SDK API User's Guide (Bluetooth)

Version 1.4.0

Display Audio

Solution Team

Release information

The following changes have been make to this document.

Change History

Date	Change
06 Dec. 2017	First release for v1.0.0
14 Nov. 2018	Second release for v1.1.0
18 Feb. 2019	Third release for v1.2.0
24 May. 2019	Fourth release for v1.3.0
30 Sep. 2019	Fifth release for v1.4.0

Contact us

[11595] BundangYemiji Bldg. 12F, 31 Hwangsaeul-ro 258 beon gil, Bundang-gu, Sungnam-city, Gyeonggi-do, Korea.

TEL: 82-31-698-7400

FAX:82-31-698-7455

http://www.nexell.co.kr

Table of contents

Chap 1. Library information

1.1 Overview

- Library name : libnxbt.so
- Interface class name : INX_BT
- Interface header : INX_BT.h
- Dependency libraries
  - Shared libraries : libappbt.so (Broadcom BT wrapper), libnxalsa.so (BT Nexell audio platform), libxml2.so (v2.9.7)
  - Static library : libbsa.a (Broadcom BT stack, Included in libappbt.so)
- Operational prerequisites : bsa_server (Bluetooth simple API server daemon)

1.2 INX_BT class instance

The NXBT class provides functions to easily handle the broadcom BT stack

Header	#include <INX_BT.h>
Access class instance	extern INX_BT* getInstance(void)

