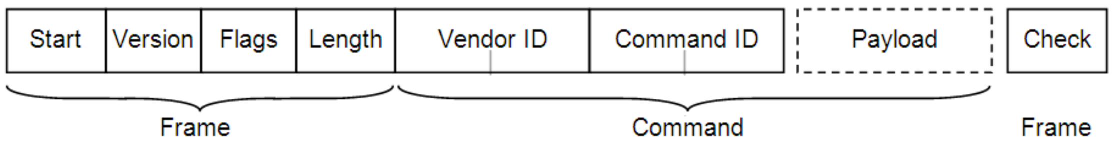
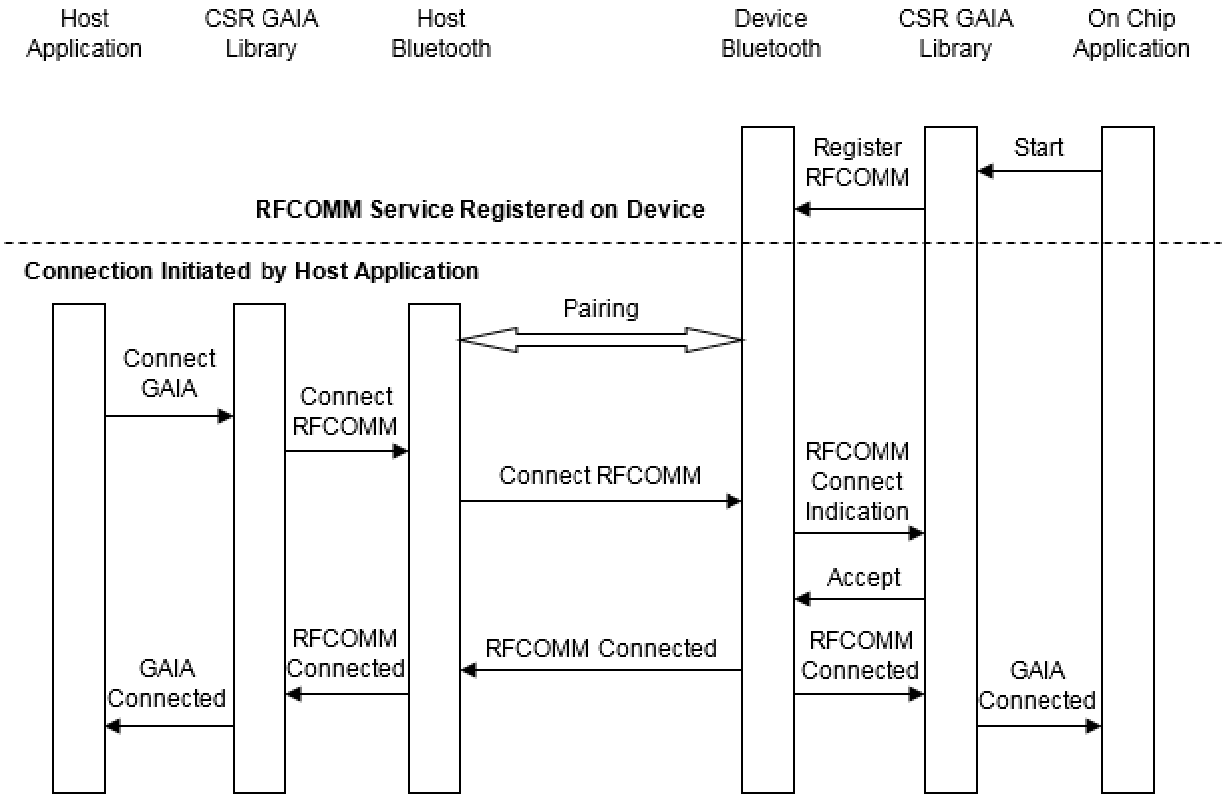


Figure 2-2 Packet framing

Description of Packet Fields:

- Start: One octet with the fixed value `0xff`.
- Version: One octet. This field indicates the protocol version in use, currently 1.
- Flags: One octet. Bits within this field control protocol options:
- Bit[0]: If set, a single octet check is present
- Bit[1:7]: Reserved, must be 0
- Vendor ID: This 16-bit field qualifies the command ID. All commands in this document have the Vendor ID assigned by the Bluetooth SIG, `0x000a`.
- Command ID: This 16-bit field identifies the individual command.
- Payload: The payload contains any information required to be passed by a specific command. It consists of zero or more octets depending on the command.
- Check: One octet. If present this field is determined by XORing together the other octets in the packet.

The start, length and check fields together enable the receiver to validate each packet. An acknowledgement (ACK) packet is sent in response to each valid command received. Packets with invalid check fields are silently ignored.



