Communication API Programming Guide for iOS

Version: V1.2.4.1222

22-Dec-2013

Evolution follow-up

Revision	Type of modification	Author	Date
1.0.0	Creation	Huang Guoqiang	19/03/2013
1.0.1	1.Fix the description of void closeDevice()	Huang Guoqiang	29/03/2013
1.0.2	1.modify **–(int)**cancelExchange from Blocking way to Non-blocking and read result via callback function **–(void)**onError:(int) code Param:(NSString*) msg. 2.add Error Code TRANS_ERROR_USER_CANCEL_SUCCESS and TRANS_ERROR_USER_CANCEL_FAILED. 3.fix diagrams Phone State machine and Normal case. 4.add +(NSString*)getLibVersion to get the lib version. 4.add Exception case when application send reset-pinpad command.	Huang Guoqiang	07/05/2013
1.1.0	1.add CommunicationManagerBase to merge BluetoothManager and AudioJackManager Interface. You can use BluetoothManager or AudioJackManager independently or use CommunicationManagerBase to indicate BluetoothManager or AudioJackManager. 2.use DeviceSearchListener instead of BluetoothSearchListener that can return Devices include Bluetooth or a generic audio jack(maybe there isn't a rp350x or rp750x, just a headset) 3.add API calling procedure for using CommunicationManagerBase	Junhua Wu	25/08/2013
1.1.1	1. fix bug about audioqueue flags. 2. fix bug about shutdown device failure. 3. reduce volume control dialog display frequency	Junhua Wu	10/10/2013
1.1.2	1. modify the search condition for AUDIOJACK	Junhua Wu	10/11/2013
1.1.3	1. fix bug about openDevice return -2 2. fix bug about exchangeData with large data 3. fix bug about filtering some peripheral while their UUID is nil.	Junhua Wu	29/11/2013
1.2.2.1212	1. fix bug for BluetoothManager. 2. Add function about connect to remote device directly for BluetoothManager 3. Update driver protocol for BluetoothManager. 4. Realize two modes of interface for BluetoothManager.	Junhua Wu	12/12/2013

1.2.4.1222	1. Add new methods for accessing connect state. 2. Update multiple protocol support.	Junhua Wu	22/12/2013

Table of Contents

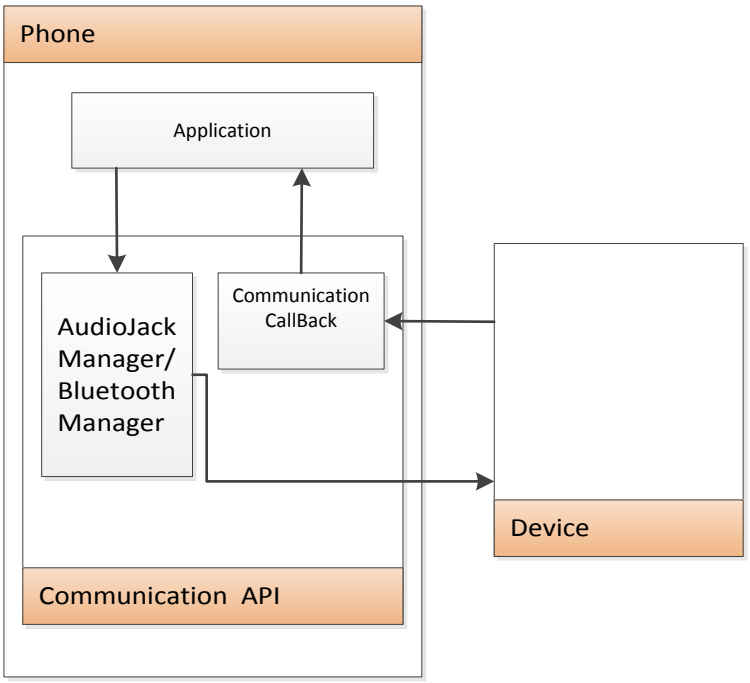
Over View

The API is designed for developer to integrate CommunicationManagerBase functionality into iOS applications. It is distributed as a static library.
The programmatic interface for Communication API is expressed as three classes CommunicationManagerBase , AudioJackManager, BluetoothManager, and two protocol CommunicationCallBack, DeviceSearchListener. BluetoothManager and AudioJackManager is extends CommunicationManagerBase. Developer have to create an instance of class and implement all the interface methods.

System Requirements
• iOS 5.0 or later

Phone and Device State machine

The AudioJackManager API is used to communicate with device through audiojack, the following diagram provides a brief overview of the interaction between API objects, device and your application.