1.3 List of all members for INX_BT

```
INX_BT(void) {}
virtual ~INX_BT(void) {}

/* NXBT manager APIs */
virtual int32_t initDevManager(void) = 0;
virtual int32_t getVersionInfoBSA(BSA_version_info_t *bsa_version) = 0;
virtual void setRecoveryCommand(const char *command) = 0;
virtual int32_t enableAutoConnection(bool enable) = 0;
virtual bool isAutoConnection(void) = 0;
virtual void autoConnection(bool enable) = 0;
virtual int32_t requestLastAVKConnectedDevIndex(void) = 0;
virtual int32_t requestLastHSConnectedDevIndex(void) = 0;
virtual int32_t acceptPairing(void) = 0;
virtual int32_t rejectPairing(void) = 0;
virtual int32_t requestPairDevice(int32_t device_index) = 0;
virtual int32_t unpairDevice(int32_t device_index) = 0;
virtual int32_t enableAutoPairing(bool enable) = 0;
virtual bool isAutoPairing(void) = 0;
virtual int32_t enableDiscoverable(bool enable) = 0;
virtual bool isDiscoverable(void) = 0;
virtual int32_t renameLocalDevice(const char *name) = 0;
virtual char* getLocalDevName(void) = 0;
virtual unsigned char* getLocalAddress(void) = 0;
virtual int32_t getPairedDevCount(void) = 0;
virtual int32_t getPairedDevInfoByIndex(int32_t device_index, char *name,
unsigned char *bd_addr) = 0;
virtual int32_t getPairedDevAddrByIndex(int32_t device_index,
unsigned char *bd_addr) = 0;
virtual int32_t getPairedDevNameByIndex(int32_t device_index, char *name) = 0;
virtual int32_t getPairedDevIndexByAddr(unsigned char *bd_addr) = 0;
virtual char* getPairedDevNameByAddr(unsigned char *bd_addr) = 0;
virtual void setALSADevName(const char *playback_avk, const char *playback_hs,
const char *capture_hs, const char *playback_hs_sco, const char *capture_hs_sco) = 0;
virtual int32_t startDiscovery(void) = 0;
virtual int32_t stopDiscovery(void) = 0;
virtual int32_t getDiscoveredDevCount(void) = 0;
virtual int32_t getDiscoveredDevInfoByIndex(int32_t device_index, char *name,
unsigned char *bd_addr, unsigned char *class_of_device, char *class_name,
int32_t *rssi) = 0;
virtual int32_t bondDevice(int32_t device_index) = 0;
virtual int32_t cancelBondingDevice(int32_t device_index) = 0;

/* NXBT AVK service APIs */
virtual int32_t openAudioAVK(void) = 0;
virtual void closeAudioAVK(void) = 0;
virtual bool isAudioStatusAVK(void) = 0;
virtual bool isConnectedAVK(void) = 0;
virtual int32_t connectToAVK(int32_t device_index) = 0;
virtual int32_t disconnectFromAVK(unsigned char *bd_addr) = 0;
virtual int32_t getConnectionNumberAVK(void) = 0;
virtual int32_t getConnectionDevAddrAVK(int32_t connected_index,
unsigned char *bd_addr) = 0;
virtual int32_t requestGetElementAttr(unsigned char *bd_addr) = 0;
virtual int32_t requestPlayerValues(unsigned char *bd_addr) = 0;
virtual int32_t playStartAVK(unsigned char *bd_addr) = 0;
virtual int32_t playStopAVK(unsigned char *bd_addr) = 0;
virtual int32_t playPauseAVK(unsigned char *bd_addr) = 0;
virtual int32_t playNextAVK(unsigned char *bd_addr) = 0;
virtual int32_t playPrevAVK(unsigned char *bd_addr) = 0;
virtual int32_t playerEqualizerAVK(unsigned char *bd_addr, unsigned char value) = 0;
virtual int32_t playerShuffleAVK(unsigned char *bd_addr, unsigned char value) = 0;
virtual int32_t playerRepeatAVK(unsigned char *bd_addr, unsigned char value) = 0;
virtual int32_t playerScanAVK(unsigned char *bd_addr, unsigned char value) = 0;

/* NXBT HS service APIs */
virtual bool isConnectedHS(void) = 0;
virtual int32_t requestCallIndicator(void) = 0;
virtual void requestCurrentCalls(void) = 0;
virtual int32_t getConnectionDevAddrHS(unsigned char *bd_addr) = 0;
virtual int32_t connectToHS(int32_t device_index) = 0;
virtual int32_t disconnectFromHS(void) = 0;
virtual int32_t pickUpCall(void) = 0;
virtual int32_t hangUpCall(void) = 0;
virtual int32_t openAudioHS(void) = 0;
virtual int32_t closeAudioHS(void) = 0;
virtual bool isOpenedAudioHS(void) = 0;
virtual void muteMicrophoneHS(bool mute) = 0;
virtual bool isMutedMicrophoneHS(void) = 0;
virtual int32_t dialPhoneNumber(const char *number) = 0;
virtual int32_t reDialPhoneNumber(void) = 0;
virtual int32_t setATCommandDTMF(char key) = 0;
virtual int32_t requestCallNumber(void) = 0;
virtual int32_t requestCallOperName(void) = 0;
virtual int32_t getCurrentBattChargingStatus(void) = 0;
virtual int32_t startVoiceRecognition(void) = 0;
virtual int32_t stopVoiceRecognition(void) = 0;

/* NXBT PBC service APIs */
virtual bool isConnectedPBC(void) = 0;
virtual int32_t connectToPBC(int32_t device_index) = 0;
virtual int32_t disconnectFromPBC(void) = 0;
virtual int32_t abortPBC(void) = 0;
virtual int32_t getContactFromPBC(void) = 0;
virtual int32_t getCallHistoryFromPBC(void) = 0;

/* NXBT MCE service APIs */
virtual bool isConnectedMCE(void) = 0;
virtual int32_t connectToMCE(int32_t device_index) = 0;
virtual int32_t disconnectFromMCE(void) = 0;
virtual int32_t abortMCE(void) = 0;
virtual int32_t startNotifyServerFromMCE(void) = 0;
virtual int32_t stopNotifyServerFromMCE(void) = 0;
virtual int32_t getParserBmsg(char *fullName, char *phoneNumber, char *msgBody) = 0;

/* NXBT UI callback register functions */
virtual void registerMGTOpenCbManager(void *pObj, void (*cbFunc)(void *, int32_t)) = 0;
virtual void registerMGTDisconnectedCbManager(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerPairingFailedCbManager(void *pObj, void (*cbFunc)(void *,
int32_t)) = 0;
virtual void registerDiscoveryCompleteCbManager(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerPairedDevicesCbManager(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerUnpairedDevicesCbManager(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerPairingRequestCbManager(void *pObj, void (*cbFunc)(void *, bool,
char *, unsigned char *, int32_t)) = 0;
virtual void registerLinkDownEventCbManager(void *pObj, void (*cbFunc)(void *,
unsigned char *, int32_t)) = 0;a
virtual void registerOpenFailedCbAVK(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerStreamingStartedCbAVK(void *pObj, void (*cbFunc)(void *, bool)) = 0;
virtual void registerStreamingStoppedCbAVK(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerConnectionStatusCbAVK(void *pObj, void (*cbFunc)(void *, bool,
char *, unsigned char *)) = 0;
virtual void registerConnectionStatusCbAVKRC(void *pObj, void (*cbFunc)(void *,
bool)) = 0;
virtual void registerPlayStatusCbAVK(void *pObj, void (*cbFunc)(void *, int32_t)) = 0;
virtual void registerMediaElementCbAVK(void *pObj, void (*cbFunc)(void *, char *, char *,
char *, char *, int32_t)) = 0;
virtual void registerPlayPositionCbAVK(void *pObj, void (*cbFunc)(void *, int32_t,
int32_t)) = 0;
virtual void registerPlayerValuesCbAVK(void *pObj, void (*cbFunc)(void *, int32_t,
int32_t, int32_t, int32_t)) = 0;
virtual void registerListPlayerAttrCbAVK(void *pObj, void (*cbFunc)(void *, bool, bool,
bool, bool)) = 0;
virtual void registerListPlayerValuesCbAVK(void *pObj, void (*cbFunc)(void *, int32_t,
int32_t, unsigned char *)) = 0;
virtual void registerOpenFailedCbHS(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerConnectionStatusCbHS(void *pObj, void (*cbFunc)(void *, bool,
char *, unsigned char *)) = 0;
virtual void registerInbandRingSupportedCbHS(void *pObj, void (*cbFunc)(void *,
bool)) = 0;
virtual void registerCallStatusCbHS(void *pObj, void (*cbFunc)(void *, int32_t)) = 0;
virtual void registerBatteryStatusCbHS(void *pObj, void (*cbFunc)(void *, int32_t)) = 0;
virtual void registerCallOperNameCbHS(void *pObj, void (*cbFunc)(void *, char *)) = 0;
virtual void registerCurrentCalllsCbHS(void *pObj, void (*cbFunc)(void *, char *)) = 0;
virtual void registerCallNumberCbHS(void *pObj, void (*cbFunc)(void *, char *)) = 0;
virtual void registerAudioMuteStatusCbHS(void *pObj, void (*cbFunc)(void *, bool,
bool)) = 0;
virtual void registerVoiceRecognitionStatusCbHS(void *pObj, void (*cbFunc)(void *,
unsigned short)) = 0;
virtual void registerIncommingCallNumberCbHS(void *pObj, void (*cbFunc)(void *,
char *)) = 0;
virtual void registerCallIndicatorCbHS(void *pObj, void (*cbFunc)(void *, char *)) = 0;
virtual void registerCallIndicatorParsingValuesCbHS(void *pObj, void (*cbFunc)(void *,
int32_t, int32_t, int32_t, int32_t, int32_t, int32_t, int32_t)) = 0;
virtual void registerOpenFailedCbPBC(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerConnectionStatusCbPBC(void *pObj, void (*cbFunc)(void *, bool)) = 0;
virtual void registerNotifyGetPhonebookCbPBC(void *pObj, void (*cbFunc)(void *,
int32_t)) = 0;
virtual void registerListDataCbPBC(void *pObj, void (*cbFunc)(void *,
unsigned char *)) = 0;
virtual void registerOpenFailedCbMCE(void *pObj, void (*cbFunc)(void *)) = 0;
virtual void registerConnectionStatusCbMCE(void *pObj, void (*cbFunc)(void *, bool)) = 0;
virtual void registerNotifyGetMessageCbMCE(void *pObj, void (*cbFunc)(void *)) = 0;
```