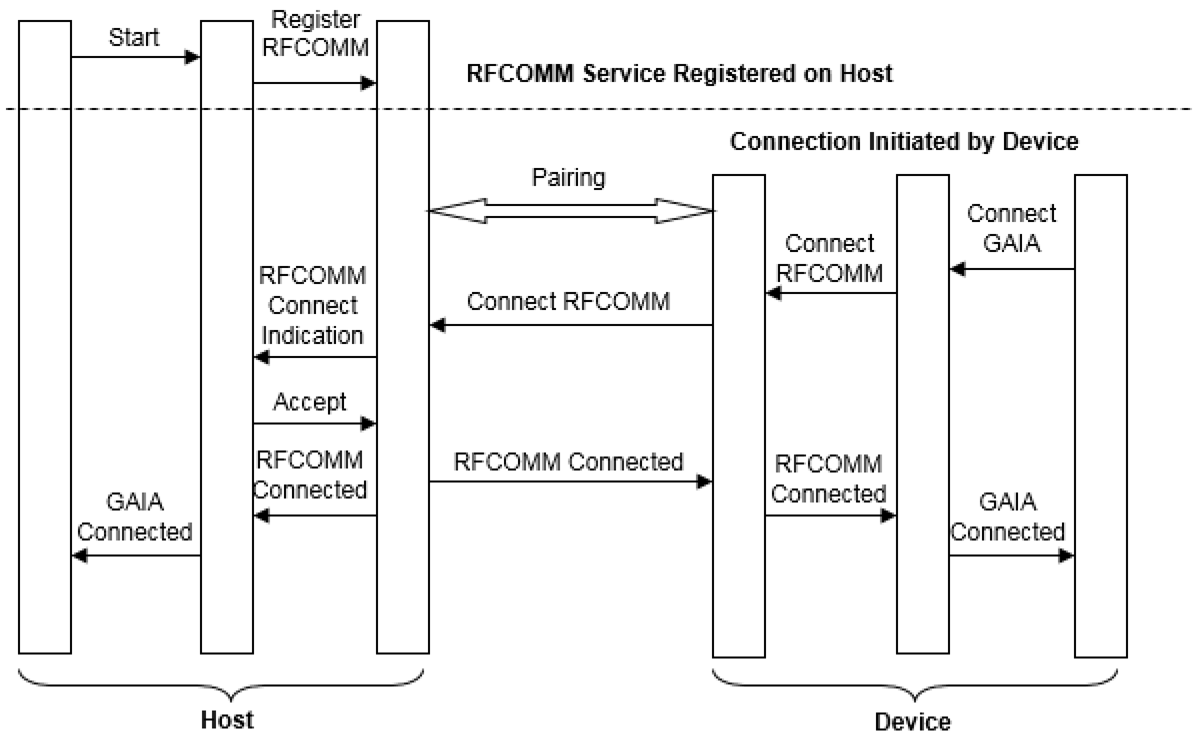


Figure 2-3 Establishing a connection over RFCOMM

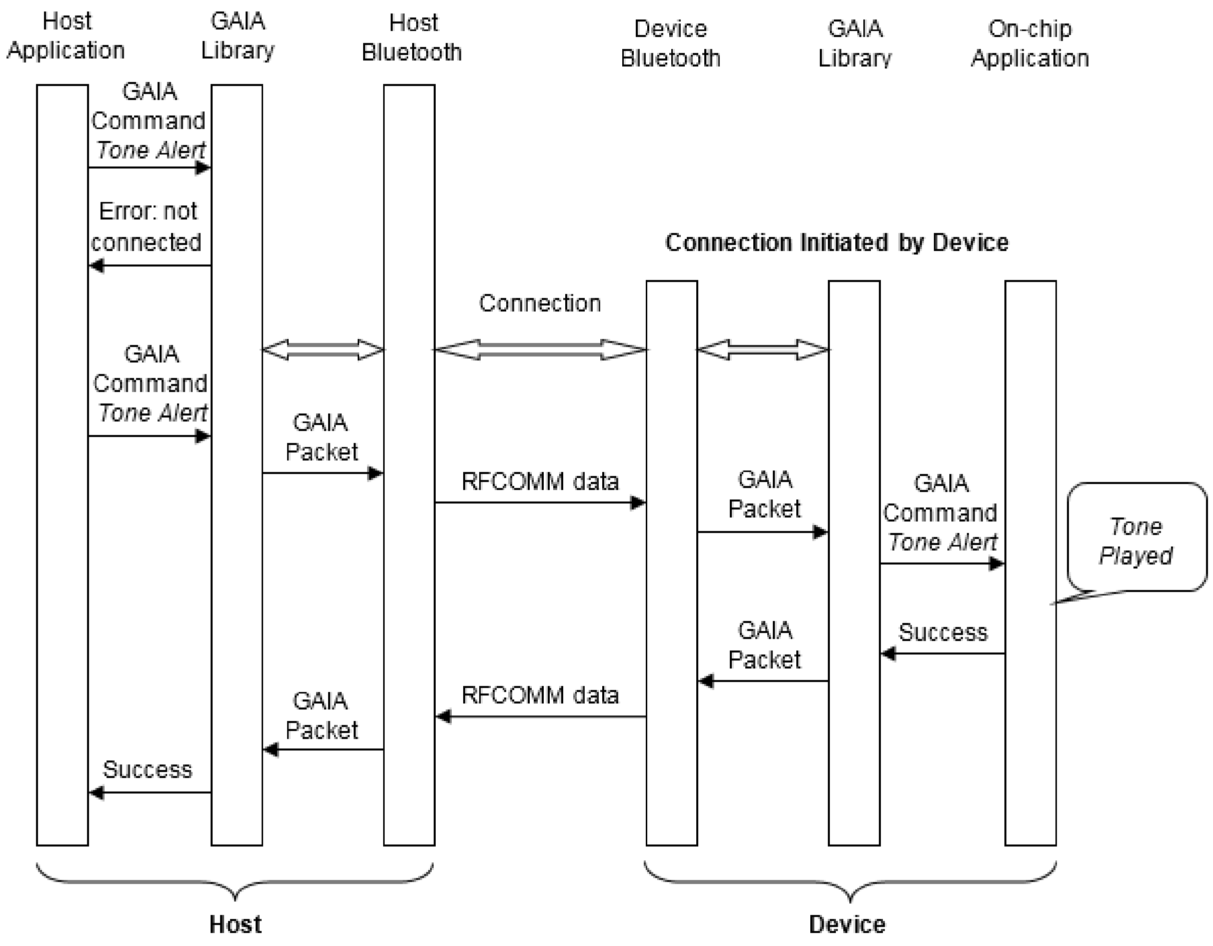


Figure 2-4 Command transmission

2.3 QTIL GAIA command and acknowledgement

Every command is acknowledged. The ACK packet has the same structure as the command packet with the value in the Command ID field being that of the initiating command with the top bit set. For example, the command `0x0001` would be acknowledged with a packet containing `0x8001` in the Command ID field.

Commands can be originated by either side of the connection.

The originator may send multiple commands without waiting for each to be acknowledged, subject to implementation and resource constraints. Thus multiple QTIL GAIA packets can be coalesced into a single packet at a lower level (for example, L2CAP, USB), reducing protocol overhead. The QTIL GAIA protocol does not specify that acknowledgements are received in order.

