HM Headphones Protocols_General v2.0

Author	Bright.Sun, Beet.Li, Kevin.Zheng, Gavin Gan	
Auditor	William Luk, Marc Richarme, Pheobe Lang, Alex Yu, John Xu, John Qin	
Location	Shenzhen	
Telephones	+86 755 86682353	
Released by	**Signature**	**Date**
Accepted by	**Signature**	**Date**

Document History

Version	Date	Changes	Changes Done By
1.0	6/29/2018	Draft version	**App team**
1.1	7/2/2018	Revised	**William Luk**
1.2	7/2/2018	Specific Adjust for Project BES AKG N200NC	**Beet Li**
1.3	7/11/2018	Add device status partion	**Beet Li**
1.4	13/7/2018	Support EQ settings	**Beet Li**
1.5	3/8/2018	Correcting some mistakes & support Firmware verison	**Beet Li**
1.6	20/8/2018	Adding 3.5 tp define EQ part	**Beet Li**
1.7	21/8/2018	EQ detail improve	**Beet Li**
1.8	3/9/2018	EQ data transfer improve	**Beet Li**
1.9	17/9/2018	Add Multi-AI cmd, improve protocal details	**Beet Li**
1.9.2	26/12/2018	Add BT Status, add OTA upgrade Status	**Beet Li**
1.9.3	2/1/2019	Add color id for recognize the DUT model	**Beet Li**
1.9.4	2/21/2019	Add double CRCs check	**Beet Li**
1.9.5	5/7/2019	Add new command “auto play/pause” enable ADV Manufacturer Data Movement change	**Beet Li**
2.0.0	4/7/2019	1. Add new more color ID. 2. Add new CMD “Find my buds” 3. Add new DUT Status for TWS “TWS connection status” 4. Add new EQ category “Personi-Fi” EQ. (Still in Pending)	**Beet Li**
2.0.1	16/7/2019	1. confirm new EQ category Personi-Fi EQ. and Personi-Fi EQ	**Beet Li**
2.0.2	23/7/2019	1. Remove DJ EQ Category. 2. Add new cmd - setPersoniFiPresets 3. Add new cmd – reqPersoinFiPresests. 4. Add new cmd – retPersoniFiPresets.	**Beet Li**
2.0.3	5/8/2019	1. Add new command for request find my buds status	**Beet Li**
2.0.4	30/8/2019	2. Add customize touch panel control support	**Beet Li**
2.0.5	19/11/2019	1. Add new command for support noise canceling gain control	**Beet Li**
2.0.6	01/06/2020	Add more gestures and funcs for touch panel	**Beet Li**
2.0.7	Mar-2-2020	Add HP Analytics info data support	**Beet Li**
2.0.8	Mar-19-2020	Add xiaowei ai setting	**Beet Li**

2.0.9	Jun-10-2020	Add Personi-Fi Hearing Test enable/disable Add change volume for Personi-Fi Hearing Test	**Gavin Gan**
2.0.10	Jun-28-2020	Update command TalkThru/AbientAware Update command AutoOff	**Gavin Gan**

CONTENTS

1. Scope

This document describes the app to work with BES headphones.

2. Transport

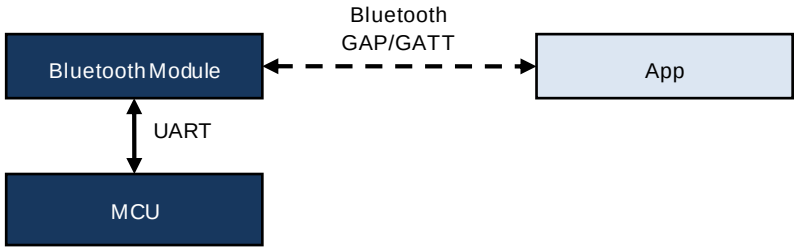

Figure 2-1 App-Dev Transport

App will discovery device Bluetooth GAP profile, see **Section 2.1** for details. There will be 3 different data transports to run this protocol:

- GATT (BLE)

The AKG BES Products adopt this documents shall supports BLE services list in **Table 2-1**.

Primary Service	Service UUID
GAP	00001800-0000-1000-8000-00805f9b34fb
GATT	00001801-0000-1000-8000-00805f9b34fb
BLE_RX_TX	65786365-6c70-6f69-6e74-2e636f6d0000

Table 2-1 Bluetooth LE Services(0xfddf Harman International)

GAP service is used for device discovery. `BLE_RX_TX` service is used for transfer data packets between App and device. Device require to broadcast the advertisement data with attaching HARMAN UUID Service (0xFDDF)as 16 bit format.

2.1 Device Discovery

App shall try to discovery device via GAP profile.When App discovered a peripheral with the GAP profile, it can read device manufacture data in **Table 2-2**.