Chap 2. Pure virtual functions

2.1 NXBT manager APIs

2.1.1 initDevManager

`int32_t initDevManager(void)`
Description This function connects to bsa_server.
Arguments void
Return Value 0 : Success, -1 : Fail

2.1.2 getVersionInfoBSA

`int32_t getVersionInfoBSA(BSA_version_info_t *bsa_version)`
Description This function gets BSA version information.
Arguments bsa_version : Input structure to store BSA version - server_version : BSA server version - fw_version : Firmware version
Return Value 0 : Success, -1 : Fail

2.1.3 setRecoveryCommand

`void setRecoveryCommand(const char *command)`
Description This function sets bsa_server command option for recovery.
Arguments command : bsa_server command.
Return Value void

2.1.4 enableAutoConnection

`int32_t enableAutoConnection(bool enable)`
Description This function decides whether or not to apply the automatic connection scenario.
Arguments enable : True or false
Return Value 0 : Success, -1 : Fail

2.1.5 isAutoConnection

`bool isAutoConnection(void)`
Description This function checks automatic connection is true or false.
Arguments void
Return Value True : Enabled, False : Disabled

2.1.6 autoConnection

`void autoConnection(bool enable)`
Description This function applies automatic connection.
Arguments enable : True or false
Return Value void

2.1.7 requestLastAVKConnectedDevIndex

`int32_t requestLastAVKConnectedDevIndex(void)`
Description This function request latest AVK connected device index from the bt_devices.xml.
Arguments void
Return Value Device's index

2.1.8 requestLastHSConnectedDevIndex

`int32_t requestLastHSConnectedDevIndex(void)`

Description This function request latest HS connected device index from the bt_devices.xml.
Arguments void
Return Value Device’s index

2.1.9 acceptPairing

`int32_t acceptPairing(void)`
Description This function accepts the pairing request.
Arguments void
Return Value 0 : Success, -1 : Fail

2.1.10 rejectPairing

`int32_t rejectPairing(void)`
Description This function rejects the pairing request.
Arguments void
Return Value 0 : Success, -1 : Fail

2.1.11 requestPairDevice

`int32_t requestPairDevice(int32_t device_index)`
Description This function rejects the pairing request.
Arguments device_index : Index of the paired device.
Return Value 0 : Success, -1 : Fail

2.1.12 unpairDevice

`int32_t unpairDevice(int32_t device_index)`
Description This function unpairs the paired device.

Arguments device_index : Index of the paired device
Return Value 0 : Success, -1 : Fail

2.1.13 enableAutoPairing

`int32_t enableAutoPairing(bool enable)`
Description This function decides whether or not to apply automatic pairing.
Arguments enable : True or false
Return Value 0 : Success, -1 : Fail

2.1.14 isAutoPairing

`bool isAutoPairing(void)`
Description This function checks the status of auto-pairing.
Arguments void
Return Value True : Enabled, False : Disabled

2.1.15 enableDiscoverable

`int32_t enableDiscoverable(bool enable)`
Description This function decides whether or not to apply the discoverable mode.
Arguments enable : True or false
Return Value 0 : Success, -1 : Fail

2.1.16 isDiscoverable

`bool isDiscoverable(void)`
Description This function checks the discoverable mode.
Arguments void

Return Value True : Enabled, False : Disabled

2.1.17 renameLocalDevice

`int32_t renameLocalDevice(const char *name)`
Description This function changes the local device's name.
Arguments name : Name to be changed
Return Value 0 : Success, -1 : Fail

2.1.18 getLocalDevName

`char* getLocalDevName(void)`
Description This function reads the local device's name.
Arguments void
Return Value Local device's name

2.1.19 getLocalAddress

`unsigned char* getLocalAddress(void)`
Description This function reads the local device's address.
Arguments void
Return Value Local device's address

2.1.20 getPairedDevCount

`int32_t getPairedDevCount(void)`
Description This function gets the number of paired devices.
Arguments void
Return Value

Number of paired devices

2.1.21 getPairedDevInfoByIndex

`int32_t getPairedDevInfoByIndex(int32_t device_index, char *name, unsigned char *bd_addr)`
Description This function retrieves information about devices paired by index.
Arguments device_index : Index of the paired devices name : Input buffer to store name bd_addr : Input buffer to store 6bytes address
Return Value 0 : Success, -1 : Fail

2.1.22 getPairedDevAddrByIndex

`int32_t getPairedDevAddrByIndex(int32_t device_index, unsigned char *bd_addr)`
Description This function retrieves address about devices paired by index.
Arguments device_index : Index of the paired devices bd_addr : Input buffer to store 6bytes address
Return Value 0 : Success, -1 : Fail

2.1.23 getPairedDevNameByIndex

`int32_t getPairedDevNameByIndex(int32_t device_index, char *name)`
Description This function retrieves name about devices paired by index.
Arguments device_index : Index of the paired devices name : Input buffer to store name
Return Value 0 : Success, -1 : Fail