By convention, the first octet of the payload of an ACK packet holds a status value. see Appendix QTIL GAIA command status codes. This is not mandated by the protocol and vendors' own commands may behave differently.

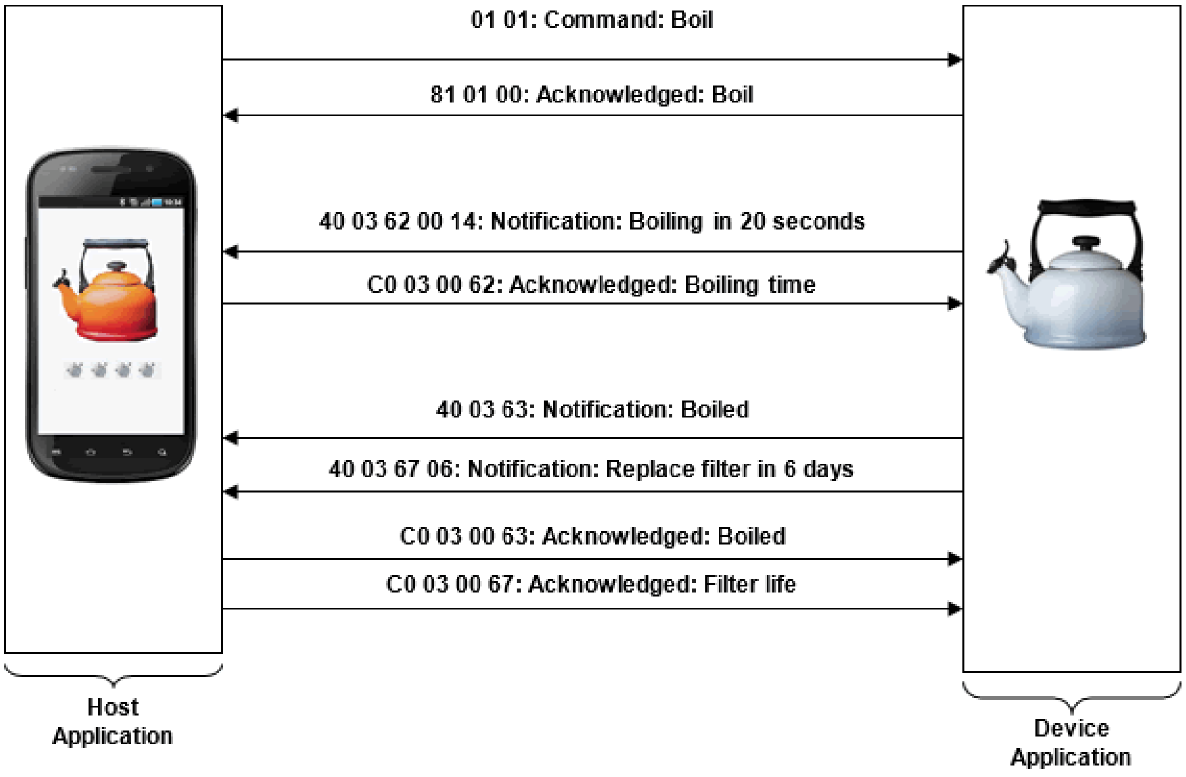

01 01: Command: Boil
81 01 00: Acknowledged: Boil
40 03 62 00 14: Notification: Boiling in 20 seconds
C0 03 00 62: Acknowledged: Boiling time
40 03 63: Notification: Boiled
40 03 67 06: Notification: Replace filter in 6 days
C0 03 00 63: Acknowledged: Boiled
C0 03 00 67: Acknowledged: Filter life
Host Application
Device Application

Figure 2-5 Command and acknowledgement sequence

3 QTIL GAIA commands

Commands may be handled internally by the QTIL GAIA library or passed to the application code for action.