Data Type	Description
Manufacturer Specific Data	A unique ID to identify device and its role.
Broadcaster/Receiver Data	**4-6bytes of data is only for the devices not intended for APP.** Name type can be changed by device depending on this data

Table 2-2 GAP Manufacture

Data Type	Size	Description
PID	2 bytes	Product ID
Color ID	1 bytes	Transformed by Model ID 0: Black/ 1: White / 2: Blue / 3: Red / 4: Green 5: Purple / 6: GoldSilver / 7: Silver. / 8: Grey / 9: Beige
SrcName1	2 bytes	A CRC16 value of source device name 1.
SrcName2	2 bytes	A CRC16 value of source device name. 2

Table 2-3 Manufacturer Data Format

Firstly, the App shall check if the `SrcName` (CRC16 value) is identical as it-self. Only if they are identified, App will be allowed to connect target device. The same CRC16 algorithm in Linux Kernel has used for calculate this field. The polynomial is 0x8005 (x^16 + x^15 + x^2 + 1). Refer to:

https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/tree/include/linux/crc16.h

https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/tree/lib/crc16.c

Secondly, this manufacturer data should be moved behind the advertisement service "FDDF" as the advertisement service data Of "FDDF"

2.2 Data Transfer

App and speaker device uses `BLE_RX_TX` service to talk with each other via private protocols defined in follow sections. There are 2 characteristics in the service that used for send and receive data packet.

Characteristic	UUID	Access	Size
RX_CHAR	0x6578 6365 6c70 6f69 6e74 2e63 6f6d 0001	Read/Notify	Max 60 bytes
TX_CHAR	0x6578 6365 6c70 6f69 6e74 2e63 6f6d 0002	Write	Max 60 bytes

Table 2-4 GATT characteristics for packet transfer

Device will send notification to App with packet bytes in `RX_CHAR` characteristic. One notification always carries only one data packet. App shall register the notification once the connection established.

App can also write packets to `TX_CHAR` characteristic. A packet shall write to the characteristic by single write operation.

3. Packet Format

The general packet format defines in **Table 3-1**.

Section	Field	Size	Description
Header	Identifier	1 byte	Always 0xAA.
	CmdID	4 bits (MSB)	Command ID.
	SubCmdID	4 bits (LSB)	Sub command ID.
	PayloadLen	1 byte	Payload length (0 ~ 0x3C)
Payload	Actually data packet.		

Table 3-1 Package format

The field `PayloadLen` defines packet length without headers. Therefore the total package length equals `PayloadLen` + 3.

3.1 Command Summary

Category	CmdID	Name	Sub-CmdID	R/T	ACK	Description
General	0x0	DevACK	0x0	RX	No	Some commands need ACK.
		AppACK	0x1	TX	No	Some commands need App ACK.
		DevByeBye	0x2	RX	AppACK	Device want disconnect.
		AppByeBye	0x3	TX	DevACK	App want disconnect.
		DevFinAck	0x4	RX	No	device finish sending data
		AppFinAck	0x5	TX	No	App finish writing data
Device Info	0x1	ReqDevInfo	0x1	TX	RetDevInfo	App request device information.
		RetDevInfo	0x2	RX	No	Device report information to App.
		ReqDevAnalyticsInfo	0x3	Tx	RetDevAnalyticsInfo	Refer to 3.10
		RetDevAnalyticsInfo	0x4	Rx	No	Refer to 3.10

		ReqAnalyticsInfoClear	0x5	Tx	DevAck	Refer to 3.10
Device Status	0x2	ReqDevStatus	0x1	TX	RetDevStutas	App request device information. **Refer to 3.4**
		RetDevStatus	0x2	RX	No	Device report information to App. **Refer to 3.4**
		ReqFindMyBuds	0x3	TX	RetFindMyBuds	Refer to 3.4.3
		RetFindMyBuds	0x4	Rx	No	Refer to 3.4.3
Remote Control	0x3	ANC	0x1	R/T	DevACK	Write ANC enable status. Payload: 0x00/0x01 means OFF/ON
		Ambient Aware Mode and TalkThru	0x2	R/T	DevACK	Write TalkThru/AbientAware Mode status. 0b0000000X: TalkThru status 0b000000Y0: AbientAware status 0 – off 0b00000001 – TalkThru 0b00000010 – AbientAware 0b00000011 – TalkThru & AbientAware
		Auto Off	0x3	R/T	DevACK	Write AutoOff status, 2 options of payload, the 2nd option is preferred in new device: 1. Payload length 1 byte: 1 bit(MSB): 0/1 means disable/enable 7 bits(LSB): auto off time value /mins; 2. Payload length 2 bytes: auto off time value in minutes, 0 means disabled.
		Multi AI Button	0x4	R/T	DevAck	**0/1/2/3/ means Off/GA/Alexa/XIAOWEI**
		Auto Play	0x5	R/T	DevAck	Auto play/pause enable status. Payload: 0x00/0x01 means OFF/ON
		FindMyBuds	0x6	R/T	DevAck	Send this cmd to beep earbuds. Payload: 0x10: start beep master 0x11: start beep slave 0x00: stop beeping master 0x01: stop beeping slave
EQ Control	0x4	EQ preset	0x0	Tx	DevACK	**Set EQ preset on device**
		EQ settings	0x1	Tx	DevACK	**Set custom graphic/soundX EQ settings to device**