2.1.24 getPairedDevIndexByAddr

`int32_t getPairedDevIndexByAddr(unsigned char *bd_addr)`
Description This function retrieves index about devices paired by device's address.
Arguments

bd_addr : Input buffer to store 6bytes address
Return Value 0 : Success, -1 : Fail

2.1.25 getPairedDevNameByAddr

`char* getPairedDevNameByAddr(unsigned char *bd_addr)`
Description This function retrieves name about devices paired by devices's address.
Arguments bd_addr : Input buffer to store 6bytes address
Return Value Paired device's name

2.1.26 setALSADevName

`void setALSADevName(const char *playback_avk, const char *playback_hs, const char *capture_hs, const char *playback_hs_sco, const char *capture_hs_sco)`
Description This function sets ALSA device name.
Arguments playback_avk : device for AVK playback. playback_hs : device for HS playback. capture_hs : device for HS capture. playback_hs_sco : device for HS SCO playback capture_hs_sco : device for HS SCO capture
Return Value void

2.1.27 startDiscovery

`int32_t startDiscovery(void)`
Description This function starts device discovery
Arguments void
Return Value 0 : Success, -1 : Fail

2.1.28 stopDiscovery

`int32_t stopDiscovery(void)`
Description

This function stops device discovery
Arguments void
Return Value 0 : Success, -1 : Fail

2.1.29 getDiscoveredDevCount

`int32_t getDiscoveredDevCount(void)`
Description This function gets the number of devices discovered.
Arguments void
Return Value The number of devices discovered

2.1.30 getDiscoveredDevInfoByIndex

`int32_t getDiscoveredDevInfoByIndex(int32_t device_index, char *name, unsigned char *bd_addr, unsigned char *class_of_device, char *class_name, int32_t *rssi)`
Description This function retrieves information about devices discovered by index.
Arguments device_index : Index of the discovered devices name : Input buffer to store name bd_addr : Input buffer to store 6bytes address class_of_device : Input buffer to store 3bytes class class_name : Input buffer to store class name
Return Value 0 : Success, -1 : Fail

2.1.31 bondDevice

`int32_t bondDevice(int32_t device_index)`
Description This function bonds discovered device.
Arguments device_index : Index of discovered device.
Return Value 0 : Success, -1 : Fail

2.1.32 cancelBondingDevice

`int32_t cancelBondDevice(int32_t device_index)`
Description This function cancels the bonding of the device being bonded.
Arguments device_index : Index of the device being bonded
Return Value 0 : Success, -1 : Fail

2.2 NXBT AVK service APIs

2.2.1 openAudioAVK

`int32_t openAudioAVK(void)`
Description This function opens the AVK ALSA audio device.
Arguments void
Return Value 0 : Succeed, -1 : Failed, -2 : HS audio is currently running

2.2.2 closeAudioAVK

`void closeAudioAVK(void)`
Description This function closes the AVK ALSA audio device.
Arguments void
Return Value void

2.2.3 isAudioStatusAVK

`bool isAudioStatusAVK(void)`
Description This function checks whether it is opened with AVK ALSA audio device.
Arguments void
Return Value True : Opened, False : Closed or Not opened

2.2.4 isConnectedAVK(void)

`bool isConnectedAVK(void)`
Description This function checks whether it is connected with AVK service.
Arguments void
Return Value True : Connected, False : Disconnected

2.2.5 connectToAVK

`int32_t connectToAVK(int32_t device_index)`
Description This function tries to connect to the AVK profile service.
Arguments device_index : Index of the paired devices
Return Value 0 : Success, -1 : Fail, -2 : Cancel

2.2.6 disconnectFromAVK

`int32_t disconnectFromAVK(unsigned char *bd_addr)`
Description This function tries to disconnect from AVK profile service.
Arguments bd_addr : Address of AVK connected device
Return Value 0 : Success, -1 : Fail

2.2.7 getConnectionNumberAVK

`int32_t getConnectionNumberAVK(void)`
Description This function gets the number of paired devices.
Arguments void
Return Value Number of AVK connected device

2.2.8 getConnectionDevAddrAVK

`int32_t getConnectionDevAddrAVK(int32_t connected_index, unsigned char *bd_addr)`
Description This function gets the address of AVK connected device by index.
Arguments connected_index : Index of AVK connected device bd_addr : Input buffer to store 6bytes address
Return Value 0 : Success, -1 : Fail

2.2.9 requestGetElementAttr

`int32_t requestGetElementAttr(unsigned char *bd_addr)`
Description This function requests for getting elements.
Arguments bd_addr : Address of AVK connected device
Return Value 0 : Success, -1 : Fail

2.2.10 requestPlayerValues

`int32_t requestPlayerValues(unsigned char *bd_addr)`
Description This function requests for player setting values.
Arguments bd_addr : Address of AVK connected device
Return Value 0 : Success, -1 : Fail

2.2.11 playStartAVK

`int32_t playStartAVK(unsigned char *bd_addr)`
Description This function starts audio playback.
Arguments bd_addr : Address of AVK connected device
Return Value 0 : Success, -1 : Fail

2.2.12 playStopAVK

`int32_t playStopAVK(unsigned char *bd_addr)`
Description This function stops audio playback.
Arguments bd_addr : Address of AVK connected device
Return Value 0 : Success, -1 : Fail

2.2.13 playPauseAVK

`int32_t playPauseAVK(unsigned char *bd_addr)`

Description This function pauses audio playback.
Arguments bd_addr : Address of AVK connected device
Return Value 0 : Success, -1 : Fail