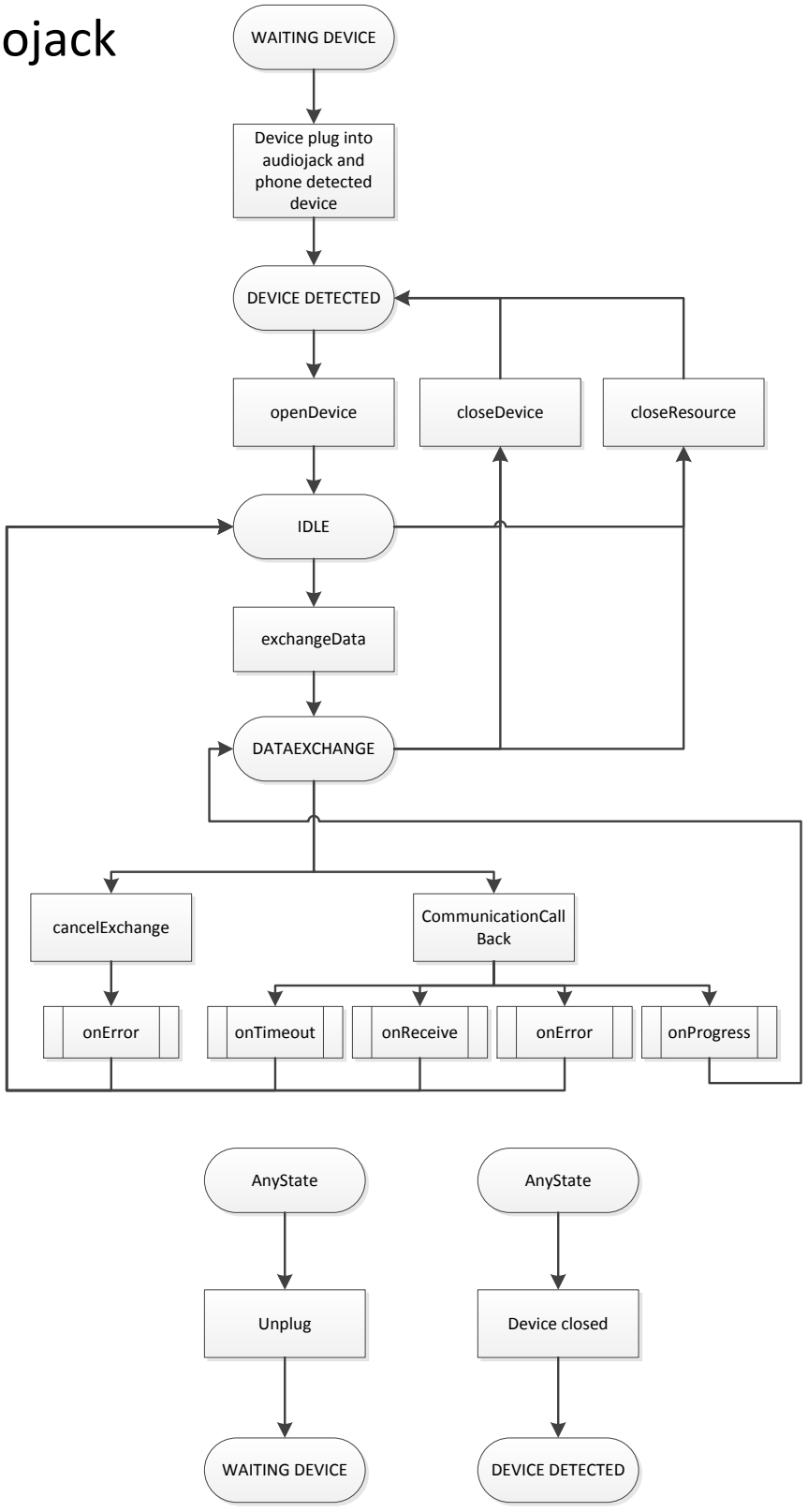


Phone State machine

WAITING DEVICE	Device is not connected or open Bluetooth, phone is waiting device.
DEVICE DETECTED	Phone has detected device by audio jack or Bluetooth.
IDLE	Phone and device is not exchanging data , but can exchange data with device anytime.
DATAEXCHANGE	Phone is exchanging data with device.

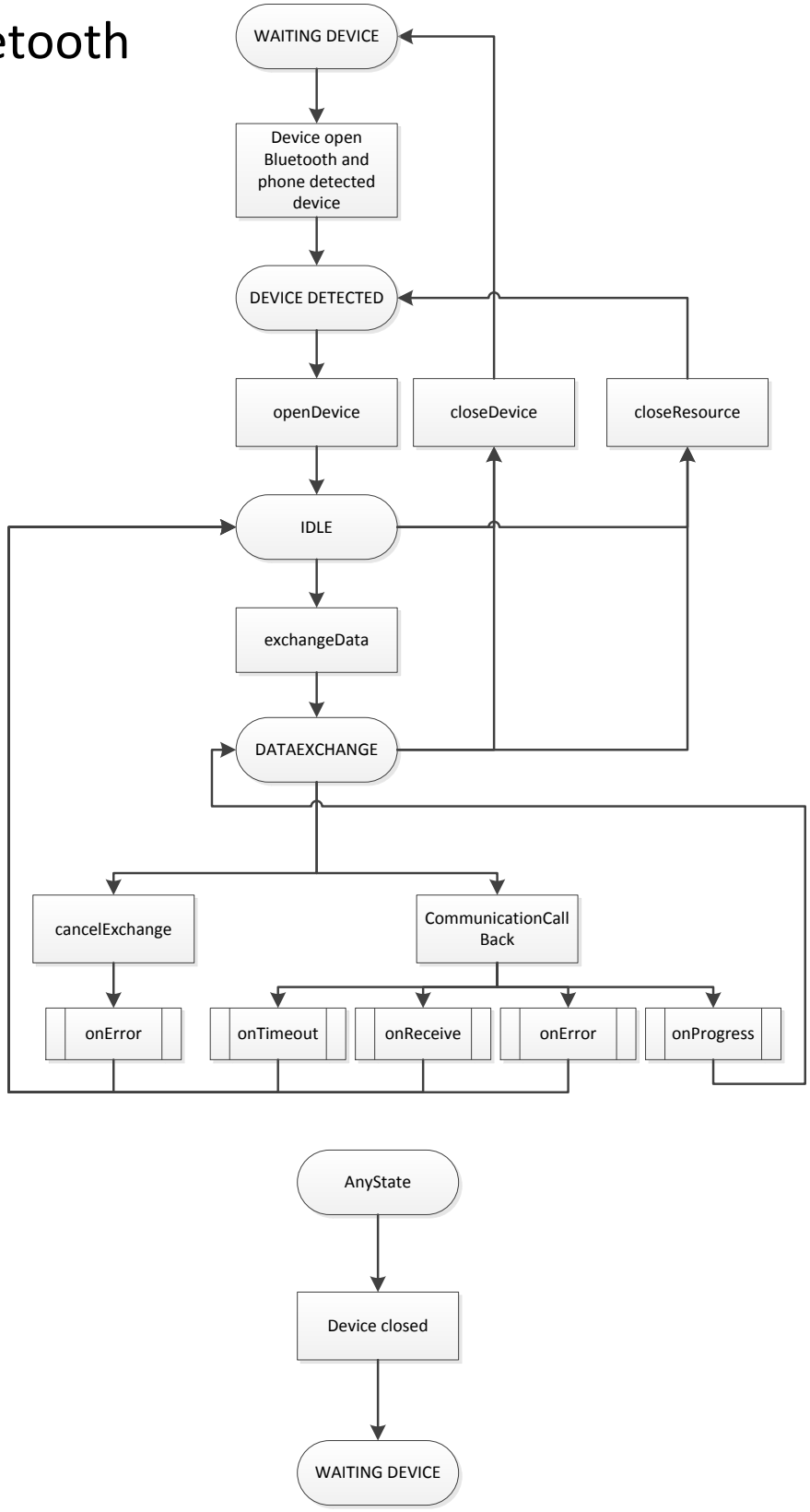
The following diagrams show the life cycle and the state of phone.