						Detail refer to 3.5
		ReqCurrenEQs	0x2	Rx	RetCurrentEQs	**Request current EQ settings info Detail refer to 3.5**
		RetCurrentEQs	0x3	Rx	No	**Return the current EQ setting to app Refer to 3.5**
		Personi-Fi Presets	0x4	Tx	DevAck	**Save Personi-Fi presets on device**
		ReqPersoni-FiPresets	0x5	Rx	RetPersoni-FiPresets	**Request personi-Fi presets, refer to 3.5**
App info	0x5	---	---	---	---	**---**
App Status	0x6	SyncAppStatus	0x1	Tx	DevAck	**App Report app status to device Refer to 3.7**
TouchPanel Control	0x7	setPanelActs	0x1	Tx	DevAck	**Refer to 3.8**
		reqPanelActs	0x2	Tx	retPanelActs	
		retPanelActs	0x3	Rx	---	
Noise Canceling Gains controi		setNCGainControl	0x4	Tx	DevAck	**Refer to 3.9**
		reqNCGainControl	0x5	Tx	retNCGainControl	
		retNCGainControl	0x6	Rx	---	

Table 3-2 Command summary

As the data exchange becomes busy when concurrently playing audio (A2DP) and App communication (BLE) together, it was necessary to reduce data packets. The above integrated multi commands enquiry into one single packets in order to reduce the BLE traffic.

3.2 ACK

3.2.1 Device ACK

Some commands requires a special "`DevACK`" confirmation feedback from device to APP, shows in **Table 3-3**. It depends on the features requirement.

Table 3-3 Dev ACK packet format

Section	Field	Size	Description
Header (Identifier: 0xAA, **Command ID: 0x00**, PayloadLen: 0x02)			
Payload	RequestCmdID	1 byte	The command ID which is sent from App.
	StatusCode	1 byte	Response status.

Usually a "`StatusCode` 0" indicates the success and otherwise indicates failure or error occurred.

3.2.2 APP ACK

In App, there provides `AppACK` to acknowledge device; it depends on the features requirement.

Table 3-4 Dev ACK packet format

Section	Field	Size	Description
Header (Identifier: 0xAA, **Command ID: 0x01**, PayloadLen: 0x02)			
Payload	RequestCmdID	1 byte	The command ID Which is sent from device.
	StatusCode	1 byte	Response status.

Usually a "`StatusCode` 0" indicates success and other none 0 codes indicate failure or error occur.

3.2.3 Disconnection

Device and App may need to disconnect with each other. Before establish on disconnection, a "ByeBye" command (refer to table at section 3.1) can be used on this purpose. Once ACK was received, then the formal disconnection will be announced.

Table 3-5 Dev ByeBye packet format

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x02, PayloadLen: 0x01)			
Payload	MsgCode	1 byte	Message codes that describe the quit reason.

Table 3-6 AppByeBye packet format

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x03, PayloadLen: 0x01)			
Payload	MsgCode	1 byte	Message codes that describe the quit reason.

Under Payload, the quit messages code defined as blow:

Code	Description
0x00	Unknown.
0x01	Device power off, App quit.

Table 3-7 quit messages code

3.3 Device Info

The following table 3-8 extended the details of device info [request or feedback] from section 3.1.

Table 3-8 Device Info commands

Category	Cmd ID	Name	Sub-CmdID	R/T	ACK	Description
Device Info	0x1	ReqDevInfo	0x1	TX	RetDevInfo	App request device information.
		RetDevInfo	0x2	RX	No	Device report information to App.

Two scenarios were involved here: - 1) Device Information request via command, 2)Auto feedback from device.

3.3.1 Device information request via command

App can query device information by `ReqDevInfo` command (**Table 3-9**).

Table 3-9 ReqDev Info packet format

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x11, PayloadLen: 0x00)			
Payload	NA.		