2.2.14 playNextAVK

`int32_t playNextAVK(unsigned char *bd_addr)`
Description This function plays the next song.
Arguments bd_addr : Address of AVK connected device
Return Value 0 : Success, -1 : Fail

2.2.15 playPrevAVK

`int32_t playPrevAVK(unsigned char *bd_addr)`
Description This function plays the previous song.
Arguments bd_addr : Address of AVK connected device
Return Value 0 : Success, -1 : Fail

2.2.16 playEqualizerAVK

`int32_t playEqualizerAVK(unsigned char *bd_addr, unsigned char value)`
Description This function sets the equalizer mode.
Arguments bd_addr : Address of AVK connected device value : Equalizer mode
Return Value 0 : Success, -1 : Fail

2.2.17 playShuffleAVK

`int32_t playShuffleAVK(unsigned char *bd_addr, unsigned char value)`
Description

This function sets the shuffle mode.
Arguments bd_addr : Address of AVK connected device value : Shuffle mode
Return Value 0 : Success, -1 : Fail

2.2.18 playRepeatAVK

`int32_t playRepeatAVK(unsigned char *bd_addr, unsigned char value)`
Description This function sets the repeat mode.
Arguments bd_addr : Address of AVK connected device value : Repeat mode
Return Value 0 : Success, -1 : Fail

2.2.19 playScanAVK

`int32_t playScanAVK(unsigned char *bd_addr, unsigned char value)`
Description This function sets the scan mode.
Arguments bd_addr : Address of AVK connected device value : Scan mode
Return Value 0 : Success, -1 : Fail

2.3 NXBT HS service APIs

2.3.1 isConnectedHS

`bool isConnectedHS(void)`
Description This function checks whether it is connected with HS service.
Arguments void
Return Value True : Connected, False : Disconnected

2.3.2 requestCallIndicator

`int32_t requestCallIndicator(void)`
Description This function requests indicator call string.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.3 requestCurrentCalls

`int32_t requestCurrentCalls(void)`
Description This function requests current call string.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.4 getConnectionDevAddrHS

`int32_t getConnectionDevAddrHS(unsigned char *bd_addr)`
Description This function gets the address of HS connected device by index.
Arguments bd_addr : Address of HS connected device
Return Value 0 : Success, -1 : Fail

2.3.5 connectToHS

`int32_t connectToHS(int32_t device_index)`
Description This function tries to connect to the HS profile service.
Arguments device_index : Index of the paired devices
Return Value 0 : Success, -1 : Fail

2.3.6 disconnectFromHS

`int32_t disconnectFromHS(void)`
Description This function tries to disconnect from HS profile service.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.7 pickUpCall

`int32_t pickUpCall(void)`
Description This function picks up the call.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.8 hangUpCall

`int32_t hangUpCall(void)`
Description This function hangs up the call.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.9 openAudioHS

`int32_t openAudioHS(void)`

Description This function opens HS audio.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.10 closeAudioHS

`int32_t closeAudioHS(void)`
Description This function closes HS audio.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.11 isOpenedAudioHS

`bool isOpenedAudioHS(void)`
Description This function checks audio HS is opened.
Arguments void
Return Value True : Enabled, False : Disabled

2.3.12 muteMicrophoneHS

`void muteMicrophoneHS(bool mute)`
Description This function decides whether or not to mute microphone.
Arguments mute : True or false
Return Value void

2.3.13 isMutedMicrophoneHS

`bool isMutedMicrophoneHS(void)`
Description This function checks mic is muted.

Arguments void
Return Value True : Enabled, False : Disabled

2.3.14 dialPhoneNumber

`int32_t dialPhoneNumber(const char *number)`
Description This function tries to dial.
Arguments number : Destination phone number
Return Value 0 : Success, -1 : Fail

2.3.15 reDialPhoneNumber

`int32_t reDialPhoneNumber(void)`
Description This function tries to redial.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.16 setATCommandDTMF

`int32_t setATCommandDTMF(char key)`
Description This function sends AT command.
Arguments key : Dial keypad’s key
Return Value 0 : Success, -1 : Fail

2.3.17 requestCallNumber

`int32_t requestCallNumber(void)`
Description This function requests my phone call number.
Arguments void

Return Value
0 : Success, -1 : Fail

2.3.18 requestCallOperName

`int32_t requestCallOperName(void)`
Description This function requests the call operator's name.
Arguments void
Return Value 0 : Success, -1 : Fail

2.3.19 getCurrentBattChargingStatus

`int32_t getCurrentBattChargingStatus(void)`
Description This function gets battery status value.
Arguments void
Return Value Battery charging status value (0 ~ 5)

2.3.20 startVoiceRecognition

`int32_t startVoiceRecognition(void)`
Description This function starts voice recognition.
Arguments Void
Return Value 0 : Success, -1 : Fail

2.3.21 stopVoiceRecognition

`int32_t stopVoiceRecognition(void)`
Description This function stops voice recognition.
Arguments Void
Return Value 0 : Success, -1 : Fail

2.4 NXBT PBC service APIs

2.4.1 isConnectedPBC

`bool isConnectedPBC(void)`
Description This function checks whether it is connected with PBC service.
Arguments void
Return Value True : Connected, False : Disconnected

2.4.2 connectToPBC

`int32_t connectToPBC(int32_t device_index)`
Description This function tries to connect to the PBC profile service.
Arguments device_index : Index of the paired devices
Return Value 0 : Success, -1 : Fail

2.4.3 disconnectFromPBC

`int32_t disconnectFromPBC(void)`
Description This function tries to disconnect from the PBC profile service.
Arguments void
Return Value 0 : Success, -1 : Fail

2.4.4 abortPBC

`int32_t abortPBC(void)`
Description This function tries to abort the PBC profile service.
Arguments void
Return Value 0 : Success, -1 : Fail