Phone State for Audiojack



phone state machine for audio jack

Phone State for Bluetooth



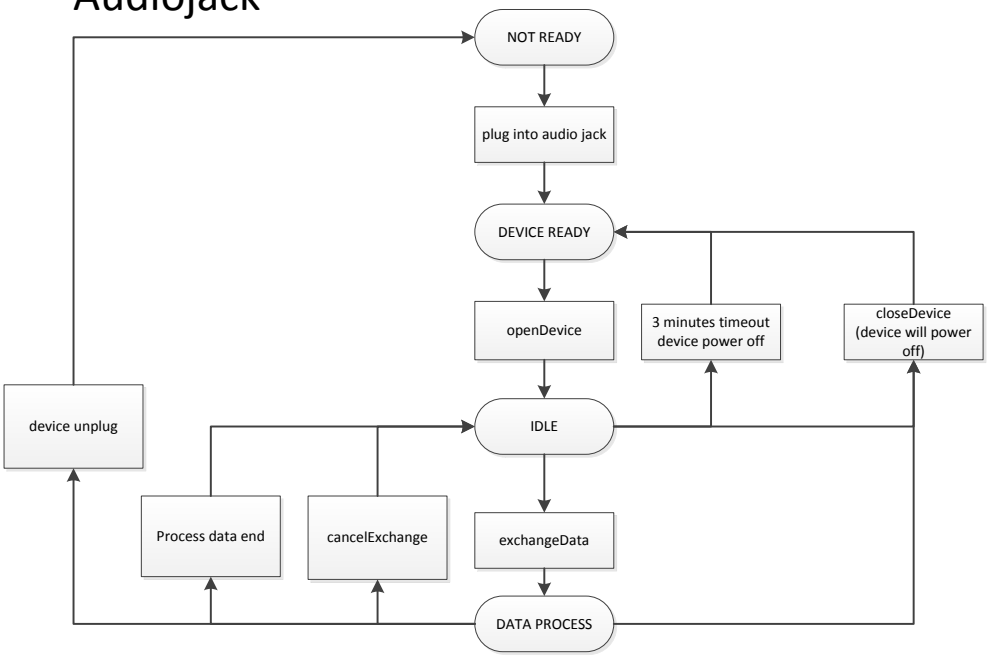
phone state machine for Bluetooth

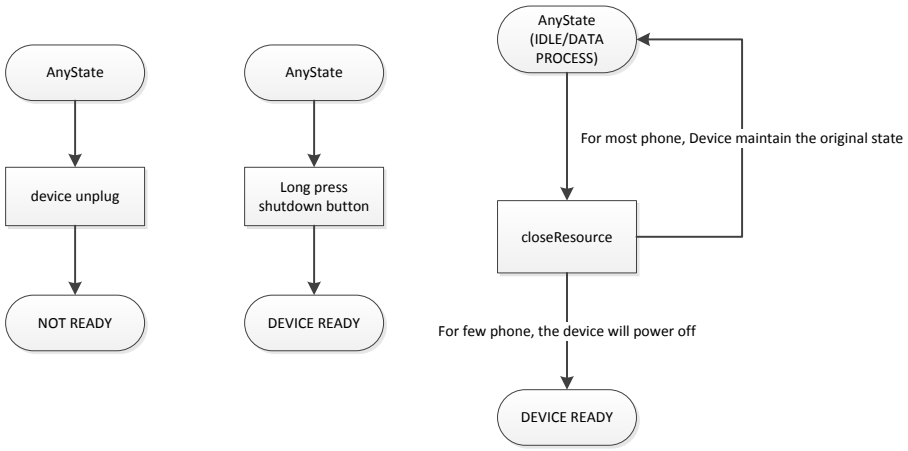
Device State machine

NOT READY	Device isn’t plug into audio jack or Device isn’t open Bluetooth.
DEVICE READY	Device is connected the phone by audio jack or Device has opened Bluetooth.
IDLE	Device is power on, establish connection with phone and ready to receive the data.
DATA PROCESS	Device is working including receiving and processing data.

The following diagrams show the life cycle and the states of device.

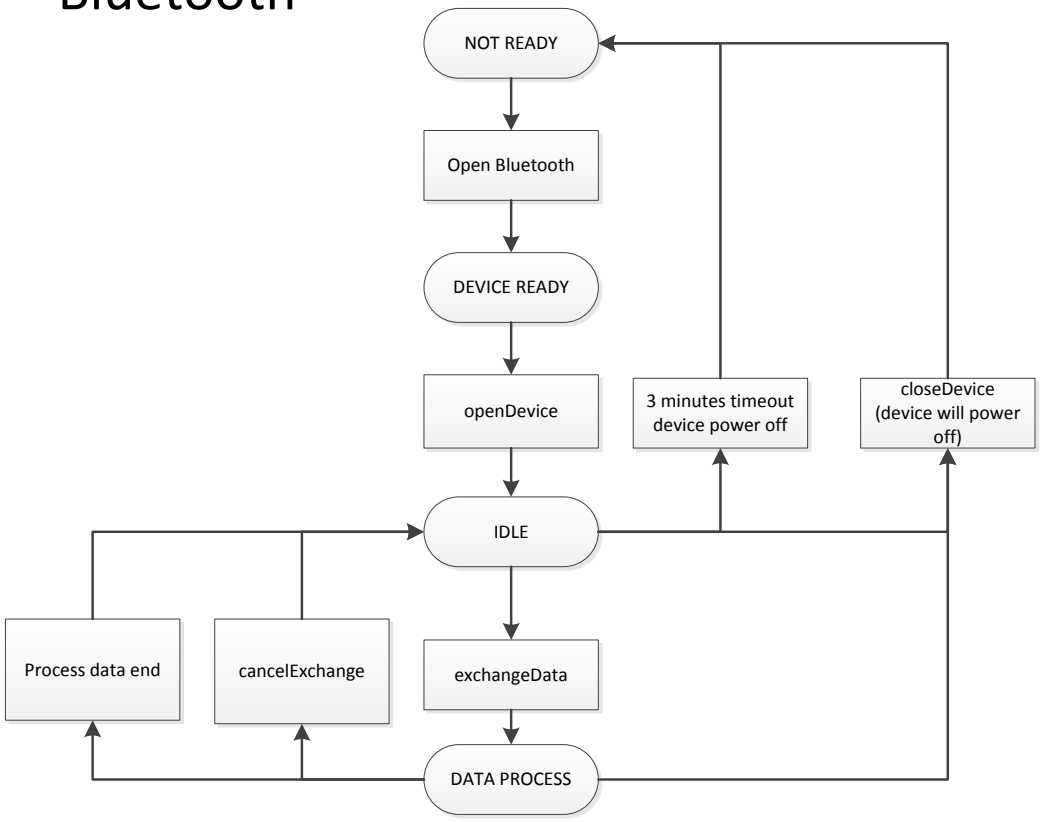
Device State for Audiojack

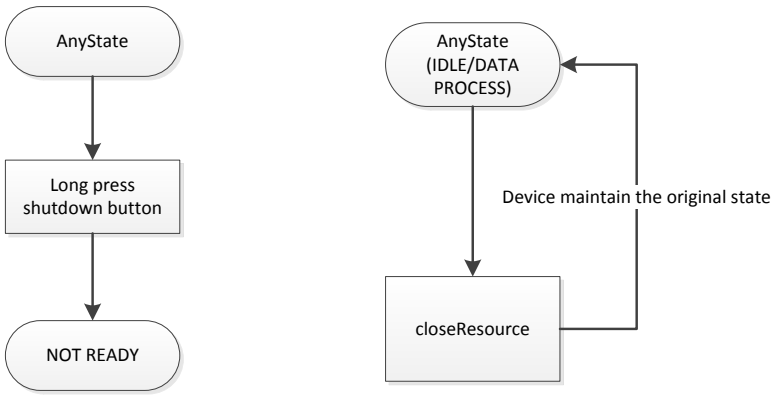




device state machine for audio jack

Device State for Bluetooth





device state machine for Bluetooth

API Specification

CommunicationManagerBase

CommunicationManagerBase Constructor
+(CommunicationManagerBase*) sharedInstance:**(DeviceCommunicationChannel)** channel Create a **CommunicationManagerBase** Singleton to communicate with device. Parameter: *channel*: BLUETOOTH or AUDIOJACK indicates which communication channel that device connected with the phone. Returns: *AudioJackManager* success to **create** AudioJackManager singleton or *BluetoothManager success to create* BluetoothManager singleton *nil* if failed

