BLUETOOTH ANDROID APPLICATION

Installing Command:

adb -s 2b51875525057ece install -t -r -g app-debug.apk

To Launch the application:

adb -s 2b51875525057ece shell am start -n bluetooth.app.bluetooth_application/.MainActivity

To Call the apis:

adb_broadcast_cmd =

adb -s {} shell am broadcast -a bluetooth.app.bluetooth_application.receiveArgument --es methodName {}

adb_broadcast_cmd_with_arg = adb -s {} shell am broadcast -a bluetooth.app.bluetooth_application.receiveArgument --es methodName {} --es arguments {}

Date : 03/01/2024

1) API Name: turnOnBluetooth()

Description: Turns on the Bluetooth based on its current state.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name : turnOnBluetooth

Arguments : None

Event To Be Verified:

isEnabled: Bluetooth Turned On / Bluetooth Already Turned On

2) API Name: turnOffBluetooth()

Description: Turns off the Bluetooth based on its current state.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name : turnOffBluetooth

Arguments : None

Event To Be Verified:

isEnabled: Bluetooth Turned Off / Bluetooth Already Turned Off

3) API Name: bluetoothScan()

Description: Initiates a Bluetooth scan for nearby devices.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter): bluetoothScan

Event To Be Verified:

startDiscoveryProcess: The startDiscoveryProcess returns true

MyReceiver: Discovered device: {deviceName}, {deviceAddress}

4) API Name: scanLeDevice()

Description: Initiates a Bluetooth Low Energy (LE) scan for nearby devices.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

If ref is bluez: API Name: scanLeDevice
Arguments: scanPeriod-> The LE Scan Timeout

If ref is android: API Name: scanLeDevice
Arguments: scanPeriod, (android) refName

Event To Be Verified:

Scanning: LE Scan Started

onLeScanResult**:** Discovered device: {deviceName}, {deviceAddress}

5) API Name: leConnect()

Description: Establish a le connection with the specified remote device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

If ref is bluez: API Name: leConnect
Arguments: address

If ref is android: API Name: leConnect
Arguments: None

Event To Be Verified:

connectionState: Connected to GATT server. Starting service discovery. 0

6) API Name: leAdvertise()

Description: Start advertising the device using Bluetooth LE.

Input Parameters: None

Possible Argument(to be passed as runtime parameter):

API Name: leAdvertise
Arguments: None

Event To Be Verified: startAdvertisement: Advertisement Started

7) API Name: addBleService()

Description:

Adds a Bluetooth GATT service and its characteristics , descriptors with the specified service name.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: addBleService
Arguments: serviceName

Event To Be Verified:

addService: {serviceName} Service Added.

8) API Name: gattReadOperation()

Description:

Performs Bluetooth gatt read operations(read characteristics and descriptors)

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: gattReadOperation
Arguments: serviceName-characteristicName-operation(readCharacter/readDescriptor)

Event To Be Verified:

(Based on the Operation)

For read character:

readCharacteristic: READ STATUS {status} /

readCharacteristic: ACTION_DATA_READ_VALUE: {value}

readCharacteristic: **ACTION_DATA_READ**

For read descriptor:
readDescriptor: ACTION_DATA_READ_DESCRIPTOR_VALUE: {value}

readDescriptor: **ACTION_DESCRIPTOR_READ**

9) API Name: disconnect()

Description: Disconnect the Bluetooth LE connection.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: disconnect
Arguments: remoteAddress

Event To Be Verified:

connectionStateAfterDisconnect: Connection state: 0/

getConnectedDevices: null/

connectionState: Disconnected from GATT server. Status: 0

10) API Name: stopScanning()

Description:

Stops a Bluetooth Low Energy (LE) scan.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: stopScanning
Arguments: None

Event To Be Verified:

stopScanning: LE Scan Stopped

11) API Name: readCharacteristic()

Description:

Reads a BluetoothGattCharacteristic from the remote device.