Generic functions such as Read Battery Voltage are handled internally by library code. If the library does not recognize a command, it forwards the command to the device application code. Application-specific commands such as Increase Volume are handled in this way.

Commands handled by library code are always available. Commands handled by application code may not be available in all implementations.

The categories of QTIL GAIA commands are:

Configuration Commands	These modify the configuration of the device. It may be necessary to reset the device before these changes take effect. These commands do not automatically cause a reset so that several can be applied before a reset is carried out.
Control Commands	Control command requests are carried out immediately, with no need for a device reset.
Polled Status Commands	These request information about the status of the remote device.
Partition and File Commands	These commands allow read access to files within the file system. On devices that support external file systems, partitions can be updated and mounted
Debugging Commands	These enable an application developer to verify the operation of the protocol.
Notification Commands	The QTIL GAIA protocol supports notifications, that is, asynchronous indications of a change in device state. This eliminates the need for the host to continually poll the device for this information. This reduces radio traffic and increases battery life.

See the *ADK 4.3 Qualcomm GAIA Sink Command Reference* document for details of the commands available in the Sink application.

3.1 QTIL GAIA configuration commands

Command	Command ID	Default Handler	Description
Set LED Configuration	`0x0101`	Application	Determines patterns of LED indicators to show various states and events of the device. For example, a headset may show alternating red and blue LEDs to indicate that it is ready to accept connections from a phone
Get LED Configuration	`0x0181`	Application	
Set Tone Configuration	`0x0102`	Application	Configures indication tones generated by the device. For example, a headset could be configured to beep quietly every few seconds to remind the user that a call had been muted.
Get Tone Configuration	`0x0182`	Application	
Set Default Volume	`0x0103`	Application	Sets the initial volume settings for the device. The command payload consists of three octets with values from 0 to 15 representing the settings for indication tones, telephone speech and streamed music respectively.
Get Default Volume	`0x0183`	Application	
Factory Default Reset	`0x0104`	Application	Restores all settings to a selected default configuration.
Get Configuration ID	`0x0184`	Application	Gets the current default configuration ID.
Set Voice Prompt Configuration	`0x0106`	Application	Configures the device voice prompts to select a different language, voice, and so on.
Get Voice Prompt Configuration	`0x0186`	Application	
Set General Features	`0x0107`	Application	Configures the device behavior, corresponding to the Features tab of the Sink Configuration tool.
Get General Features	`0x0187`	Application	
Set User Event Configuration	`0x0108`	Application	Configures Events determining how the device reacts to button presses in different states. For example, pressing Volume up and Volume down on a headset during a call to mute the audio.
Get User Event Configuration	`0x0188`	Application	
Set Timer Configuration	`0x0109`	Application	Configures timeouts and intervals, such as the automatic switch-off time.
Get Timer Configuration	`0x0189`	Application	
Set Audio Gains	`0x010a`	Application	Determines the speaker and A2DP gain associated with each HFP volume level.
Get Audio Gains	`0x018a`	Application	
Set Power Configuration	`0x010c`	Application	Configures the battery and charger monitor and control system.
Get Power Configuration	`0x018c`	Application	
Set User-Defined Tone Configuration	`0x010e`	Application	Allows new tone patterns to be created to indicate states and events within the device.
Get User-Defined Tone Configuration	`0x018e`	Application	
Get Mounted Partitions	`0x01a0`	Application	Gets a bitmap of the currently mounted partitions containing voice prompts

3.2 QTIL GAIA control commands

Command	Command ID	Default Handler	Description
Change Volume	0x0201	Application	Increases or decreases the volume. The command payload is a single octet with a value of 0x00 (volume up) or 0x01 (volume down).
Device Reset	0x0202	Library	Causes the device to warm reset and load any changes made by the configuration commands.
Get Boot mode	0x0282	Library	Gets the current boot mode
Power Off	0x0204	Application	Powers off the device.
Set Volume Orientation	0x0205	Application	Swaps the orientation of the device volume control buttons. The command payload is a single octet with a value of 0x00 (normal) or 0x01 (inverted).
Get Volume Orientation	0x0285	Application	Gets the current orientation of the device volume control buttons. The acknowledgement payload carries two octets holding the command status (0x00 indicating success) and orientation, 0x00 (normal) or 0x01 (inverted).
Set LED Control	0x0207	Application	Enables or disables LED indicators on the device. Disabling the LEDs helps to conserve power. The command payload is a single octet with a value of 0x00 (disable) or 0x01 (enable).
Get LED Control	0x0287	Application	Gets whether the device LED indicators are enabled or disabled.
Set Voice Prompt Control	0x020a	Application	Turns on and off voice prompts from the device. The command payload is a single octet with a value of 0x00 (off) or 0x01 (on).
Get Voice Prompt Control	0x028a	Application	Gets whether device voice prompts are turned on or off. The acknowledgement payload carries two octets holding the command status (0x00 indicating success) and state, 0x00 (off) or 0x01 (on).
Change Text-to-Speech Language	0x020b	Application	Selects the next available language for Text-to-Speech functions.
Set Speech Recognition Control	0x020c	Application	Turns simple speech recognition, on the device, on and off. The command payload is a single octet with a value of 0x00 (off) or 0x01 (on).
Get Speech Recognition Control	0x028c	Application	Gets whether the device speech recognition is turned on or off. The acknowledgement payload carries two octets holding the command status (0x00 indicating success) and state, 0x00 (off) or 0x01 (on).
Alert LEDs	0x020d	Application	Displays a pattern on the device LED indicators.
Alert Tone	0x020e	Application	Plays a sequence of audio tones on the device.