CommunicationManagerBase Methods	**Descriptions**
+ (int) searchDevices:(id<DeviceSearchListener>) dsl detectAudioDevice:(**BOOL)** detectAudioDevice detectBluetooth:(**BOOL)** detectBluetoothDevice timeout:(**long)** timeout	start discovering device around, the searching result will return by **DeviceSearchListener,** see detail in **DeviceSearchListener** interface Parameter: *dsl*: search result will return by it, *detectAudioDevice*: If application want to detect which a device connected phone by audio jack, set this value to true, else false. *detectBluetoothDevice*: If application want to detect which a device connected phone by bluetooth, set this value to true, else false. *timeout*: the max time to search, this parameter is not used now, just reserve for future. Returns: 0 phone start to discover device -1 phone don't support Bluetooth, -2 phone has not power on Bluetooth, -3 phone has error when startDiscovery Bluetooth Device.
+(NSString*) getLibVersion	Return library version.
-(int) openDevice:(**NSString*)** identifier	See detail in AudioJackManager or BluetoothManager
-(int)openDevice:(NSString*) identifier cb:**(id<CommunicationCallBack>)** cb	See detail in AudioJackManager or BluetoothManager

mode:(DeviceCommunicationMode)mode	
-(int) exchangeData:(**NSData*)** data timeout:**(long)** timeout cb:**(id<CommunicationCallBack>)** cb	See detail in AudioJackManager or BluetoothManager
-(int) exchangeData:(**NSData*)** data timeout:**(long)** timeout	See detail in AudioJackManager or BluetoothManager
-(int) cancelExchange	See detail in AudioJackManager or BluetoothManager
-(void) closeDevice	See detail in AudioJackManager or BluetoothManager
-(void) closeResource	See detail in AudioJackManager or BluetoothManager

AudioJackManager

AudioJackManager Constructor
+(AudioJackManager*)sharedInstance; Create a AudioJackManager Singleton to communicate with device. Returns *AudioJackManager* success to create AudioJackManager singleton

AudioJackManager Methods	**Descriptions**
-**(int)**openDevice:(NSString* identifier)	Initialize the AudioQueueNewOutput, AudioQueueNewInput, thread resource and send boot signal to open the device. Parameter identifier is reserve. Returns 0 represent opening success Returns-1 represent initialize AudioQueueNewOutput failed Returns-2 represent initialize AudioQueueNewInput failed Returns-3 represent shake failed Returns-4 represent no device detected.
-(int)openDevice:(NSString*) identifier cb:**(id<CommunicationCallBack>)** cb mode:(DeviceCommunicationMode)mode	Reserve. Do nothing.

-**(void)**closeDevice	Try send close signal 3 times to close the device and finally, release the AudioQueueNewOutput, AudioQueueNewInput, thread resource.
-**(void)**closeAudioResource	Release the AudioQueueNewOutput, AudioQueueNewInput, thread resource.
-**(int)**exchangeData:**(NSData*)** data timeout**:(long)**timeout cb:**(id< CommunicationCallBack >)**cb	Do the data exchanging between app and device in the specified timeout. . It will call CommunicationCallBack after data exchanging finished. See the protocol CommunicationCallBack. Returns 0 represent phone can send data Returns -1 represent the current data exchanging is not completed, it can't do exchange now. Returns -2 represent phone is not opened. Returns-4 represent no device detected.
-(int) exchangeData:(**NSData*)** data timeout:**(long)** timeout	Reserve. Do nothing.
-(int)cancelExchange	Try send cancel command 3 times to cancel the current data exchanging. Returns 0 represent can cancel the current exchange, it will call onError when cancel command finish. See the detail in Error Code. Returns -1 represent there is not data exchanging , no need to cancel. Returns -2 represent phone is not opened. -3 represent phone is canceling, can't cancel again. Returns-4 represent no device detected.
+(NSString*)getLibVersion	Return library version.

BluetoothManager

BluetoothManager Constructor
+(BluetoothManager) sharedInstance; Get a BluetoothManager Singleton after calling shareInstance. Returns: *BluetoothManager* which is created by shareInstance; *nil* if failed

BluetoothManager Methods	**Descriptions**

-(int) openDevice:**(NSString*)** identifier	Set up connection with the specified Bluetooth identifier Parameter: *Identifier*: for Bluetooth maybe it's an address or uuid. Returns: 0 represent setup connection success, -1 represent the address is Illegal, -2 represent setup connection failed. -4 phone don't support Bluetooth, -5 phone has not power on Bluetooth, -6 phone has unknown error.
-(int)openDevice:(NSString*) identifier cb:**(id<CommunicationCallBack>)** cb mode:(DeviceCommunicationMode)mode	Set up connection with the specified Bluetooth identifier Parameter: *Identifier*: for Bluetooth maybe it's an address or uuid. *cb:* exchange callback. mode:indicate the communication mode. Returns: 0 represent setup connection success, -1 represent the address is Illegal, -2 represent setup connection failed. -4 phone don't support Bluetooth, -5 phone has not power on Bluetooth, -6 phone has unknown error.
-(void) closeDevice	Disconnected Bluetooth and release resource.
-(void) closeResource()	Disconnected Bluetooth and release resource.
-(int) exchangeData:**(NSData*)** data, timeout:**(long)** timeout cb:**(id<CommunicationCallBack>)** cb	Do the data exchanging between app and device in the specified timeout. It will call **CommunicationCallBack** after data exchanging finished. See the **CommunicationCallBack** protocol. Parameter: *data:* exchange data, *timeout:* exchange timeout, *cb:* exchange callback. Returns : 0 represent phone can send data -1 represent the current data exchanging is not completed, it can't do exchange now. -2 represent phone is not opened.

	-4
int cancelExchange()	Try send cancel command to cancel the current data exchanging Returns: 0 represent can cancel the current exchange, it will call onError() when cancel command finish. See the detail in Error Code. -1 represent represent there is not data exchanging , no need to cancel. -2 represent phone is not opened.