Input Parameters:

characteristic -> The BluetoothGattCharacteristic to be read.

Possible Argument(to be passed as a runtime parameter): None (internally called)

12) API Name: readDescriptor()

Description:

Reads a BluetoothGattDescriptor from the remote device.

Input Parameters:

descriptor -> The BluetoothGattDescriptor to be read.

Possible Argument(to be passed as a runtime parameter): None (internally called)

13) API Name: writeCharacteristic()

Description:

Writes a BluetoothGattCharacteristic from the remote device.

Input Parameters:

characteristic -> The BluetoothGattCharacteristic to be written.

Possible Argument(to be passed as a runtime parameter): None (internally called)

14) API Name: writeDescriptor()

Description:

writes a BluetoothGattDescriptor from the remote device.

Input Parameters:

descriptor -> The BluetoothGattDescriptor to be write.

Possible Argument(to be passed as a runtime parameter): None (internally called)

Date : 04/01/2024

15) API Name: getAddressOfDevice()

Description: Returns the hardware address of the local Bluetooth adapter.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getAddressOfDevice
Arguments: None

Event To Be Verified:

getLocalAddress: The Address Of The Device Is {address}

16) API Name: getAdapter()

Description: Get a handle to the default local Bluetooth adapter.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getAdapter
Arguments: None

Event To Be Verified:

getAdapter: The Default Adapter Is {btAdapter}

17) API Name: getRemoteAddress()

Description: Get the hardware address of the remote device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getRemoteAddress
Arguments: remoteDeviceAddress

Event To Be Verified:

getRemoteAddress: The Address Of The Remote Device Is {refDeviceAddress}

18) API Name: stopAdvertisement()

Description: Stop the ongoing advertisement.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: stopAdvertisement
Arguments: None

Event To Be Verified:

stopAdvertisement: Advertisement Stopped

19) API Name: getRemoteClass()

Description: Get the Bluetooth class of the remote device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getRemoteClass
Arguments: remoteDeviceAddress

Event To Be Verified:

getRemoteDeviceClass: The Class Of The Remote Device Is {refBtClass}

20) API Name: getRemoteBondState()

Description: Get the bond state of the remote device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getRemoteBondState
Arguments: remoteDeviceAddress

Event To Be Verified:

getRemoteBondState: The Bond State Of The Remote Device Is {refBondState}

21) API Name: getRemoteName()

Description: Get the friendly Bluetooth name of the remote device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getRemoteName
Arguments: remoteDeviceAddress

Event To Be Verified:

getRemoteName: The Name Of The Remote Device Is {refDevicename}

22) API Name: getRemoteType()

Description: Get the Bluetooth device type of the remote device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getRemoteType
Arguments: remoteDeviceAddress

Event To Be Verified:

getRemoteType: The Type Of The Remote Device Is {refBtType}

23) API Name: getRemoteUuids()

Description: Returns the supported features (UUIDs) of the remote device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getRemoteUuids
Arguments: remoteDeviceAddress

Event To Be Verified:

getRemoteUuids: The Supported uuids Of The Remote Device Is {refDeviceUuids}

24) API Name: cancelDiscovery()

Description: Cancel the currents device discovery process.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: cancelDiscovery
Arguments: None

Event To Be Verified:

cancelDiscovery: The Status of cancel discovery Is true

25) API Name: startDiscoveryProcess()

Description: Start the remote device discovery process.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: startDiscoveryProcess
Arguments: None

Event To Be Verified:

startDiscoveryProcess: The startDiscoveryProcess returns true

26) API Name: checkBluetoothState()

Description: Return true if Bluetooth is currently enabled and ready for use.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: checkBluetoothState
Arguments: None

Event To Be Verified:

checkBluetoothState: The state of Bluetooth {state}

27) API Name: checkScanState()

Description: Return true if the local Bluetooth adapter is currently in the device discovery process.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: checkScanState
Arguments: None