Command	Command ID	Default Handler	Description
Alert Event	`0x0210`	Application	Plays a sequence of LEDs and audio tones on the device as configured with the associated event.
Alert Voice	`0x0211`	Application	Plays the indicated voice prompt
Set TTS Language	`0x0212`	Application	Sets the TTS Language for the device
Get TTS Language	`0x0292`	Application	Gets the current TTS Language
Set Bass Boost/Plus	`0x0215`	Application	Sets bass boost/plus on or off
Get Bass Boost/Plus	`0x0295`	Application	Gets the current bass boost enable state
Set 3D Enhancement/virtualization	`0x0216`	Application	Sets the 3D enhancement/virtualization on or off
Get 3D Enhancement/virtualization	`0x0296`	Application	Gets the current 3D enhancement/virtualization enable state
Switch EQ Control	`0x0217`	Application	Selects the next available equalizer bank
Toggle Bass Boost/Plus	`0x0218`	Application	Toggles the bass boost/plus feature on/off
Toggle 3D Enhancement/virtualization	`0x0219`	Application	Toggles the 3D enhancement/virtualization feature on/off
Set EQ Parameter	`0x021a`	Application	Sets a parameter of the parametric equalizer
Get EQ Parameter	`0x029a`	Application	Gets a parameter of the parametric equalizer
Set EQ Group Parameter	`0x021b`	Application	Sets a group of parameters of the parametric equalizer
Get EQ Group Parameter	`0x029b`	Application	Gets a group of parameters of the parametric equalizer
Enter Pairing Mode	`0x021d`	Application	Puts the device into Bluetooth Pairing mode

3.3 QTIL GAIA debug commands

Command	Command ID	Default Handler	Description
No Operation	`0x0700`	Library	Requests the device to perform no operation (other than to send an acknowledgement packet). This serves to establish that the QTIL GAIA protocol handler is running.

3.4 QTIL GAIA polled status commands

Command	Command ID	Default Handler	Description
Get API Version	`0x0300`	Library	Gets the highest supported protocol version and the on-chip API version. The returned payload contains the command status and, if successful, three further octets representing the protocol version, the API major revision and API minor revision.
Get Current RSSI	`0x0301`	Library	Gets the Received Signal Strength Indication reported by the remote device. The returned payload contains the command status and, if successful, one further octet representing the signal strength in decibel milliwatts in two's complement form.
Get Current Battery Level	`0x0302`	Library	Gets the battery level reported by the remote device. The returned payload contains the command status and, if successful, two further octets representing the battery level in millivolts.
Get Module ID	`0x0303`	Library	Gets module identification information from the remote device. The returned payload contains the 16-bit Hardware ID, the 16-bit Design ID and the 32-bit Module ID.
Get Application Version	`0x0304`	Library	Gets the application software to identify itself. The acknowledgement payload contains an eight-octet application version identifier optionally followed by null-terminated human-readable text.
Get PIO State	`0x0306`	Application	Gets the state of the device digital inputs as read by the application
Read ADC	`0x0307`	Application	Gets the state of a specified analog input as read by the application

3.5 QTIL GAIA partition and file commands

Command	Command ID	Default Handler	Description
Get Storage Partition status	`0x0610`	Library	Gets the status of the specified partition
Open Storage Partition	`0x0611`	Library	Opens a partition ready for write
Write Storage Partition	`0x0615`	Library	Writes data to the previously opened partition
Close Storage Partition	`0x0618`	Library	Closes the previously opened partition
Mount Storage Partition	`0x061a`	Library	Mounts a specified partition into the union file system
Open File	`0x0621`	Library	Opens a file in the file system

Command	Command ID	Default Handler	Description
Read File	`0x0624`	Library	Reads data from the opened file
Close File	`0x0628`	Library	Closes the previously opened file
DFU Request	`0x0630`	Host	Requests a Device Firmware Upgrade from the Host
DFU Begin	`0x0631`	Library	Begins the Device Firmware Upgrade process

3.6 QTIL GAIA notification commands

Command	Command ID	Default Handler	Description
Register Notification	`0x4001`	Library	Establishes that the host is to be informed when a specified event occurs on the device. For example, that the battery voltage has fallen to a preset level. The command payload may contain event-specific data, for example one or more thresholds at which notification is to occur.
Get Notification	`0x4081`	Library	Gets details of the established notifications of a given type. The acknowledgement payload depends on the event type
Cancel Notification	`0x4002`	Library	Cancels the notification of a specified event.
Event Notification	`0x4003`	Host	Sent by the device whenever the criterion established in a Register Notification command is met.

3.7 QTIL GAIA example exchanges

Examples below show QTIL GAIA packets in hexadecimal with colons separating the header, payload and checksum components.