Once device received this commands, it replied with "`RetDevInfo`" commands to list all device information to App.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x12, PayloadLen: 0xnn)			
Payload	Device name	16 bytes	UTF-8, max 16 bytes. It is writable only. App can get the device name by BLE name.
	Product ID	2 bytes	Product ID.
	Model ID	1 bytes	Model ID.
	Battery info	1 bytes	1 bit (MSB): charging status, 1 means battery charging. 7 bit (LSB), 0-100 present 0% - 100%.
	MAC Address	6 bytes	Bluetooth MAC address.
	Firmware version	3 bytes	Byte 1: major Byte 2: minor Byte 3: revison Ex. Firmware version is "v1.2.3" Byte 1: 0x01/ Byte 2: 0x02 / Byte 3: 0x03
	Detail Battery Info	2 bytes	Byte 1: Left Earbud/HP battery info Byte 2: Right Earbud/HP battery info 1 bit (MSB): charging status, 1 means battery charging. 7 bit (LSB), 0-100 present 0% - 100%.
	Current Voltage	4 bytes	Byte1~Byte2: Left Earbud/HP current voltage Byte3~Byte4 : Right Earbud/HP current voltage /mv

Table 3-10 Device information format

3.3.2 Auto feedback from device

When firstly App connected to device, the device shall report all designed information to App **actively**, **without any App query**.

Device will also send `RetDevInfo` packets to App **actively**, to notify information changes.

Device shall resend packet until `AppACK` being dispatched from App.

3.4 Device Status

The following table 3-11 extended the details of device status [request or feedback] from section 3.1.

Table 3-10 Device Status commands

Category	Cmd ID	Name	Sub-CmdID	R/T	ACK	Description
Device Status	0x2	ReqDevStatus	0x1	TX	RetDevStatus	App request device Status.
		RetDevStatus	0x2	RX	No	Device report status to App.

Two scenarios were involved here: - 1) Device Status request via command, 2)Auto feedback from device.

3.4.1 Device Status request via command

App can query device information by `ReqDevInfo` command (**Table 3-9**).

Table 3-12 ReqDev Status packet format

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x21, Payload Len: 0x01)			
Payload	Status type (Status id).		

Table 3-12-1 ReqDev Status Type List

Category	CmdID	Name	Sub-CmdID	R/T	Description
Device Status type	0x3	All Status	0x0	R/T	All status including ANC, AA Mode, Auto Off, Battery, EQ settings
		ANC	0x1	R/T	Read/Write ANC enable status. Payload: 0x00/0x01 means OFF/ON
		Ambient Aware Mode and Talk Thru	0x2	R/T	Read AA/Talk Mode status. 0b0000000X: Talk Thru status 0b000000Y0: Abient Aware status 0 – off 0b00000001 – Talk Thru 0b00000010 – Abient Aware 0b00000011 – Talk Thru & Abient Aware
		Auto Off	0x3	R/T	Read Auto Off status, 2 different response as below, the 2nd option is preferred in new device: 1. Payload length 1 byte: 1 bit(MSB): 0/1 means disable/enable 7 bits(LSB): auto off time value /mins 2. Payload length 2 bytes: auto off time value in minutes, 0 means disabled.
		EQ preset	0x4	R/T	**Return EQ current preset** **Payload:** **Byte1: preset index**
		Multi AI	0x5	R/T	0/1/2/3 means off/GA/Alexa/xiaowei
		BT Connection Status	0x6	R	0/1 means disconnected/connected
		OTA upgrade Status	0x7	R	0/1/2 means not upgrading/JBL is upgrading/GA is upgrading
		Auto play/pause enable status	0X8	R	0/1 Means off/on (If device not support this feature, also return 0).
		TWS connection Status	0x9	R	0/1 means Slave disconnected/connected

		Device status for Personi-Fi mode	0xa	R	0/1 Means device busy/device free
	0x4	In ear status (IR test)	0x1	R	Return in ear status.
		Sealing Test Status	0x2	R	Return sealing test result.

Once device received this commands, it replied with "`RetDevStatus`" commands to list all current specific status to App.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x22, Payload Len: 0xnn)			
Payload	Status Type	1 byte	Status type id
	ANC Status	1 byte	0x00/0x01 means OFF/ON
	Ambient Aware Mode Status and TalkThru Mode	1 byte	0b0000000X: TalkThru 0b000000Y0: AbientAware 0 – off 0b00000001 – TalkThru 0b00000010 – AbientAware 0b00000011 – TalkThru & AbientAware
	Auto Off Status 1	1 byte	Payload length 1 byte: 1 bit(MSB): 0/1 means disable/enable 7 bits(LSB): auto off time value /mins (If device doesn't support this field, filled with 0xFF, refer to Auto Off Status 2)
	EQ Preset	1 byte	**0/1/2/3/4** **means off / jazz/ vocal/ bass/ custom**
	Multi AI Index	1 byte	0/1/2 means off/GA/Alexa
	BT Connection Status	1 byte	0/1 means disconnected/connected
	OTA upgrade Status	1 byte	0/1/2 means not upgrading/JBL is upgrading/GA is upgrading
	Auto play/pause enable status	1 byte	0/1 Means off/on (If device not support this feature, also return 0).
	TWS connection Status	1 byte	0/1 means Slave disconnected/connected
	Auto Off Status 2	2 bytes	Payload length 2 bytes: auto off time value in minutes, 0 means disabled (If device doesn't this field, filled with 0xFFFF)
	Follow info not included in getAllStatus (0x30)		
	Device Status for Personi-Fi mode	1 byte	0x00 – DEVICE_BUSY 0x01 – DEVICE_FREE App connects(BLE connection) to the headphone. It can be DEVICE_BUSY if there is a phone call / music streaming in progress. If not, it receives DEVICE_FREE; Receives a call in either Personi-Fi mode or Hearing test mode. The app receives DEVICE_BUSY status. When call/streaming ends. The app receives DEVICE_FREE status.