Event To Be Verified:

checkScanState: The state of bluetooth scan {scanState}

28) API Name: getNameOfBluetooth()

Description: Get the friendly Bluetooth name of the local Bluetooth adapter.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getNameOfBluetooth
Arguments: None

Event To Be Verified:

getNameOfBluetooth: The name of Bluetooth {name}

29) API Name: bondedDevices()

Description: Return the set of BluetoothDevice objects that are bonded (paired) to the local adapter.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: bondedDevices
Arguments: None

Event To Be Verified:

bondedDevices: The bonded devices {listOfDevices}

30) API Name: getScanMode()

Description: Get the current Bluetooth scan mode of the local Bluetooth adapter.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getScanMode
Arguments: None

Event To Be Verified:

getScanMode: The scan mode of bluetooth {scanMode}

31) API Name: getState()

Description: Get the current state of the local Bluetooth adapter.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getState
Arguments: None

Event To Be Verified:

getState: The state of Bluetooth {state}

Date : 09/01/2024

32) API Name: getRemoteDeviceInstance()

Description: Get the Bluetooth device type of the remote device.

Input Parameters: address -> The Bluetooth address of the remote device.

Possible Argument(to be passed as a runtime parameter): None(internally called)

Event To Be Verified:

getRemoteDeviceInstance: deviceAddress: {address}

33) API Name: pairDevice()

Description:

Initiates the pairing process with a remote Bluetooth device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: pairDevice
Arguments: remoteAddress

Event To Be Verified:

pairDevice: Pairing initiated successfully

bondReceiver: Bond state changed for device {btDeviceaddress}: BOND_BONDED

34) API Name: profileConnection()

Description:

Initiates the connection to a Bluetooth profile.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: profileConnection
Arguments: profileName, remoteAddress

- A2DP-remoteAddress
- HEADSET-remoteAddress

Event To Be Verified:

onServiceConnected: Bluetooth profile connected: 2

connectProfile: Connection result: true

35) API Name: connectProfile()

Description:

Invokes the "connect" method on the Bluetooth profile using reflection.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter): None (internally called)

Event To Be Verified:

connectProfile: Connection result: true

Date : 10/01/2024

36)) API Name: audioStream()

Description:

Plays an audio file using MediaPlayer.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: audioStream
Arguments: Filepath of the audio.

Event To Be Verified:

Current Playing Audio: {indexOfListOfSongs}

isMediaPlaying: isPlaying returns true

37) API Name: getConnDevices()

Description: Retrieves a list of Bluetooth devices that are currently connected to the profile.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getConnDevices
Arguments: None

Event To Be Verified:

getConnectedDevices: {deviceList}

38) API Name: getProfileConnState()

Description: Retrieves the connection state of the Bluetooth profile

associated with the remote device.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getProfileConnState
Arguments: remoteDevice

Event To Be Verified:

getProfileConnectionState associated with the remote device: 2

(2 -> for connected, 1 -> connecting, 3 -> disconnecting, 0-> disconnected)

39) API Name: profileConnectionState()

Description: Retrieves the connection state of a specific Bluetooth profile.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: profileConnectionState
Arguments: profileType

Event To Be Verified:

geProfileConnectionState: 2

40) API Name: maxConnectedAudioDevices()

Description: Retrieves the maximum number of connected audio devices supported.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: maxConnectedAudioDevices
Arguments: None

Event To Be Verified:

getMaxConnectedAudioDevices: {number of devices}

41) API Name: getBtLeScanner()

Description: Retrieves the Bluetooth LE scanner instance.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getBtLeScanner
Arguments: None

Event To Be Verified:

getBluetoothLeScanner: {leScannerObject}

42) API Name: getBtLeAdvertiser()

Description: Retrieves the Bluetooth LE advertiser instance.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getBtLeAdvertiser
Arguments: None