DeviceInfo

DeviceInfo Methods	Descriptions
-(NSString*) getName	Get the device name, refers to the Bluetooth indicates Bluetooth' name and it is null for the audio jack.
-(NSString*) getIdentifier	Get the device Identifier, refers to the Bluetooth indicates device's address or uuid and it is null for the audio jack.
-(DeviceCommunicationChannel)getDevChannel	Get the device's communication channel, the DeviceCommunicationChannel.BLUETOOTH for Bluetooth and DeviceCommunicationChannel.AUDIOJACK for audio jack.

DevcieCommunicationMode

DeviceInfo Methods	Descriptions
MASTERSLAVE	Master-slave mode, exchangeData and timeout onstitute a session.
DUPLEX	Duplex mode,exchangeData just send data with out timeout.Device can take the initiative to send data.

DeviceCommunicationChannel

DeviceCommunicationChannel member	Descriptions
AUDIOJACK	AUDIOJACK indicates that device connected with phone by audio jack
BLUETOOTH	BLUETOOTH indicates that device connected with phone by Bluetooth

DeviceSearchListener

DeviceSearchListener ~~Interface~~ protocol Methods
-(void) discoverOneDevice**:(DeviceInfo*)** devInfo when discovering a new Bluetooth device ,this function will notify application the new Bluetooth device's name and address
-(void) discoverComplete when searching complete, this function will notify application search complete.

CommunicationCallBack

CommunicationCallBack protocol ~~Interface~~ Methods
-(void) onSendOK Callback this function when phone send the data ok.
-(void) onReceive:(**NSData*)** data Callback this function when exchange the data ok, and the parameter ***data*** stores the data from device send to phone.
-(void) onTimeout Callback this function when data exchanging timeout.
-(void) onError:**(NSInteger)** code message:**(NSString*)** msg Callback this function when exchange the data failed, see the Error Code.
void onProgress:(**NSData***) data Callback this function when receive MSG data.

Error Code

Error code description

Code:3 **MSG:** Failed to decode audio waveform, prompt send command again.
Code:4 **MSG:** Not enough memory, call closeAudioResource and open device again.
Code:5 **MSG:** Timeout, prompt check the device is power on.
Code:6 **MSG:** Byte format error, prompt send command again.
Code:7 **MSG:** Frame format error, prompt send command again.
Code:8 **MSG:** Unknown Error, call closeAudioResource and open device again.
Code:9 **MSG:** AudioTrack write data error, you must call openDevice again.
Code:10 **MSG:** AudioRecord read data error, you must call openDevice again.
Code:11 **MSG:** Exchange state error, prompt send command again.
Code:12 **MSG:** Cancel success.
Code:13 **MSG:** Cancel failed.
Code:20 **MSG:** Bluetooth channel disconnected

API usage

API Calling procedure

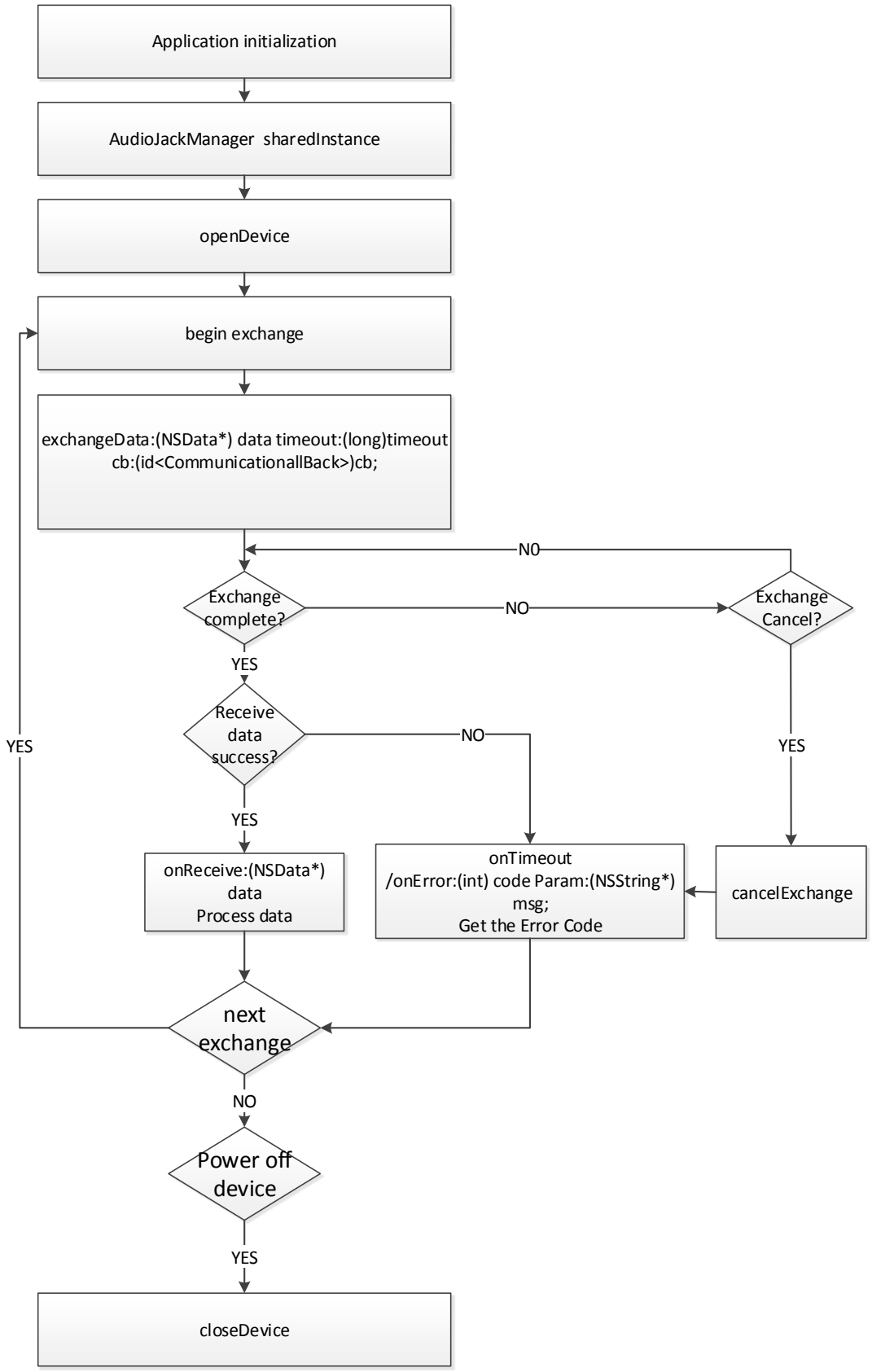
AudiojackManager

Developer starts by creating an instance of AudioJackManager by calling ***[AudioJackManager sharedInstance]***. When device pluged into the phone, developer can call *openDevice* to power on the device. Developer calls ***-(int)exchangeData:(NSData*) data timeout:(long)timeout cb:(id<AudioJackCallback>)cb*** to send the data, if succeeded, the phone will wait for the data from device in the parameter *timeout.* When exchanging completed, the ***AudioJackCallBack*** will be called according to the result. See the detail in Error Code.

