

Stream SDK for iOS

July 23, 2015



The NeuroSky® product families consist of hardware and software components for simple integration of this biosensor technology into consumer and industrial end-applications. All products are designed and manufactured to meet consumer thresholds for quality, pricing, and feature sets. NeuroSky sets itself apart by providing building block component solutions that offer friendly synergies with related and complementary technological solutions.

Reproduction in any manner whatsoever without the written permission of NeuroSky Inc. is strictly forbidden. Trademarks used in this text: eSense™, CogniScore™, ThinkGear™, MindSet™, MindWave™, NeuroBoy™, NeuroSky®

NO WARRANTIES: THE NEUROSKY PRODUCT FAMILIES AND RELATED DOCUMENTATION IS PROVIDED "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, INCLUDING PATENTS, COPYRIGHTS OR OTHERWISE, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT SHALL NEUROSKY OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, COST OF REPLACEMENT GOODS OR LOSS OF OR DAMAGE TO INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE NEUROSKY PRODUCTS OR DOCUMENTATION PROVIDED, EVEN IF NEUROSKY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. , SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES.

USAGE OF THE NEUROSKY PRODUCTS IS SUBJECT OF AN END-USER LICENSE AGREEMENT.

Contents

| | |
|---|-----------|
| Stream Development Guide | 4 |
| Introduction | 4 |
| Stream SDK for iOS Contents | 4 |
| Supported Neurosky Hardware | 4 |
| Supported iOS Version | 5 |
| SDK Features | 5 |
| Your First Project: StreamSDKDemo | 5 |
| Develop Your Own Neurosky Hardware Enabled Apps for iOS | 5 |
| Configure Your Environment | 6 |
| Set Up the Stream | 8 |
| Handle Data Received | 8 |
| Handle Accessory Connection and Disconnection | 9 |
| Start and Stop the Data Stream | 9 |
| Log Messages | 10 |
| Further Considerations | 10 |
| References | 10 |
| Stream API Reference | 11 |
| Overview | 11 |
| Configuration | 11 |
| Get the Shared Stream Manager | 11 |
| Init Accessory Session | 11 |
| Tear Down Accessory Session | 11 |
| Set Parser | 11 |
| Set path for recording the Data Stream | 11 |
| Start and Stop recording the Data Stream | 11 |
| Get SDK Version | 11 |
| Enable SDK Log | 12 |
| Property | 12 |
| Class Methods | 12 |
| sharedInstance | 12 |
| Instance Methods | 12 |
| initWithConnectWithFile | 12 |
| initWithConnectWithAccessorySession | 13 |
| tearDownAccessorySession | 13 |
| setParser | 13 |
| setRecordStreamFilePath | 13 |
| setRecordStreamFilePath: | 14 |
| startRecordRawData | 14 |
| stopRecordRawData | 14 |
| getVersion | 14 |
| enableLog | 14 |
| Enum | 14 |
| DeviceType | 14 |
| MindDataType | 15 |
| BodyDataType | 15 |
| ParserType | 15 |

| | |
|--|----|
| Connection Status | 15 |
| Raw Data Record Error | 16 |
| Stream Delegate Protocol Reference | 16 |
| Overview | 16 |
| Protocol Definition | 16 |
| Instance Methods | 17 |

Stream Development Guide

Introduction

This guide will teach you how to use **Stream SDK for iOS** to write iOS applications that can acquire bio-signal data from NeuroSky's Hardware. This will enable your iOS apps to receive and use bio-signal data such as EEG and ECG acquired via Classic Bluetooth, UART-30PIN and File source formatted as a Stream.

This guide (and the entire **Stream SDK for iOS** for that matter) is intended for programmers who are already familiar with standard iOS development using Xcode and Apple's iOS SDK. If you are not already familiar with developing for iOS, please first visit Apple's web site for instruction and tools to develop iOS apps.

If you are already familiar with creating typical iOS apps, then the next step is to make sure you have downloaded NeuroSky's **Stream SDK for iOS**. Chances are, if you're reading this document, then you already have it.

Stream SDK for iOS Contents

- Development Guide (this document)
- API Reference (this document)
- SDK static library and headers
 - StreamSDK.a
 - TGSEEGPower.h
 - TGStream.h
 - TGStreamDelegate.h
 - TGStreamEnum.h
- StreamSDKDemo example project for iOS

You'll find the "StreamSDK.a" in the `lib/` folder, and the "StreamSDKDemo example project" in the `SampleProject/` folder.