Event To Be Verified:

getBluetoothLeAdvertiser: {leAdvertiserObject}

Date : 12/01/2024

43) API Name: disconnectProfile()

Description: Disconnects to the Bluetooth profile.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: disconnectProfile
Arguments: address

Event To Be Verified:

disconnectProfile: disconnectionResult : true

44) API Name: enableDiscoverableMode()

Description: Enables the discoverable mode of the Bluetooth adapter.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: enableDiscoverableMode
Arguments: None

Event To Be Verified:

isDiscoverable: Discoverable Mode Changed: On

45) API Name: disableDiscoverableMode()

Description: Disables the discoverable mode of the Bluetooth adapter.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: disableDiscoverableMode
Arguments: None

Event To Be Verified:

isDiscoverable: Discoverable Mode Changed: Off

46) API Name: unPairDevice()

Description: Unpairs the Bluetooth device with the specified remote address.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: unPairDevice
Arguments: None

Event To Be Verified:

bondReceiver: Bond state changed for device {address}: BOND_NONE

47) API Name: getDiscoveryTimeout()

Description: Retrieves the discoverable timeout for Bluetooth device discovery.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getDiscoveryTimeout
Arguments: None

Event To Be Verified:

getDiscoveryTimeout: {timeout}

48) API Name: getBondedDevices()

Description: Retrieves a set of all currently bonded Bluetooth devices.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getBondedDevices
Arguments: None

Event To Be Verified:

getBondedDevices: {listOfBondedDevices}

49) API Name: getBondState()

Description: Retrieves the bond state of a Bluetooth device with the specified address.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getBondState
Arguments: remoteAddress

Event To Be Verified:

getBondStat: {bondState}

Date : 17/01/2024

50) API Name: audioStart()

Description:

Starts playing the audio using the MediaPlayer.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: audioStart
Arguments: None

Event To Be Verified:

audioStart: Audio Playing Started

51) API Name: audioPause()

Description:

Pauses the currently playing audio.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: audioPause
Arguments: None

Event To Be Verified:

audioPause: Audio Playing Paused

52) API Name: audioStop()

Description:

Stops the currently playing audio.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: audioStop
Arguments: None

Event To Be Verified:

audioStop: Audio Playing Stopped

53) API Name: audioPrepare()

Description:

Prepares the MediaPlayer for playback.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: audioPrepare
Arguments: None

Event To Be Verified:

audioPrepare: Audio Playing Prepared

54) API Name: isAudioLooping()

Description:

Checks if the audio is set to loop. Returns True if audio is set to loop, false otherwise.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: isAudioLooping
Arguments: None

Event To Be Verified:

isAudioLooping: {true/false}

55) API Name: setAudioVolume()

Description:

Sets the audio volume.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: setAudioVolume
Arguments: None

Event To Be Verified:

setAudioVolume: Set Audio Volume {volume}

56) API Name: getAudioPlaybackPosition()

Description:

Retrieves the current playback position of the audio.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getAudioPlaybackPosition
Arguments: None

Event To Be Verified:

getAudioPlaybackPosition: current position {position}

57) API Name: getAudioDuration()

Description:

Retrieves the total duration of the audio.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getAudioDuration
Arguments: None

Event To Be Verified:

getAudioDuration: {duration}

58) API Name: seekToPosition()

Description:

Seeks to a specific position in the audio.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: seekToPosition
Arguments: None

Event To Be Verified:

seekToPosition: current position {position}

59) API Name: releaseMediaPlayer()

Description:

Releases the resources used by the MediaPlayer.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter): None (internally called)

Event To Be Verified:

releaseMediaPlayer: Media Player Released

60) API Name: isMediaPlaying()

Description:

Checks if the media is currently playing.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: isMediaPlaying
Arguments: None

Event To Be Verified:

isMediaPlaying: isPlaying returns true

Date : 22/01/2024

61) API Name: nextAudio()

Description:

Plays the next audio file in the playlist.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: nextAudio
Arguments: None

Event To Be Verified:

Current Playing Audio: {indexOfListOfSongs}

62) API Name: previousAudio()

Description:

Plays the previous audio file in the playlist.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: previousAudio
Arguments: None

Event To Be Verified:

Current Playing Audio: {indexOfListOfSongs}

63) API Name: getAudioList()

Description:

Retrieves a list of audio files in the specified folder.