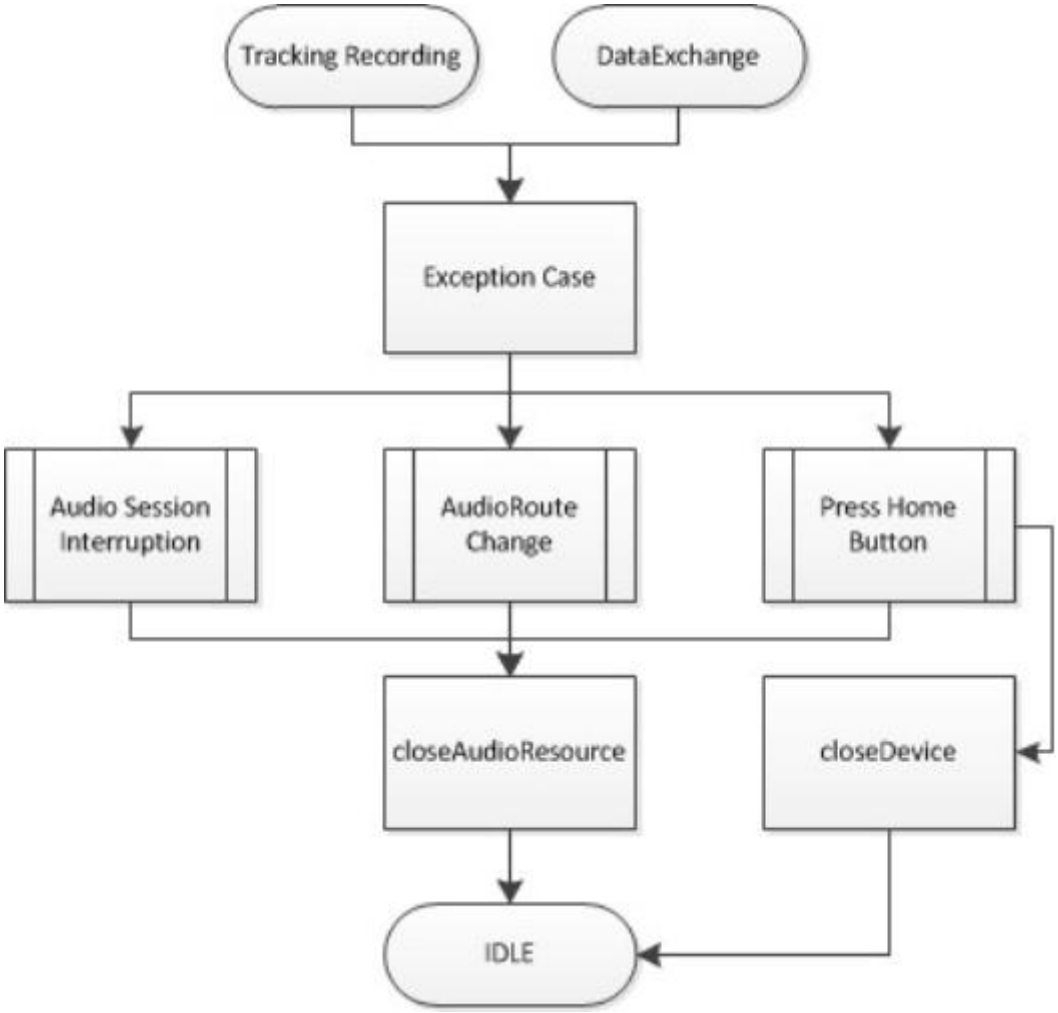
Developer can call ***closeDevice*** to power off the device after exchanging finished, but if you don't want to power off the device, just do noting.

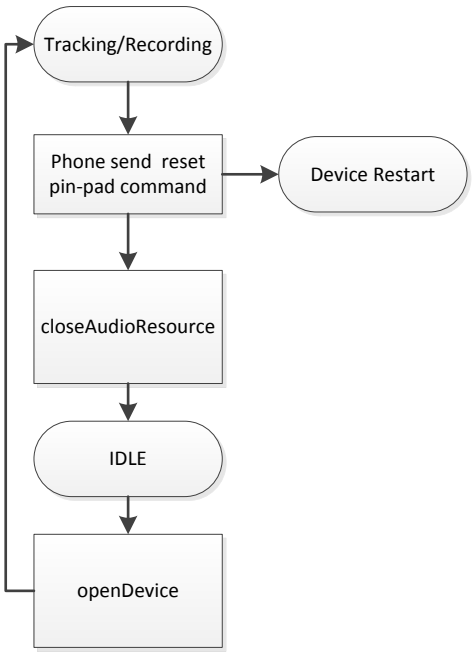
Notice that ***closeAudioResource*** must be called when the below exceptions occur.

Normal case:



Exception case:





when application send reset-pinpad command, application has to call closeAudioResource and