Supported Neurosky Hardware

The following Neurosky hardware are currently supported:

- MindWave Mobile - TGAM for Mind (EEG)
- CardioChip Starter Kit - BMD101 for Body (ECG)

Important: Before using any iOS application that uses the Stream SDK for iOS, make sure you have paired the Neurosky Hardware to your iOS device by carefully following the instructions in the User Manual that came with each Neurosky Hardware!

Supported iOS Version

- Support iOS Version 6.0 later

SDK Features

- Support Automatic Reference Counting in this release.

Your First Project: StreamSDKDemo

Important: Apple has announced that simulator not support External Accessory frameworks. Testing External Accessory applications will require access to a real iOS device going forward. For how to set up a test environment using real iOS device, please visit iOS Developer Library: [App Distribution Guide](#).

StreamSDKDemo is a sample project we've included in the **Stream SDK for iOS** that demonstrates how to setup, connect, and handle data to a Neurosky Hardware. Add the project to your Xcode environment by following these steps:

1. In Xcode, select **File** —> **Open** —>
2. Browse in the Stream SDK to select the StreamSDKDemo directory
3. Click the Open button
4. Update the code signing options in the project target settings
5. Select **Product** —> **Run** to compile, link and start StreamSDKDemo in the Xcode emulator.

Note: This is an example application. It may not be completely compliant with Apple's guidelines for building deploy-able applications.

At this point, you should be able to browse the code, make modifications, compile, and deploy the app to your device or emulator just like any typical iOS application.

Develop Your Own Neurosky Hardware Enabled Apps for iOS

For most applications, using the Stream iOS API is recommended. It reduces the complexity of managing Neurosky Hardware's accessory connections and handles parsing of the data stream from Neurosky Hardware's accessory. To make a brainwave-sensing application, all you need to do is to import a library, add the requisite setup and tear-down functions, and assign a delegate object to which accessory event notifications will be dispatched.

Some limitations of the Stream SDK for iOS API include:

- Can only communicate with one attached Neurosky Hardware Enabled accessory

The "Stream API Reference" contains descriptions of the classes and protocols available in the Stream iOS API.

The Stream SDK for iOS also includes the `StreamSDKDemo` sample project, which is a simple demo iOS application that displays the connection with Neurosky Hardware.

Configure Your Environment

In order for you app to communicate with any Neurosky hardware module, you must include the `Supported external accessory protocols` key in your app's `Info.plist` file.

This key contains an array of strings that identify the communications protocols that your app supports.

Add `com.neurosky.thinkgear` to the list of supported external accessory protocols.

Your project window should now look similar to this:

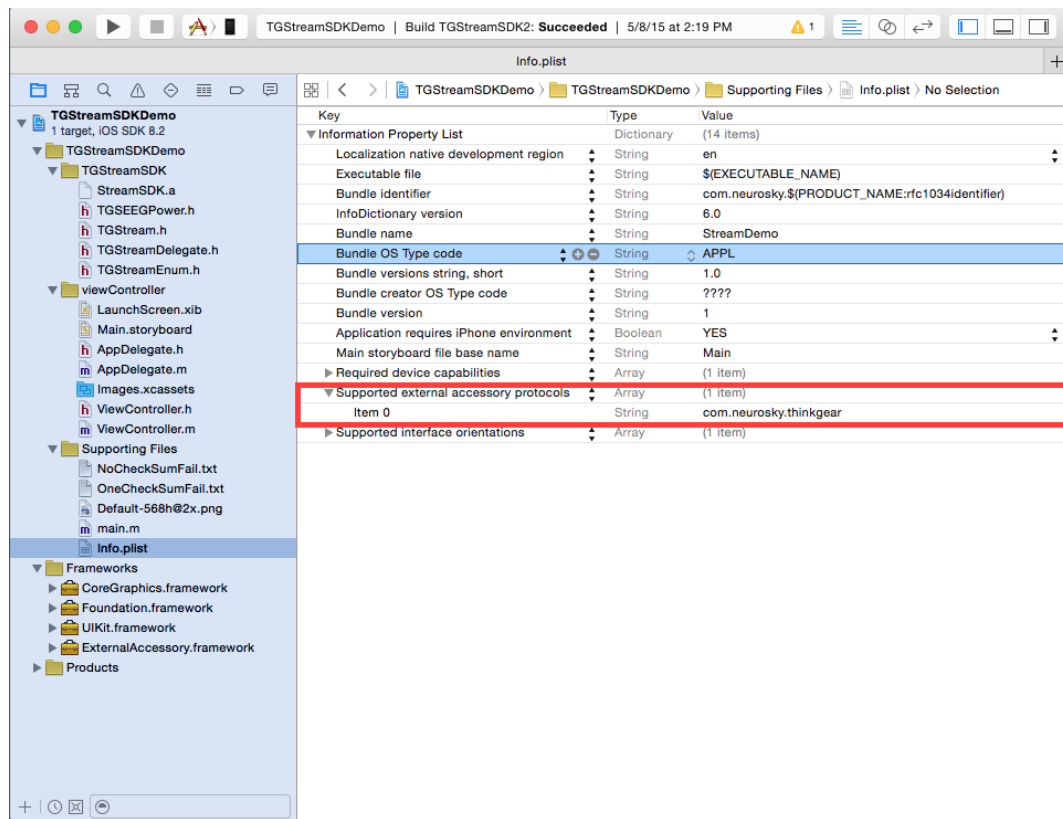


Figure 1.1: Add Supported External Accessory

Copy the following directories from the `lib` directory in the SDK for iOS into the `StreamSDK` group in your project:

- `StreamSDK.a`
- `TGSEEGPower.h`

- `TGStream.h`
- `TGStreamDelegate.h`
- `TGStreamEnum.h`

Your project window should now look similar to this:

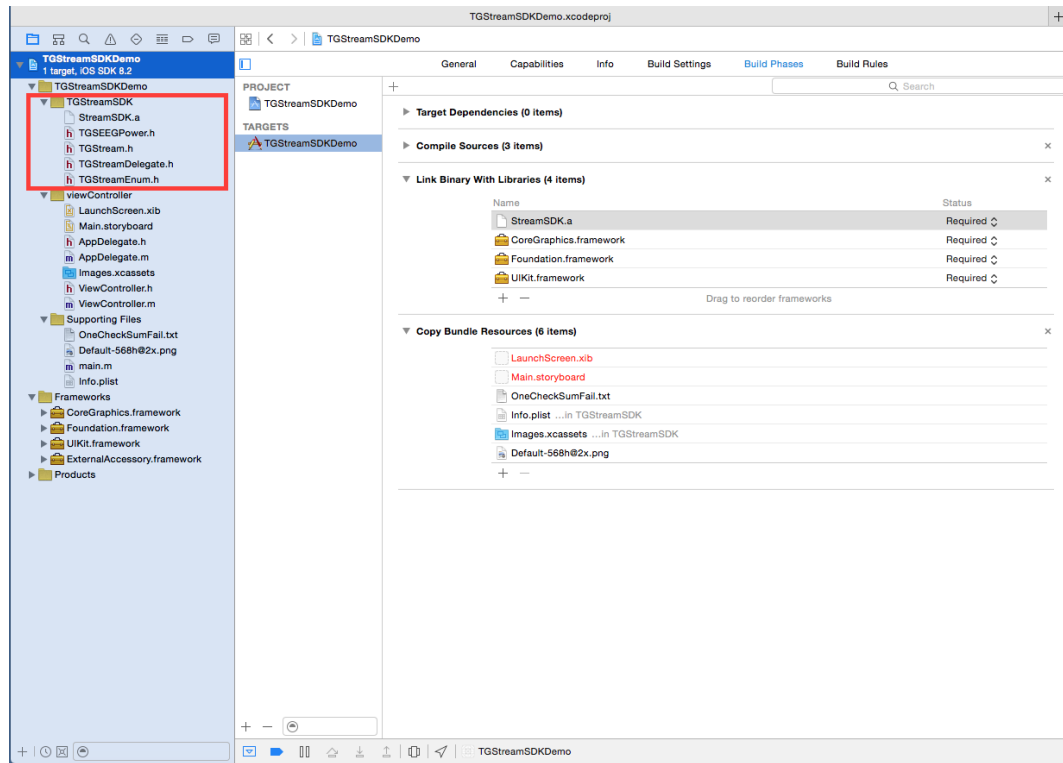


Figure 1.2: Copy Stream SDK iOS library to project

Next, add the ExternalAccessory frameworks to the project.