2.4.5 getContactFromPBC

`int32_t getContactFromPBC(void)`
Description This function imports contacts from PBC profile service.
Arguments void
Return Value 0 : Success, -1 : Fail

2.4.6 getCallHistoryFromPBC

`int32_t getCallHistoryFromPBC(void)`
Description This function gets the call log from PBC profile service.
Arguments void
Return Value 0 : Success, -1 : Fail

2.5 NXBT MCE service APIs

2.5.1 isConnectedMCE

`bool isConnectedMCE(void)`
Description This function checks whether it is connected with MCE service.
Arguments void
Return Value True : Connected, False : Disconnected

2.5.2 connectToMCE

`int32_t connectToMCE(int32_t device_index)`
Description This function tries to connect to the MCE profile service.
Arguments device_index : Index of the paired devices
Return Value 0 : Success, -1 : Fail

2.5.3 disconnectFromMCE

`int32_t disconnectFromMCE(void)`
Description This function tries to disconnect from the MCE profile service.
Arguments device_index : Index of the paired devices
Return Value 0 : Success, -1 : Fail

2.5.4 abortMCE

`int32_t abortMCE(void)`
Description This function tries to abort the MCE profile service.
Arguments void
Return Value 0 : Success, -1 : Fail

2.5.5 startNotifyServerFromMCE

`int32_t startNotifyServerFromMCE(void)`
Description This function starts MNS(Message Notification Server) from the MCE profile service.
Arguments void
Return Value 0 : Success, -1 : Fail

2.5.6 stopNotifyServerFromMCE

`int32_t stopNotifyServerFromMCE(void)`
Description This function stops MNS(Message Notification Server) from the MCE profile service.
Arguments void
Return Value 0 : Success, -1 : Fail

2.5.7 getParserBmsg

`int32_t getParserBmsg(char *fullName, char *phoneNumber, char *msgBody)`
Description This function gets the parsed B message.
Arguments fullName : The sender phoneNumber : Sender’s phone number msgBody : Message content
Return Value 0 : Success, -1 : Fail

2.6 NXBT UI callback register functions

2.6.1 registerMGTOpenCbManager

```
void registerMGTOpenCbManager(void *pObj, void (*cbFunc)(void *, int32_t))
```

Description

Notify when MGT is successfully open.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Result
  - ➔ 0 : Success, -1 : Fail

Return Value

void

2.6.2 registerMGTDisconnectedCbManager

```
void registerMGTDisconnectedCbManager(void *pObj, void (*cbFunc)(void *))
```

Description

Notify when MGT is disconnected.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler

Return Value

void

2.6.3 registerDiscoveryCompleteCbManager

```
void registerDiscoveryCompleteCbManager(void *pObj, void (*cbFunc)(void *))
```

Description

Notify when discovery is complete.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler

Return Value

void

2.6.4 registerPairingFailedCbManager

```
void registerPairingFailedCbManager(void *pObj, void (*cbFunc)(void *, int32_t))
```

Description

Notify when pairing is failed.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Fail reason
  - ➔ 0x05 : Rejected

Return Value

void

2.6.5 registerPairedDevicesCbManager

```
void registerPairedDevicesCbManager(void *pObj, void (*cbFunc)(void *))
```

Description

Notify when paired device list is updated.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler

Return Value

void

2.6.6 registerUnpairedDevicesCbManager

```
void registerUnpairedDevicesCbManager(void *pObj, void (*cbFunc)(void *))
```

Description

Notify when device is unpaired.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler

Return Value

void

2.6.7 registerPairingRequestCbManager

```
void registerPairingRequestCbManager(void *pObj, void (*cbFunc)(void *, bool, char *,
unsigned char *, int32_t))
```

Description

Notify when receive the pairing request.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Automatic mode
- Device's name
- Device's address
- Pairing code (6 digits)

Return Value

void

2.6.8 registerLinkDownEventCbManager

```
void  registerLinkDownEventCbManager(void  *pObj,  void  (*cbFunc)(void  *,  unsigned  char
*,int32_t))
```

Description

Notify when receive the link down event.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Device's address
- Reason code
  - ➔ 0x08 : RF signal is disconnected
  - ➔ 0x13 : Turn off the BT module on the remote device or unpair the connected local device
  - ➔ 0x16 : Unpairing connected remote devices from the local device

Return Value

void

2.6.9 registerOpenFailedCbAVK

```
void registerOpenFailedCbAVK(void *pObj, void (*cbFunc)(void *))
```

Description

Notify when AVK open is failed.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
Return Value void

2.6.10 registerStreamingStartedCbAVK

`void registerStreamingStartedCbAVK(void *pObj, void (*cbFunc)(void *, bool))`
Description Notify when A2DP streaming is started.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - True : ALSA open succeed, False : ALSA open failed
Return Value void

2.6.11 registerStreamingStoppedCbAVK

`void registerStreamingStoppedCbAVK(void *pObj, void (*cbFunc)(void *))`
Description Notify when A2DP streaming is stopped.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler
Return Value void

2.6.12 registerConnectionStatusCbAVK

`void registerConnectionStatusCbAVK(void *pObj, void (*cbFunc)(void *, bool, char *, unsigned char *))`
Description Notify when AVK connection status is changed.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - Connection status - Device's name

- Device's address

Return Value

void

2.6.13 registerConnectionStatusCbAVKRC

```
void registerConnectionStatusCbAVKRC(void *pObj, void (*cbFunc)(void *, bool))
```

Description

Notify when AVKRC connection status is changed.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Connection status

Return Value

void

2.6.14 registerPlayStatusCbAVK

```
void registerPlayStatusCbAVK(void *pObj, void (*cbFunc)(void *, int32_t))
```

Description

Notify when play status is changed.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Play Status
  - ➔ 0x00 : Stopped
  - ➔ 0x01 : Playing
  - ➔ 0x02 : Paused

Return Value

void

2.6.15 registerMediaElementCbAVK

```
void registerMediaElementCbAVK(void *pObj, void (*cbFunc)(void *, char *, char *, char *,
char *, int32_t))
```

Description

Notify when media elements are updated.