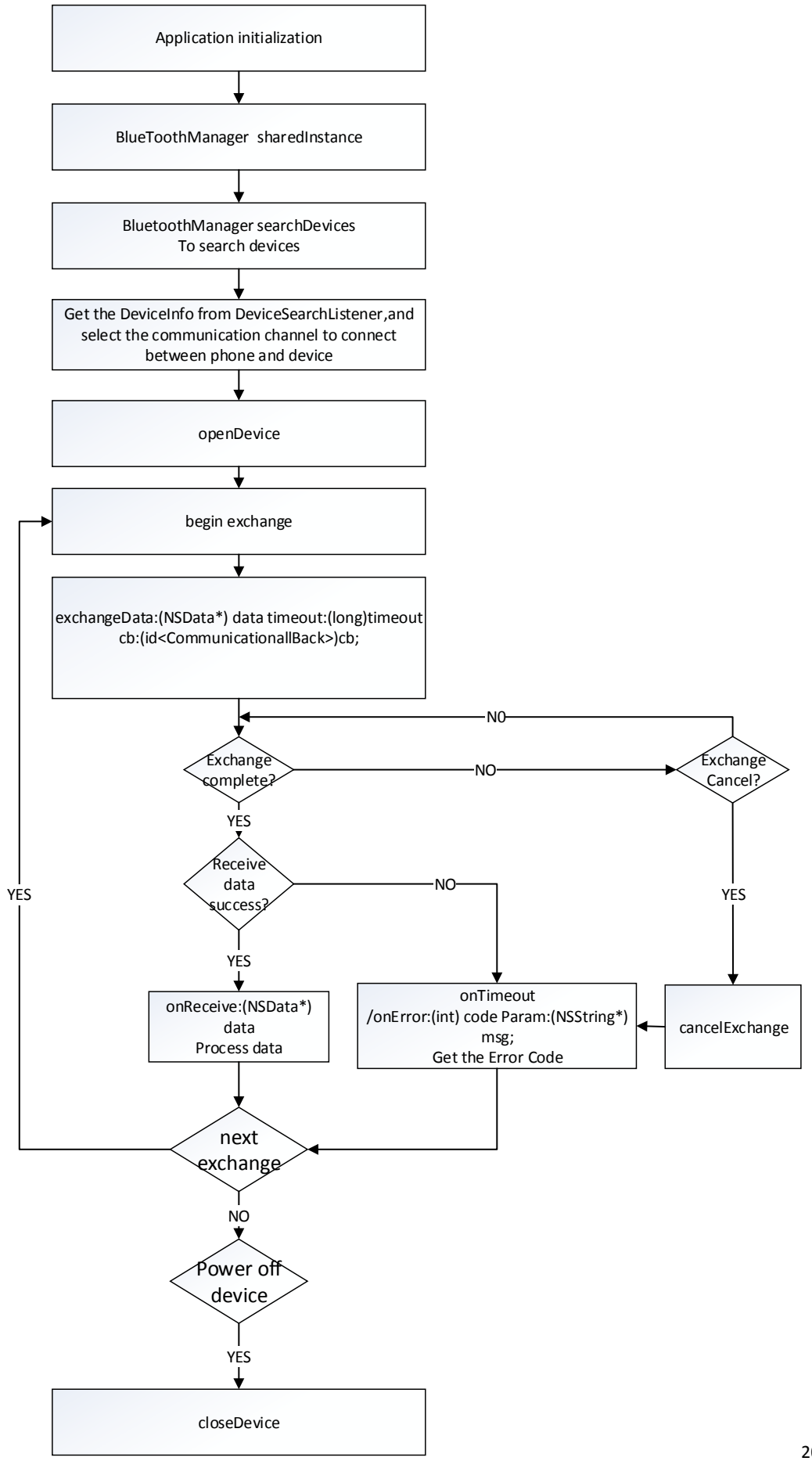
call openDevice again.

BluetoothManager

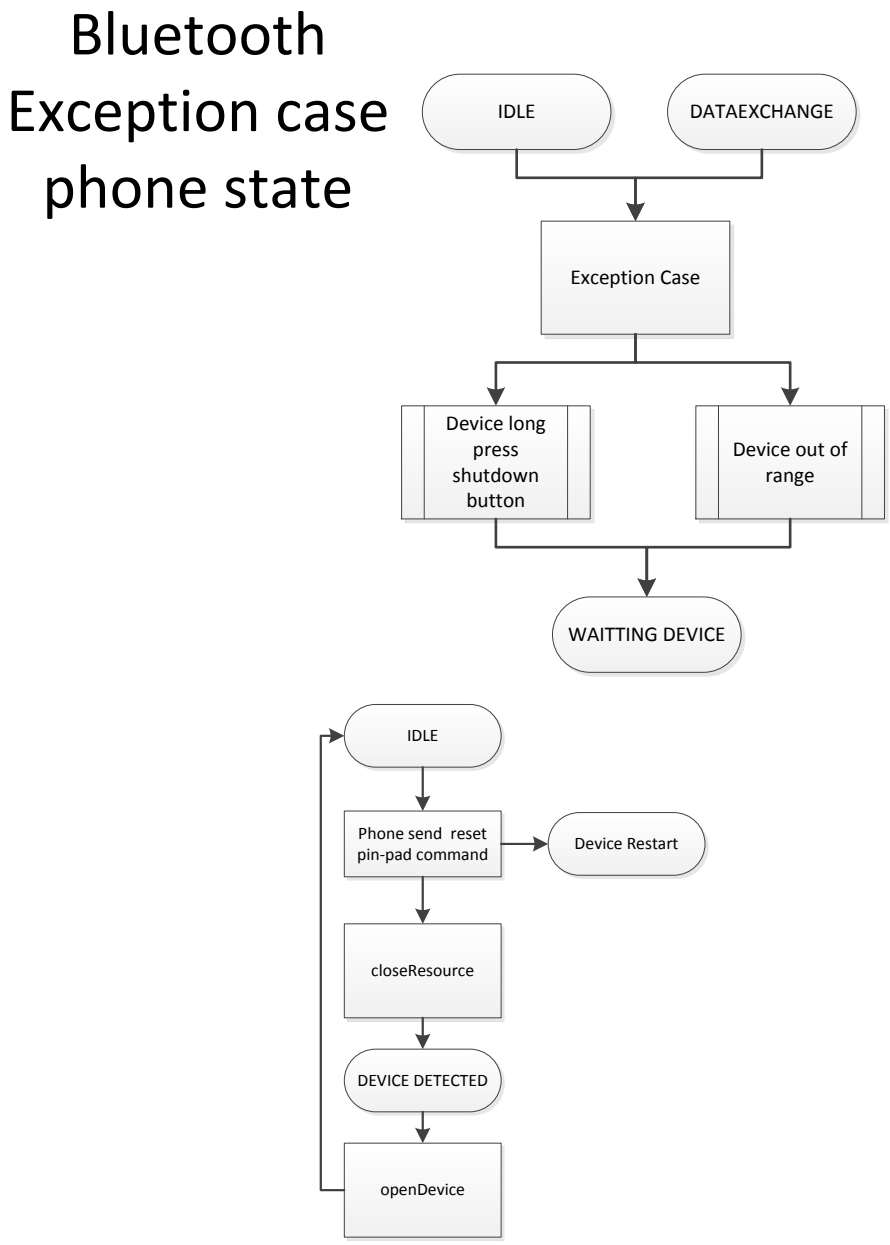
Developer starts by creating an instance of BluetoothManager by calling **[*BluetoothManager sharedInstance]*,** and passing in an Android Application Context. **[*BluetoothManager shareInstance]*** gets the instance of BluetoothManager. When device opening the Bluetooth, developer can call *searchDevices**(id<DeviceSearchListener>)** dsl* to discover device, Then you can get the device information from *DeviceSearchListener*, It return a *DeviceInfo* class that include communication channel and device identifier. And calling *openDevice:**(NSString*)** identifier* to setup connection with device by Bluetooth. Developer calls *exchangeData:**(NSData*)***data *timeout:**(long)** timeout cb:**(id<CommunicationCallBack>)** cb* to send the data, if succeeded, the phone will wait for the data from device in the parameter *timeout.* When exchanging completed, the ***CommunicationCallBack*** will be called according to the result. See the detail in Error Code.

Developer can call ***closeDevice*** to power off the device and disconnect Bluetooth connection, but if you don't want to power off the device, you can call ***closeResource*** just disconnect Bluetooth connection.