	In ear status (IR test)	2 bytes	0x00 – Left device not in ear; 0x01 – Left device in ear;
			0x00 – Right device not in ear. 0x01 – Right device in ear.
	Sealing Test Status	2 bytes	0x00 – Left device not ready; 0x01 – Left device ready;
			0x00 – Right device not ready. 0x01 – Right device ready.
	…	…	…

Table 3-13 Device Status format

If the status type is not value “All status(0x30)”, the retDevStatus playload only need to carry the relavtive status value accordingly.

3.4.2 Auto feedback from device

Device will also send `RetDevStatus` packet to App **actively**, to notify status changes.

Device shall resend packet until `AppACK` being dispatched from App.

3.4.3 Find my buds Status via command

To request the find my buds status, refer to below command format to request the status from app to device:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x23**, PayloadLen: 0x0)			

Table 3-13-1 req command format for FindMyBuds

Once device receive above reqFindMyBudsStatus cmd, device should response the retFindMyBudsStatus as below format:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x24**, PayloadLen: 0x01)			
Payload	Personi-Fi enable	1 bytes	Send this cmd to beep earbuds. Payload: 0x00: master & slave are not beeping 0x01: master & slave are beeping 0x02: only master beeping 0x03: only slave beeping 0xff: unknown

Table 3-13-2 ret command format for FindMyBuds

3.5 EQ Control

EQ control packet format generally followed by **Table3-1,** but for EQ, the payload is more complicated. The EQ control have 3 commands, set preset index/ set Graphic EQ/ get current EQ.

3.5.1 EQ Setting via command

To set up the EQ, use these two command EQ preset cmd(0x40) and EQ settings cmd(0x41). EQ Presets has 4 types, off/jazz/vocal/bass. Set up EQ presets followed by **table 3-14**.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x40**, PayloadLen: 0xnn)			
Payload	Preset index	1 bytes	**0/1/2/3/4** **means off / jazz/ vocal/ bass**

Table 3-14 EQ control preset command

And another one "custom" is only for using setting up EQ band settings followed by **table 3-15**

Section	Field			Size	Description
Header (Identifier: 0xAA, Command ID: 0x41, PayloadLen: 0xnn, packcount: 0xnn)					
Payload	PackIndex			1 byte	**Packet index is tempary taking up at the first byte of each transferring unit data** **Refer to 3.5.3**
	Preset index			1 byte	**Always value "4" for setting EQ bands**
	EQ category			1 byte	0x00/ 0x01/ 0x02 / 0x03 Design EQ/ Graphic EQ/ Total EQ / Personi-Fi EQ
	Calib			**4 bytes**	**Max gain calib value**
	Sample Rate			1 byte	Value * k (ex. Value = 48, actual rate = 48 * k)
	Gain0			4 byte	Left gain value
	Gain1			4 byte	Right gain value
	Band Count			4 byte	Number of the band count
	IIR param	Band0	Type	4 byte	Band type
			Gain	4 bytes	Gain value
			Fc	4 bytes	Frequency
			Q	4 bytes	Q value
		…	…		…
		Band (count - 1)	…		…
Footer (Identifier: 0xAA, Command category: 0x05 Command ID: 0x41)					

Table 3-15 EQ control setting command

3.5.2 Current EQ via command

App can query current EQ setting via `ReqCurrentEQs` **Table 3-16.**

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x42**, PayloadLen: 0xnn)			
Payload	EQ category	1 bytes	Request the current EQ Type 0x00/ 0x01 / 0x03 Design EQ/ Graphic EQ / Personi-Fi EQ

Table 3-16 ReqCurrentEQs command

Once device received this commands, it replied with "`RetCurrentEQs`" commands to list all specific status to App.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x43, PayloadLen: 0xnn, packet count: 0xnn)			
Payload	PackIndex	1 byte	**Packet index is tempary taking up at the first byte of each transferring unit data**

					Refer to 3.5.3
	Preset index			1 byte	**0/1/2/3/4 means off / jazz/ vocal/ bass/ custom**
	EQ category			1 byte	0x00/ 0x01 / 0x03 Design EQ/ Graphic EQ / Personi-Fi EQ
	Sample Rate			1 byte	Value * k (ex. Value = 48, actual rate = 48 * k)
	Gain0			4 byte	Left gain value
	Gain1			4 byte	Right gain value
	Band Count			4 byte	Number of the band count
	IIR param	Band0	Type	4 byte	Band type
			Gain	4 bytes	Gain value
			Fc	4 bytes	Freqency
			Q	4 bytes	Q value
		…	…		…
		Band (count - 1)	…		…
Footer (Identifier: 0xAA, Command category: 0x04 Command ID: 0x43)					