Arguments

pObj	UI handler

cbFunc UI callback stub function

- Private handler
- Title
- Artist
- Album
- Genre
- Playing time (milliseconds)

Return Value

void

2.6.16 registerPlayPositionCbAVK

```
void registerPlayPositionCbAVK(void *pObj, void (*cbFunc)(void *, int32_t, int32_t))
```

Description

Notify when play position is updated.

Arguments

pObj UI handler

cbFunc UI callback stub function

- Private handler
- Play position (milliseconds)
- Play duration (milliseconds)

Return Value

void

2.6.17 registerPlayerValuesCbAVK

```
void  registerPlayerValuesCbAVK(void  *pObj,  void  (*cbFunc)(void  *,  int32_t,  int32_t,
int32_t, int32_t))
```

Description

Notify when player setting values are updated.

Arguments

pObj UI handler

cbFunc UI callback stub function

- Private handler
- Equalizer mode
  - 0x01 : Off
  - 0x02 : On
- Repeat mode
  - 0x01 : Off
  - 0x02 : Single
  - 0x03 : All
  - 0x04 : Group
- Shuffle mode

- 0x01 : Off
- 0x02 : All
- 0x03 : Group

- Scan mode

- 0x01 : Off
- 0x02 : All
- 0x03 : Group

Return Value

Void

2.6.18 registerListPlayerAttrCbAVK

```
void registerListPlayerAttrCbHS(void *pObj, void (*cbFunc)(void *, bool, bool, bool,
bool))
```

Description

Notify when player attribute list is received.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler

- Equalizer mode

- TRUE : Enabled, FALSE : Disabled

- Repeat mode

- TRUE : Enabled, FALSE : Disabled

- Shuffle mode

- TRUE : Enabled, FALSE : Disabled

- Scan mode

- TRUE : Enabled, FALSE : Disabled

Return Value

void

2.6.19 registerListPlayerValuesCbAVK

```
void registerListPlayerValuesCbHS(void *pObj, void (*cbFunc)(void *, int32_t, int32_t,
unsigned char *))
```

Description

Notify when player value list is received.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler

- Number of values

- Player attribute ID

- Equalizer : 0x01
- Repeat : 0x02
- Shuffle : 0x03
- Scan : 0cx04

- Values
Return Value void

2.6.20 registerOpenFailedCbHS

`void registerOpenFailedCbHS(void *pObj, void (*cbFunc)(void *))`
Description Notify when HS open is failed.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler
Return Value void

2.6.21 registerConnectionStatusCbHS

`void registerConnectionStatusCbHS(void *pObj, void (*cbFunc)(void *, bool, char *, unsigned char *))`
Description Notify when HS connection status is changed.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - Connection status - Device's name - Device's address
Return Value void

2.6.22 registerInbandRingSupportedCbHS

`void registerInbandRingSupportedCbHS(void *pObj, void (*cbFunc)(void *, bool))`
Description Notify that in-band ring is supported or not.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - Supported or not

Return Value

void

2.6.23 registerCallStatusCbHS

```
void registerCallStatusCbHS(void *pObj, void (*cbFunc)(void *, int32_t))
```

Description

Notify when call status is changed.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Call status
  - ➔ 0x00 : UNKNOWN_CALL
  - ➔ 0x01 : HANG_UP_CALL
  - ➔ 0x02 : INCOMMING_CALL
  - ➔ 0x03 : READY_OUTGOING_CALL
  - ➔ 0x04 : OUTGOING_CALL
  - ➔ 0x05 : PICK_UP_CALL
  - ➔ 0x06 : DISCONNECTED_CALL

Return Value

void

2.6.24 registerBatteryStatusCbHS

```
void registerBatteryStatusCbHS(void *pObj, void (*cbFunc)(void *, int32_t))
```

Description

Notify when the battery status changes or when the value is requested.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Battery charging status value (0 ~ 5)

Return Value

void

2.6.25 registerCallOperNameCbHS

```
void registerCallOperNameCbHS(void *pObj, void (*cbFunc)(void *, char *))
```

Description

Notify when the operator'name is requested.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Call operator's name

Return Value

void

2.6.26 registerCurrentCallIsCbHS

```
void registerCurrentCallIsCbHS(void *pObj, void (*cbFunc)(void *, char *))
```

Description

Notify when receive the CLCC event.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- ➔ Private handler
- ➔ CLCC string
  - <idx>,<dir>,<status>,<mode>,<mprty>[,<number>,<type>]
    - <idx>
      - The numbering (starting with 1) of the call given by the sequence of setting up or receiving the calls (active, held or waiting) as seen by the served subscriber
    - <dir>
      - 0 (outgoing), 1 (incoming)
    - <status>
      - 0 = Active
      - 1 = Held
      - 2 = Dialing (outgoing calls only)
      - 3 = Alerting (outgoing calls only)
      - 4 = Incoming (incoming calls only)
      - 5 = Waiting (incoming calls only)
    - <mode>
      - 0 (Voice), 1 (Data), 2 (FAX)
    - <mprty>
      - 0 (Not Multiparty), 1 (Multiparty)
    - <number> – (optional)
      - Phone number
    - <type> – (optional)
      - values 128-143 : The phone number format may be a national or international format, and may contain prefix and/or escape digits
      - values 144-159 : The phone number format is an international number, including the country code prefix
      - values 160-175 : National number. No prefix nor escape digits included