Normal case:


Application initialization
BlueToothManager sharedInstance
BluetoothManager searchDevices
To search devices
Get the DeviceInfo from DeviceSearchListener,and select the communication channel to connect between phone and device
openDevice
begin exchange
exchangeData:(NSData*) data timeout:(long)timeout cb:(id<CommunicationallBack>)cb;
Exchange complete?
NO
Exchange Cancel?
YES
Receive data success?
NO
YES
YES
onReceive:(NSData*) data
Process data
onTimeout
/onError:(int) code Param:(NSString*) msg;
Get the Error Code
cancelExchange
next exchange
YES
NO
Power off device
NO
YES
closeDevice

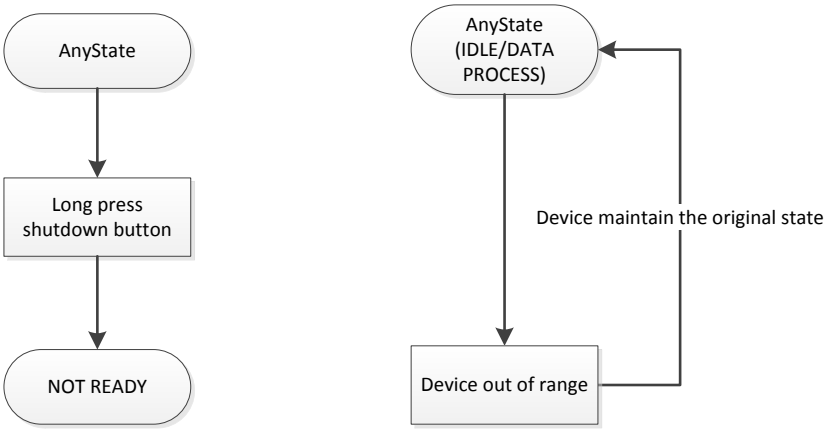
Exception case:



when application send reset-pinpad command, application has to call closeResource and call openDevice again.

Notice: We need to confirm whether phone can detect device's disconnecting quickly. if it work perfectly, closeResource can be skipped.

Bluetooth Exception case device state



CommunicationManagerBase

Develoer starts from **[*CommunicationManagerBase searchDevice:(id<DeviceSearchListener>) dsl detetAudioDevice:(BOOL) detectAudioDevice detectBluetoothDevice:(BOOL) detectBluetoothDevice timeout:(long) timeout]*** to search devices. Then you can get the device information from *DeviceSearchListener*, It return a *DeviceInfo* class that include communication channel and device identifier. You can use communication channel to creating an instance by calling **[*CommunicationManagerBase shareInstance:(DeviceCommunicationChannel) channel]***. Then call ***openDevice:(NSString*) identifier*** to connect device. Developer calls ***exchangeData:(NSData*)*** **data** ***timout:(long) timeout cb:(id< CommunicationCallBack>) cb*** to send the data, if succeeded, the phone will wait for the data from device in the parameter *timeout.* When exchanging completed, the ***CommunicationCallBack*** will be called according to the result. See the detail in Error Code.

Developer can call ***closeDevice*** to power off the device after exchanging finished, but if you don't want to power off the device, just do nothing.

Creating and Configuring a AudioJackManager

This section provides an example of major tasks a developer follows to integrate CommunicationManagerBase into an iOS application.

Add API files to Xcode project

Please add CommunicationCallBack.h, CommunicationManagerBase.h, DeviceInfo.h, DeviceSearchListener.h ,libMPOSCommunicationManager.a to XCode Project,and then you can use the static library For secondary development.

Add extra Frameworks to Xcode project

Complete the following steps:

1、 Right click Frameworks folder
2、 Choose Add
3、 Choose Existing Frameworks…
4、 Add MediaPlayer.framework,AudioToolbox.framework,CoreBluetooth.framework,UIKit.framework

Link Objective-C classes in the static library

Complete the following steps:

1、 Select Target Info
2、 Select Build
3、 Search for “Other Linker Flags”
4、 Modify value to –all_load

Change Filename Extension

Change the filename extension from .m to .mm and the filetype from Objective-C Source to Objective-C++ Source for project.

Important key

1. You have to create a AudioSession , and set the property kAudioSessionCategory_PlayAndRecord. So that, you application can play and record in the same time.
2. You have to set listeners to the AudioSession interruption and audio route changed, please refer to the following code.

```
OSStatus error = 0;
error = AudioSessionInitialize(NULL, NULL, interruptionListener,NULL);
if (error)
{
    NSLog(@"ERROR INITIALIZING AUDIO SESSION! %ld\n", error);
}
else
{
    UInt32 category = kAudioSessionCategory_PlayAndRecord;
    error = AudioSessionSetProperty(kAudioSessionProperty_AudioCategory, sizeof(category), &category);
    if (error)
        NSLog(@"couldn't set audio category!");

    error = AudioSessionAddPropertyListener(kAudioSessionProperty_AudioRouteChange, propListener, NULL);
    if (error)
        NSLog(@"ERROR ADDING AUDIO SESSION PROP LISTENER! %ld\n", error);
    UInt32 inputAvailable = 0;
    UInt32 size = sizeof(inputAvailable);

    // we do not want to allow recording if input is not available
    error = AudioSessionGetProperty(kAudioSessionProperty_AudioInputAvailable, &size, &inputAvailable);
    if (error)
        NSLog(@"ERROR GETTING INPUT AVAILABILITY! %ld\n", error);

    // we also need to listen to see if input availability changes
    error = AudioSessionAddPropertyListener(kAudioSessionProperty_AudioInputAvailable, propListener, NULL);
    if (error)
        NSLog(@"ERROR ADDING AUDIO SESSION PROP LISTENER! %ld\n", error);

    error = AudioSessionSetActive(true);
    if (error)
        NSLog(@"AudioSessionSetActive (true) failed");
}
```

3. Because AudioJackManager set the system volume maximum when sending command , the system volume menu will popup, to avoid it , you can refer the following code, if you have any better way to set the volume maximum, please let me know.

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];
    // Override point for customization after application launch.
    if ([[UIDevice currentDevice] userInterfaceIdiom] == UIUserInterfaceIdiomPhone) {
        self.viewController = [[[ViewController alloc] initWithNibName:@"ViewController_iPhone" bundle:nil] autorelease];
    } else {
        self.viewController = [[[ViewController alloc] initWithNibName:@"ViewController_iPad" bundle:nil] autorelease];
    }
    self.window.rootViewController = self.viewController;
    [self.window makeKeyAndVisible];

    MPVolumeView *myVolumeView = [[MPVolumeView alloc] initWithFrame:CGRectZero];
    [myVolumeView sizeToFit];
    [self.window addSubview:myVolumeView];
    [myVolumeView release];

    return YES;
}
```