No operation

This is the simplest complete QTIL GAIA protocol exchange. The command has no payload and the response is a simple acknowledgement of success. This example uses the optional packet checksum feature. To request a no operation the Host sends:

```
ff 01 01 00 00 0a 07 00 : : f2
```

Description of the Host request:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x01`: Flags: GAIA_FLAG_CHECK
- `0x00`: Payload length (0)
- `0x000a`: Vendor ID (CSR)
- `0x0700`: Command ID (No Operation)
- `: :`: (no payload)
- `0xf2`: Checksum (`0xff` ⊕ `0x01` ⊕ `0x01` ⊕ `0x00` ⊕ `0x00` ⊕ `0x0a` ⊕ `0x07` ⊕ `0x00`)

The expected response is:

```
ff 01 01 01 00 0a 87 00 : 00 : 73
```

Description of the expected response:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x01`: Flags: GAIA_FLAG_CHECK
- `0x01`: Payload length (1)
- `0x000a`: Vendor ID (CSR)
- `0x8700`: Acknowledged Command ID (No Operation)
- `0x00`: Status (success)
- `0x73`: Checksum (`0xff` ⊕ `0x01` ⊕ `0x01` ⊕ `0x01` ⊕ `0x00` ⊕ `0x0a` ⊕ `0x87` ⊕ `0x00` ⊕ `0x00`)

Get application version

This example does not use the packet checksum feature. To get the application version the Host sends:

```
ff 01 00 00 00 0a 03 04 : :
```

Description of the Host request:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x00`: Flags: none
- `0x00`: Payload length (0)
- `0x000a`: Vendor ID (CSR)
- `0x0304`: Command ID (Get Application Version)
- `: :`: (no payload)

A typical response would be:

```
ff 01 00 26 00 0a 83 04 :
00 01 00 00 4e 36 02 03 2c 4e 65 61 73 64 65 6e
20 36 2d 56 61 6c 76 65 20 48 65 61 64 73 65 74
20 56 32 2e 33 00 :
```

Description of example response:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x00`: Flags: none
- `0x26`: Payload length (38)
- `0x000a`: Vendor ID (CSR)
- `0x8304`: Acknowledged Command ID (Get Application Version)
- `0x00`: Status (success)
- `0x0100004e3602032c`: Software version identifier
- `0x4e656173...` The string "Neasden 6-Valve Headset V2.3"

Get current RSSI

To get the received signal strength indication from the device, the Host application sends:

```
ff 01 00 00 00 0a 03 01 : :
```

Description of the Host request:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x00`: Flags: none
- `0x00`: Payload length (0)
- `0x000a`: Vendor ID (CSR)
- `0x0301`: Command ID (Get Current RSSI)

A typical response would be:

```
ff 01 00 02 00 0a 83 01 : 00 ec :
```

Description of example response:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x00`: Flags: none
- `0x02`: Payload length (2)
- `0x000a`: Vendor ID (CSR)
- `0x8301`: Acknowledged Command ID
- `0x00`: Status (success)
- `0xec`: RSSI value (-20dBm)

Failed attempt to set default volumes

This example describes a request to set the default volumes which includes an invalid value. For the Sink application, the valid range of the speech volume is 0 to 15 (`0x00` to `0x0f`). The value 20 (`0x14`) is illegal.

The Host sends:

```
ff 01 00 03 00 0a 01 03 : 00 14 0f :
```

Description of the Host request:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x00`: Flags: none
- `0x03`: Payload length (3)
- `0x000a`: Vendor ID (CSR)
- `0x0103`: Command ID (Configure Default Volumes)
- `0x00`: Set default tone volume to 0
- `0x14`: Set default speech volume to 20 (illegal)
- `0x0f`: Set default music volume to 15

The device responds with

```
ff 01 00 01 00 0a 81 03 : 05 :
```

Description of response:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x00`: Flags: none
- `0x01`: Payload length (1)
- `0x000a`: Vendor ID (CSR)

- `0x8103`: Acknowledged Command ID
- `0x05`: Status (failure: invalid parameter)

4 QTIL GAIA notifications

Table 4-1 lists the events for which notifications may be registered in the current QTIL GAIA architecture.

Table 4-1 Notification event codes

Event	Code	Description
RSSI Low Threshold	`0x01`	Occurs whenever the Received Signal Strength Indication falls to or below a preset threshold. One or two thresholds may be set, each expressed as a single octet representing a signal level in decibel milliwatts in two's complement form.
RSSI High Threshold	`0x02`	Occurs whenever the Received Signal Strength Indication rises to or above a preset threshold. One or two thresholds may be set, each expressed as a single octet representing a signal level in decibel milliwatts in two's complement form.
Battery Low Threshold	`0x03`	Occurs when the measured battery voltage falls to or below a preset threshold. One or two thresholds may be set, each represented by an unsigned 16-bit number of millivolts.
Battery High Threshold	`0x04`	Occurs when the measured battery voltage rises to or above a preset threshold. One or two thresholds may be set, each represented by an unsigned 16-bit number of millivolts.
PIO Changed	`0x06`	Occurs when one of the digital inputs to the Qualcomm® BlueCore™ technology hardware changes, for example as a result of a button being pressed or a signal from another component of the device. The interpretation of a PIO event is dependent on the application.
Battery Charged	`0x08`	Occurs when the device hardware detects that the battery charging is complete.
Charger Connection	`0x09`	Occurs when the device hardware detects that the battery charger has become connected or disconnected
CapacitiveSensor Update	`0x0a`	Occurs when one of the capacitive sensors on the device detects a change.
User Action	`0x0b`	Occurs when a user action (for example a long button press) is detected.
Speech Recognition	`0x0c`	Occurs when the Speech Recognition engine detects an input
DFU State	`0x10`	Occurs when the Device Firmware Upgrade process enters a new state

4.1 QTIL GAIA example exchanges

QTIL GAIA packets in the following examples are shown in hexadecimal with colons separating the header, payload and checksum components.