Input Parameters:

FolderPath -> The path of the folder containing audio files.

Possible Argument(to be passed as a runtime parameter): None (internally called)

Event To Be Verified:

audioList and size: {listOfFiles}, {size}

64) API Name: createMediaPlayerObject()

Description:

Creates a MediaPlayer object for the specified audio file.

Input Parameters:

audio -> The file path of the audio file for which a MediaPlayer

object is created.

Possible Argument(to be passed as a runtime parameter): None (internally called)

Date : 30/01/2024

65) API Name: createScoConnection()

Description:

Establishes a Bluetooth SCO (Secure Communications) connection.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: createScoConnection
Arguments: None

Event To Be Verified:

createScoConnection: SCO Connection Established.

66) API Name: stopScoConnection()

Description:

Stops the established Bluetooth SCO (Secure Communications) connection.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: stopScoConnection
Arguments: None

Event To Be Verified:

stopScoConnection: SCO Connection Disconnected.

67) API Name: unPairAll()

Description:

Unpairs all the Bluetooth devices from bonded devices list.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: unPairAll
Arguments: None

Event To Be Verified:

(Same as unPairDevice and should validate through getBondedDevice())

Date : 31/01/2024

68) API Name: startRecording()

Description:

Starts audio recording to the specified file path.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: startRecording
Arguments: outputFilepath

Event To Be Verified:

audioRecorder: Audio Recording Started

69) API Name: stopRecording()

Description:

Stops the ongoing audio recording and releases resources.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: stopRecording
Arguments: None

Event To Be Verified:

audioRecorder: Audio Recording Stopped

70) API Name: getAudioVolume()

Description: Get the volume of the current playing audio.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getAudioVolume
Arguments: None

Event To Be Verified:

getStreamVolume: {volumeLevel}

71) API Name: sendResponse()
Description: Sends the response on receiving the read characteristic request.(Used only for android14)

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: sendResponse
Arguments: address

Event To Be Verified:

sendResponse: Response Sent

Date : 18/03/2024

71) API Name: btFileTransfer()
Description: Calls the method to share a file via Bluetooth.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: btFileTransfer
Arguments: path of the file to be shared.

Event To Be Verified:

sendResponse: (Yet to be added.)

Date: 02/04/2024

72) API Name: gattWriteOperation()
Description:

Performs Bluetooth gatt write operations(write characteristics and descriptors)

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: gattWriteOperation
Arguments: serviceName-characteristicName-operation(writeCharacter/writeDescriptor)-value

Event To Be Verified:

(Based on the Operation)

For write character:

writeCharacteristic : **WRITE CHARACTERISTIC SUCCESSFUL** /

writeCharacteristic: **ACTION_DATA_WRITTEN**

For write descriptor:

writeDescriptor: **WRITE DESCRIPTOR SUCCESSFUL**

writeDescriptor: **ACTION_DATA_WRITTEN**

73) API Name: removeBleService()

Description:

Removes a Bluetooth GATT service and its characteristics , descriptors with the specified service name.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: removeBleService
Arguments: serviceName

Event To Be Verified:

removeBleService: {serviceName} Service Removed Successfully.

75) API Name: getAvailableServices()

Description:

Returns the list of available services in bluetooth gatt server.

Input Parameters: None

Possible Argument(to be passed as a runtime parameter):

API Name: getAvailableServices
Arguments: None

Event To Be Verified:

getAvailableServices: The Available Services are {list}.