Return Value

void

2.6.27 registerCallNumberCbHS

`void registerCallNumberCbHS(void *pObj, void (*cbFunc)(void *, char *))`
Description Notify when my phone call number is requested.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - My phone call number
Return Value void

2.6.28 registerAudioMuteStatusCbHS

`void registerAudioMuteStatusCbHS(void *pObj, void (*cbFunc)(void *, bool, bool))`
Description Notify when audio HS and mic mute status are changed.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - Mute status - Audio HS status
Return Value void

2.6.29 registerVoiceRecognitionStatusCbHS

`void registerVoiceRecognitionStatusCbHS(void *pObj, void (*cbFunc)(void *, unsigned short))`
Description Notify when voice recognition status is changed.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - Voice recognition status
Return Value void

2.6.30 registerIncommingCallNumberCbHS

`registerIncommingCallNumberCbHS(void *pObj, void (*cbFunc)(void *, char *))`

Description

Notify the phone number when is incomming call.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- ➔ Private handler
- ➔ CLIP string
  - ■ CLIP string structure (Calling line identification notification)
    - ◆ <number>, type>
      - ● <number>
        - ■ Phone number
      - ● <type>
        - ■ values 128-143: The phone number format may be a national or international format, and may contain prefix and/or escape digits
        - ■ values 144-159: The phone number format is an international number, including the country code prefix
        - ■ values 160-175: National number. No prefix nor escape digits included

Return Value

void

2.6.31 registerCallIndicatorCbHS

`void registerCallIndicatorCbHS(void *pObj, void (*cbFunc)(void *, char *))`

Description

Notify when receive the CIND event.

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- ➔ Private handler
- ➔ CIND string
- ➔ CIND string structure (Call indicator)
  - ■ <ind>,<value>
    - ◆ <ind> : Order index of the indicator within the list retrieved from the AG with the AT+CIND=? command. The first element of the list shall have <ind>=1
    - ◆ <value> : Current status of the indicator
      - ● call
        - ■ 0 means there are no calls in progress
        - ■ 1 means at least one call is in progress
      - ● callheld
        - ■ 0 No calls held
        - ■ 1 Call is placed on hold or active/held calls swapped
        - ■ 2 Call on hold, no active call

- callsetup
  - 0 means not currently in call set up
  - 1 means an incoming call process ongoing
  - 2 means an outgoing call set up is ongoing
  - 3 means remote party being alerted in an outgoing call
- service
  - 0 implies no service. No Home/Roam network available
  - 1 implies presence of service. Home/Roam network available
- signal
  - Ranges from 0 to 5
- roam
  - 0 means roaming is not active
  - 1 means a roaming is active
- battchg
  - Ranges from 0 to 5

➔ Example
- 1st : ("call",(0,1)),("callsetup",(0-3)),("service",(0-1)),("signal",(0-5)), ("roam",(0,1)),("battchg",(0-5)),("callheld",(0-2))
- 2st : 0,0,1,5,0,4,0

Return Value

void

2.6.32 registerCallIndicatorParsingValuesCbHS

```
void registerCallIndicatorParsingValuesCbHS(void *pObj, void (*cbFunc)(void *, int32_t,
int32_t, int32_t, int32_t, int32_t, int32_t, int32_t))
```

Description

Notify when receive the CIND event. (Parsing values)

Arguments

pObj	UI handler
cbFunc	UI callback stub function

- Private handler
- Service
- callind
- call_setup
- roam
- signal_strength
- battery

Return Value

void

2.6.33 registerOpenFailedCbPBC

```
void registerOpenFailedCbPBC(void *pObj, void (*cbFunc)(void *))
```

Description

Notify when PBC open is failed.

Arguments pObj UI handler cbFunc UI callback stub function - Private handler
Return Value void

2.6.34 registerConnectionStatusCbPBC

`void registerConnectionStatusCbPBC(void *pObj, void (*cbFunc)(void *, bool))`
Description Notify when PBC connection status is changed.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - Connection status
Return Value void

2.6.35 registerNotifyGetPhoneBookCbPBC

`void registerNotifyGetPhoneBookCbPBC(void *pObj, void (*cbFunc)(void *))`
Description Notify when contact or call log is received. It is created as ‘pb_data.vcf’ file in “/etc/bluetooth/"
Arguments pObj UI handler cbFunc UI callback stub function - Private handler
Return Value void

2.6.36 registerListDataCbPBC

`void registerListDataCbPBC(void *pObj, void (*cbFunc)(void *, unsigned char *))`
Description Notify when list message is received.
Arguments pObj UI handler cbFunc UI callback stub function

- Private handler - List data
Return Value void

2.6.37 registerOpenFailedCbMCE

`void registerOpenFailedCbMCE(void *pObj, void (*cbFunc)(void *))`
Description Notify when MCE open is failed.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler
Return Value void

2.6.38 registerConnectionStatusCbMCE

`void registerConnectionStatusCbMCE(void *pObj, void (*cbFunc)(void *, bool))`
Description Notify when MCE connection status is changed.
Arguments pObj UI handler cbFunc UI callback stub function - Private handler - Connection status
Return Value void

2.6.39 registerNotifyGetMessageCbMCE

`void registerNotifyGetMessageCbMCE(void *pObj, void (*cbFunc)(void *))`
Description Notify when SMS message is received. It is created as ‘get_msg.txt’ file in “/etc/bluetooth/"
Arguments pObj UI handler cbFunc UI callback stub function - Private handler

Return Value
void