To request low battery notification

To request a notification when the battery voltage drops to 3.6 V and to 3.5 V, the host sends a Register Notification command, for example.:

```
ff 01 00 05 00 0a 40 01 : 03 0e 10 0d ac :
```

Description of example request:

- `0xff`: Start of frame
- `0x01`: QTIL GAIA Protocol version 1
- `0x00`: Flags: none
- `0x05`: Payload length (5)
- `0x000a`: Vendor ID (CSR)
- `0x4001`: Command ID (Register Notification)
- `0x03`: Event type (Battery Low Threshold)
- `0x0e10`: Battery level (3600 mV)
- `0x0dac`: Battery level (3500 mV)

Typical response:

```
ff 01 00 02 00 0a c0 01 : 00 03 : 35
```

The payload indicates success (`0x00`) registering a Battery Low Threshold event (`0x03`).

Any time the battery voltage is measured at or below either of the configured thresholds, the device sends a notification.

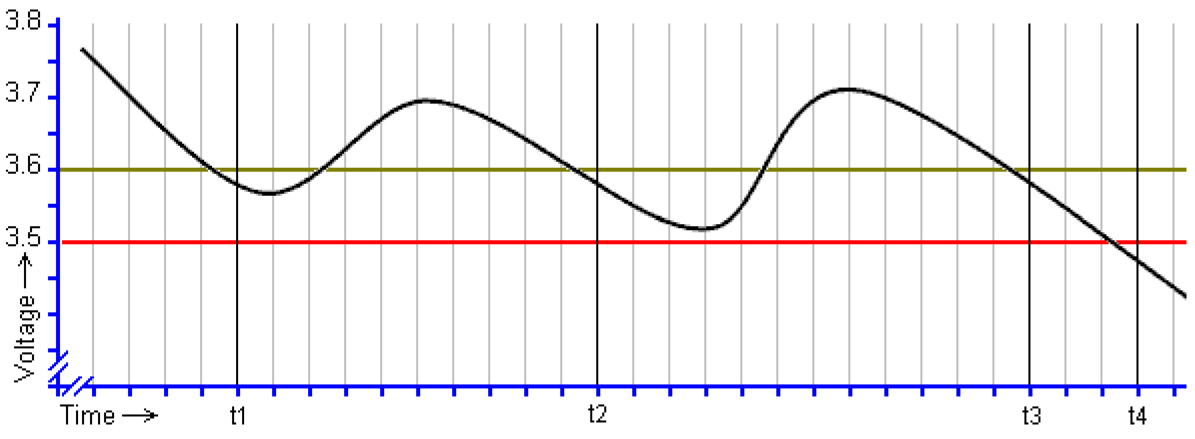


Figure 4-1 Battery threshold notification

The device periodically samples its battery voltage and at time t1 sends:

```
ff 01 00 03 00 0a 40 03 : 03 0d f1 :
```

This indicates that event `0x03` (Battery Low Threshold) has occurred at level `0x0df1` (3569 mV).

The Host should acknowledge receipt of the notification by sending:

```
ff 01 00 02 00 0a c0 03 : 00 03 :
```

The device sends further notifications at times t2, t3 and t4 as the voltage is sampled below the set thresholds.

To cancel low battery notification

To cancel low battery notifications the Host sends:

```
ff 01 00 01 00 0a 40 02 : 03 :
```

This cancels notification (`0x4002`) of Battery Low Threshold events (`0x03`).

The device acknowledges this with:

```
ff 01 00 02 00 0a c0 02 : 00 03 :
```

This indicates success (`0x00`) canceling event `0x03`.

To request RSSI low threshold notification

To set up notifications of the Received Signal Strength Indication falling to -50 dBm, the Host sends, for example:

```
ff 01 00 02 00 0a 40 01 : 01 ce :
```

This requests notification of event `0x01` (RSSI Low Threshold) at level `0xce` (-50 dBm). The device acknowledges the request with:

```
ff 01 00 02 00 0a c0 01 : 00 01 :
```

If the RSSI is later measured at, for example, -54 dBm then the device sends:

```
ff 01 00 02 00 0a 40 03 : 01 ca :
```

This indicates the event `0x01` (RSSI Low Threshold) has occurred at level `0xca` (-54 dBm).

The Host acknowledges the notification by sending:

```
ff 01 00 02 00 0a c0 03 : 00 01
```

This indicates success (`0x00`) processing of the RSSI Low Threshold event (`0x01`).

5 QTIL GAIA vendor extensions

The **Vendor Id** field in the QTIL GAIA packet allows new commands to be added without conflicting with existing or future QTIL-defined commands or with those of another vendor.

When the device receives a correctly framed QTIL GAIA packet that has a Vendor Id other than QTIL's, the packet is forwarded to vendor's own code on the device. The vendor's code should then act on the command and return a result in QTIL GAIA format.