Table 3-17 RetCurrentEQs packet

The EQ type defines the IIR Param's frequency as below table:

Category	Name	id	Band count	Band Frequecy array
EQ category	Design EQ	0x0	6	---
	Graphic EQ	0x1	10	---
	Total EQ	0X2	16	
	Personi-Fi EQ	0x3	10	

Table 3-18 EQ Category

3.5.3 Data Transfer

Due to the payload data size over large, the payload would split to < n * 80> bytes/unit to transfer, n means the amount of the units, the transfer flow as:

a. PHASE1. transfer EQ setting Header ->PHASE2. transfer payload unit data -> PHASE3. transfer App/Dev Fin Ack.

b. *App & device need to send the APP Ack/ Device Ack at per phase*

***Packet count & packet index (in table 3-15 & table 3-17) remark**:*
Packet count: the sender needs to attach the upcoming payload's total unit amount in the header.
Packet index: each unit data's first byte has to be an index value to indicate which unit data is transferring. The packet index is not belongs to any part of the whole payload. Only using for transferring, please keep in mind that remove the packet index byte when assembly the payload data is required.

3.5.4 Personi Fi EQ Presets

The personi-fi presets including Personi-Fi on/off and DJ preset index. To save up the personi-fi presets, use below format for the cmd: **table 3-19**.

Section	Field	Size	Description

Header (Identifier: 0xAA, Command ID: **0x44**, PayloadLen: 0x03)			
Payload	Personi-Fi enable	1 byte	**0/1** **means off / on**
	DJ Preset index	2 bytes	**0….255** **Dj Preset Off….DJ preset index 255**

Table 3-19 command format for Personi-Fi presets

App can request the personi-Fi Presets stored in device via below cmd:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x45, PayloadLen: 0x00)			
Payload	NA.		

Table 3-20

Device should reply the RetPersoni-FiPreset cmd as below:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x46**, PayloadLen: 0x03)			
Payload	Personi-Fi state	1 byte	**0/1** **means off / on**
	DJ Preset index state	2 bytes	**0….255** **Dj Preset Off….DJ preset index 255**

Table 3-21

3.5.5 Personi Fi EQ mode/Hearing Test

The personi-fi mode including Personi-Fi mode and hearing test. To save up the personi-fi mode and hearing test status, use below format for the SetPersoniFiHearingTest command: **table 3-22**.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x47**, PayloadLen: 0x03)			
Payload	Personi-Fi mode enable	1 byte	**0/1** **means off / on**
	Personi-Fi hearing test enable	1 byte	**0/1** **means off / on**
	Featuresto be enabled / disabled	1 byte	**NA**

Table 3-22 command format for Personi-Fi mode/Hearing test

Device should reply the RetPersoniFiHearingTest command/response as below:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x48**, PayloadLen: 0x03)			
Payload	Personi-Fi mode enable	1 byte	**0xY0 - FAILURE** **0xY1 - SUCCESS** **0x1X - BACKUP_IN_PROGRESS** **0x2X - BACKUP_COMPLETED** **0x3X - RESTORE_IN_PROGRESS** **0x4X - RESTORE_COMPLETED** **0xFF – Invalid Status** **0x10 - BACKUP_IN_PROGRESS_FAILURE** **0x20 - BACKUP_COMPLETED_FAILURE** **0x30 - RESTORE_IN_PROGRESS_FAILURE** **0x40 - RESTORE_COMPLETED_FAILURE**

			0x11 - BACKUP_IN_PROGRESS_SUCCESS **0x21 - BACKUP_COMPLETED_SUCCESS** **0x31 - RESTORE_IN_PROGRESS_SUCCESS** **0x41 - RESTORE_COMPLETED_SUCCESS**
	Personi-Fi hearing test enable	1 byte	**0xY0 - FAILURE** **0xY1 - SUCCESS** **0x1X - BACKUP_IN_PROGRESS** **0x2X - BACKUP_COMPLETED** **0x3X - RESTORE_IN_PROGRESS** **0x4X - RESTORE_COMPLETED** **0xFF – Invalid status** **0x10 - BACKUP_IN_PROGRESS_FAILURE** **0x20 - BACKUP_COMPLETED_FAILURE** **0x30 - RESTORE_IN_PROGRESS_FAILURE** **0x40 - RESTORE_COMPLETED_FAILURE** **0x11 - BACKUP_IN_PROGRESS_SUCCESS** **0x21 - BACKUP_COMPLETED_SUCCESS** **0x31 - RESTORE_IN_PROGRESS_SUCCESS** **0x41 - RESTORE_COMPLETED_SUCCESS**
	Featuresto be enabled / disabled	1 byte	**NA**