NOTE QTIL Software Development Kits provide the tools to write vendor-specific code including libraries of functions to create and exchange properly framed QTIL GAIA packets.

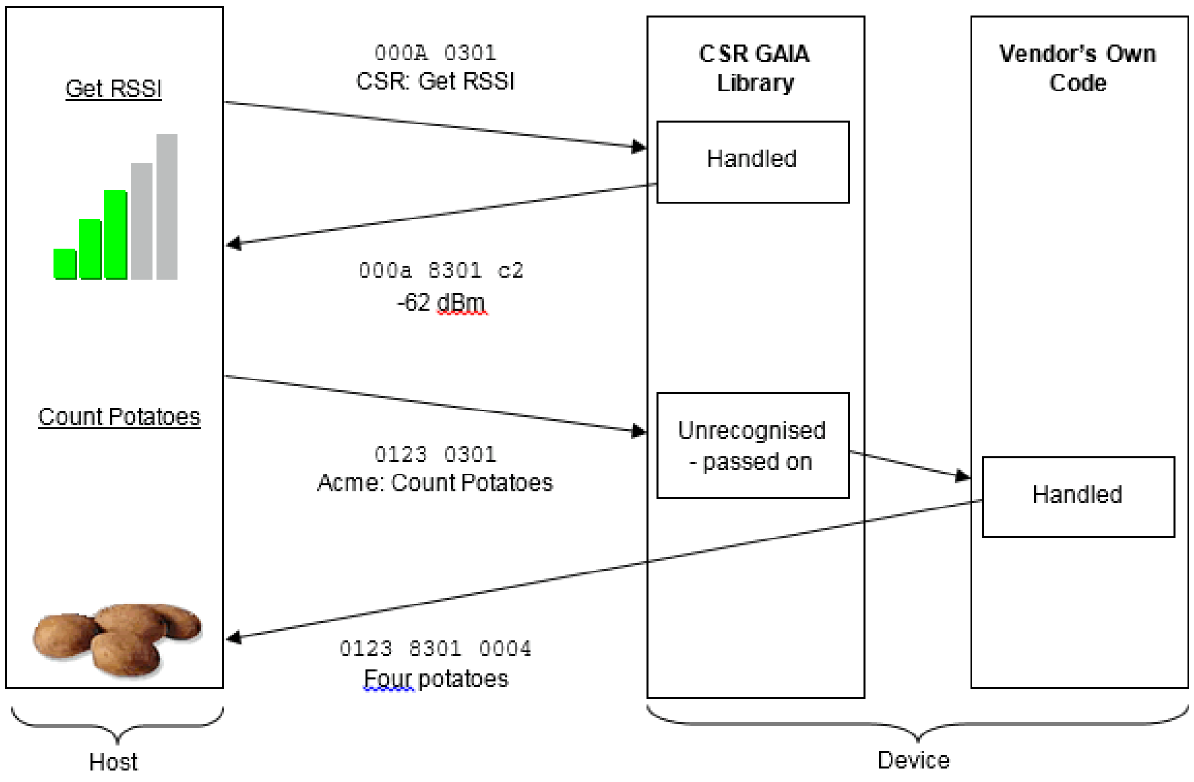


Figure 5-1 Command forwarding

A QTIL GAIA command status codes

By convention, the first octet in an acknowledgement (ACK) packet is a status code indicating the success or the reason for the failure of a command. The table below lists the status codes that are currently defined.

Table A-1 GAIA command status codes

Status	Code	Description
Success	`0x00`	The command completed successfully
Failed: Command Not Supported	`0x01`	An invalid Command ID was specified
Failed: Insufficient Resources	`0x03`	The command was valid but the device could not complete it successfully
Failed: Invalid Parameter	`0x05`	An invalid parameter was used in the command
Failed: Incorrect State	`0x06`	The device is not in the correct state to process the command.
In Progress	`0x07`	The command is in progress (1).
(1) `GAIA_STATUS_IN_PROGRESS` (`0x07`) is the value of the Command Status characteristic until the operation completes, for use by Bluetooth Smart hosts that do not support GATT Notifications but need to check that a command has succeeded.		

Document references

Document	Reference
Company Identifiers page	www.bluetooth.org
ADK 4.3 Qualcomm GAIA Sink Command	80-CF422-1 /CS-00406808-DC

Terms and definitions

Term	Definition
ACK	Acknowledgement
ADC	Analog to Digital Converter
ADK	Audio or Application Development Kit
API	Application Programming Interface
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
Bluetooth SIG	The Bluetooth Special Interest Group oversees the development of Bluetooth standards and the licensing of Bluetooth technologies and trademarks to manufacturers.
DFU	Device Firmware Upgrade
GAIA	Generic Application Interface Architecture
ID	Identifier
LED	Light-Emitting Diode
PIO	Programmable Input/Output
PS	Persistent Store
QTIL	Qualcomm Technologies International, Ltd.
RSSI	Received Signal Strength Indication
SDP	Service Discovery Protocol
SIG	Special Interest Group
SPP	Serial Port Profile
SQIF	Serial Quad I/O Flash, a nonvolatile memory technology
TTS	Text-to-Speech
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
USB-IF	The USB Implementers' Forum is responsible for issuing USB vendor IDs to product manufacturers.