Table 3-23

And app can query current personi-fi mode and hearing test mode ReqPersoniFiHearingTest command in Table 3-24.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x49**, PayloadLen: 0x00)			
Payload	Personi-Fi mode / Hearing Test	0 byte	**Response refer to command 0x48;**

Table 3-24 query personi-fi mode/hearing test command

3.5.6 Personi Fi EQ hearing test change volume

The personi-fi hearing test change volume. Use below format for the SetHearingTestVolume command: **table 3-25**.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x4a**, PayloadLen: 0x0a)			
Payload	Left Channel	1 byte	**0/1/2** **0 – no valid volume, ignore left channel in this command;** **1 – valid volume;** **2 – reset volume (headphone can restore previously set volume, if there was any).**
	Left Channel Volume	4 bytes	**Left channel volume value**
	Right Channel	1 byte	**0/1/2** **0 – no valid volume, ignore right channel in this command;** **1 – valid volume;** **2 – reset volume (headphone can restore previously set volume, if there was any).**
	Right Channel Volume	4 bytes	**Right channel volume value**

Table 3-25 command format for Personi-Fi test change volume

Device should reply the RetHearingTestVolume command/response as below:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x4b**, PayloadLen: 0x08)			
Payload	Left Channel Volume	4 bytes	**Left channel volume value**
	Right Channel Volume	4 bytes	**Right channel volume value**

Table 3-26

And app can query current hearing test volume RegHearingTestVolume command in Table 3-27.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x4c**, PayloadLen: 0x00)			
Payload	Hearing Test Volume	0 byte	**Response refer to command 0x4b.**

Table 3-27 query personi-fi mode/hearing test command

3.6 App Info

TBD.

3.7 App Status

The following table 3-19 extended the details of device status [request or feedback] from section 3.1.

Table 3-19 App Status commands

Category	Cmd ID	Name	Sub-CmdID	R/T	ACK	Description
App Status	0x6	SyncAppStatus	0x1	TX	DeviceAck	App report App Status to device

3.7.1 App Status Report via command

App can report app information by `SyncAppStatus` command (Table 3-12).

Table 3-12 SyncAppStatus packet format

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x61, PayloadLen: 0xnn)			
Payload	Connected CRC	2 bytes	The exact phone' scrc value
	…	…	…

Once device received this commands, it should reply DeviceAck to App.

3.8 Touch Panel Control

The touch panel control including:

a. Customize function for touch panel action: **table 3-8-1**.
b. Searching the current functions of the panel actions: request the customized touch panel status for **table 3-8-2. Return the customized touch panel status to App for table 3-8-3**

Set the customized panel to device via:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x71**, PayloadLen: 0x03)			
Payload	Enable action/Ges type	1 Byte	00/01 to disable/enable the Action/Gesture.
	Touch action/Ges type	1 byte	The target action/gesture to customize. Find table 3-8-4 for the action type definition.
	Touch function type	1 byte	**The fuction for the target action/gesture.** Find table 3-8-5 for the function type definition.

Table 3-8-1

Request the customized touch panel status via:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x72**, PayloadLen: 0x01)			
Payload	Touch action type	1 bytes	The target action/gesture to search. Find table 3-8-4 for the action type definition.

Table 3-8-2

Return the customized touch panel status to App via:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x73**, PayloadLen: 0x03)			
Payload	action/Gestype	1 byte	00/01 to disable/enable the Action/Gesture.
	Touch action type	1 byte	**Search result of the target action/Gesture**
	Touch function type	1 byte	**Search result of the current function on the target action/Gesture**

Table 3-8-3

The action & Function types definition.

Action/Gesture Type	Value
Left Whole Touch Panel	0x00
Right Whole Touch Panel	0x01
Left Swipe forward	0x02
Left Swipe backward	0x03
Right Swipe forward	0x04
Right Swipe backward	0x05
Left Tap	0x06
Left Double Tap	0x07
Left Triple Tap	0x08
Right Tap	0x09
Right Double Tap	0x0a
Right triple Tap	0x0b
Left Tap&Hold	0x0c
Left Double Tap&Hold	0x0d
Right Tap&Hold	0x0e
Right DoubleTap&Hold	0x0f

Function type	Value
Default	0x00
Volume up	0x01
Volume down	0x02
Ambient Aware	0x03
TalkThru	0x04
Next Track	0x05
Previous Track	0x06
…TBD	…

Table 3-8-4

3.9 Noise Canceling Gain Control

This noise canceling control including gains adjustment for ANC/AA/TT currently.

Set the noise canceling control to device via:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x74**, PayloadLen: 0x05)			
Payload	type	1 Byte	The target type to set the control 01: ANC, 02: AA 03: TT
	Gains	4 bytes	The gain db value to set Range: -6.0 to 6.0 db, Step size/Uint: 0.2db

Table 3-9-1

Request the noise canceling gains control status from device via:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x75**, PayloadLen: 0x01)			
Payload	type	1 bytes	The target type to set the control 01: ANC, 02: AA 03: TT

Table 3-9-2

Return the noise canceling gains control status to App via:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x76**, PayloadLen: 0x05)			
Payload	type	1 Byte	The target type to set the control 01: ANC, 02: AA 03: TT
	Gains	4 bytes	The gain db value to set Range: -6.0 to 6.0 db, Step size/Uint: 0.2db

Table 3-9-2

3.10 HP Analytics Info Data

FW would collect the user events (ex. Playback duration, anc status, play/pause times, voice AI status etc), App would get those events and upload to the Cloud. Once app finish receiving analytics info data, app would need to request a cmd to clean all those data so that to record next set of analytics data.

App can request the analytics data via:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x13, Payload Len: 0x00)			
Payload	N/A		

Table 3-10-1

Device would return the data via:

Section	Field	Size	Description

Header (Identifier: 0xAA, Command ID: 0x14, Payload Len: 0xxx, according to versiong number)	
Payload	Data…

Table 3-10-2

The Payload Data of the format as below

```
struct
{  /*recorde on both side*/
      unsigned Left_right: 1 bit;            /*Left=0; Right=1;*/
      unsigned is 0x47: 1 bit;                              /* Tws=1; noTws=0*/
      unsigned roleSwitchEnabled: 1 bit; /* Enabled=1;*/
      unsigned state:4 bits;                 /* 0:data OK; 1:erasered, 2:check sum error……*/
      unsigned VoiceAssistant: 1 byte;     /*Current VA; None:0,  AMA:1,  GVA:2, XiaoWei:3*/
      unsigned version: 2 byte;              /*e.g.1.2.3, Ox1230*/
/* recorded by master(<--phone)*/
   int16_t GA_times;              /*How many times is the voice assistant queried*/
   int16_t AMA_times;
   int16_t Xiaowei_times;
   int16_t Play_Pause_times;
   int16_t Prev_Next_times;
   int16_t VA_activity_times;
   int16_t Manual_Pairing_times;
   int16_t AA_active_times;
   int16_t TT_active_times;
   int16_t ANC_times;

   int32_t PowerOn_duration_Tws;
   int32_t Playtime_duration_Tws;
   int32_t BTconnect_duration_Tws;
   int32_t Voicecall_duration_Tws;

   /* recorded respectively*/
   int16_t LowBatt_warning_times_R;
   int16_t LowBatt_warning_times_L;    // only used for tws slave, if stereo, this is not used.

   int32_t PowerOn_duration_single_R;
   int32_t Playtime_duration_single_R;
   int32_t BTconnect_duration_single_R;
   int32_t Voicecall_duration_single_R;

   int32_t PowerOn_duration_single_L;
   int32_t Playtime_duration_single_L;
   int32_t BTconnect_duration_single_L;
   int32_t Voicecall_duration_single_L;
}Analytics.
```

Please refer to the "TWS Analytics High Level Design" doc for any changes/update for the data format/struct.

App can request to clean the analytics info data via:

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: 0x15, Payload Len: 0x00)			
Payload	NA		

Table 3-10-3

After receiving the clean request, device should return the **DevAck** as response to app if the clean is success.

3.11 Smart switch

App can set smart switch information by `SetSmartSwitch` command (**Table3-11-1**)

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x81**, PayloadLen: 0x08)			
Payload	AAC	2 bytes	**AAC value in Hex format**
	SBC	2 bytes	**SBC value in Hex format**
	Latency	2 bytes	**Latency in milliseconds in Hex format**
	Second Latency	2 bytes	**Latency in milliseconds in Hex format. Ignore this field if device doesn't support the second latency.**

Table 3-11-1 Set Smart Switch command

App can query smart switch information by `ReqSmartSwitch` command (**Table 3-11-2**).

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x82**, PayloadLen: 0x00)			
Payload	Smart Switch Info	0 byte	Response refer to command 0x83

Table 3-11-2 Query Smart switch command

Once device received `SetSmartSwitch/ReqSmartSwitch command`, it replied with "`RetSmartSwitch`" command to App.

Section	Field	Size	Description
Header (Identifier: 0xAA, Command ID: **0x83**, PayloadLen: 0x08)			
Payload	AAC	2 bytes	**AAC value in Hex format**
	SBC	2 bytes	**SBC value in Hex format**
	Latency	2 bytes	**Latency in milliseconds in Hex format**
	Second Latency	2 bytes	**Latency in milliseconds in Hex format. Fill this field with 0xFFFF if device doesn't support second latency.**

Table 3-11-3 Ret Smart switch command

4. Error Handling

During ACK session: If no reply from device within 300ms, it became time out. The App shall retry the commands for 3 times with the same 300ms rule. If all retry failed or any other exceptions were being happened, App and device should take the BLE back to re-connection state.