1. Navigate to your project settings
2. Select your target
3. Select Build Phases
4. Expand **Link Binary With Libraries**
5. Click on + and select `StreamSDK.a` and click **Add**

Your project window should now look similar to this:

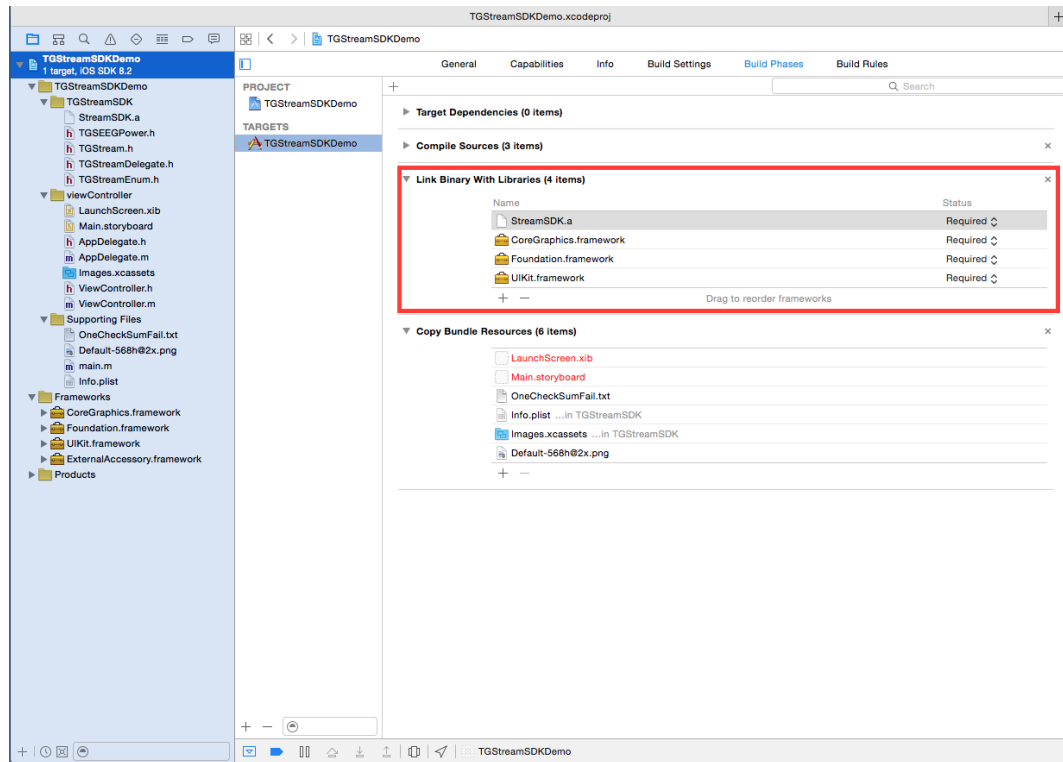


Figure 1.3: Add frameworks to project

Then import the appropriate header files into the requisite classes.

Set Up the Stream

Set up the `Stream` should be performed as early as necessary. Typically, this would be in the `viewDidLoad` method in the `ViewController` class. Simply add the following two lines to that method:

```
TGStreamInstance=[ TGStream sharedInstance];
TGStreamInstance.delegate=self;
```

This sets up the shared `TGStream` instance. The delegate can be set to any class that implements the `TGStreamDelegate` protocol — in this case, it's an instance of `ViewController`.

Handle Data Received

Since the delegate object was set to be a `ViewController` instance, we have to edit its class definition to indicate support of the `TGStreamDelegate` protocol. In the sample project file, the class definition in `ViewController.h` looks similar to the following:

```
@interface ViewController : UIViewController
```

Simply modify the definition in the following way:

```
@interface ViewController : UIViewController<TGStreamDelegate>
```

And in the implementation file, implement the method. A few `NSLog` calls are provided as a trivial example of accessing the `datatype data obj` parameter. Check the "Stream API Reference" for a full list of the supported data.

```
-(void) onDataReceived: (NSInteger)datatype data: (int)data obj: (NSObject *)obj
deviceType: (DEVICE_TYPE) deviceType{

    NSLog(@"dataType: %d data: %d deviceType: %d", (int)datatype, data, (int) deviceType);

    if (obj) {

        TGSEEGPower *EEGPowerInstance=(TGSEEGPower *)obj;
        NSString *TGSEEGPowerSrting=[ NSString stringWithFormat:@"%@\n TGSEEGPower delta-- %d\n
TGSEEGPower theta-- %d\n TGSEEGPower lowAlpha-- %d\n TGSEEGPower highAlpha-- %d\n TGSEEGPower
lowBeta-- %d\n TGSEEGPower highAlpha-- %d\n TGSEEGPower lowGamma-- %d\n TGSEEGPower lowGamma--
%d\n ",

                                [self NowString],
                                TGSEEGPowerInstance.delta,
                                TGSEEGPowerInstance.theta,
                                TGSEEGPowerInstance.lowAlpha,
                                TGSEEGPowerInstance.highAlpha,
                                TGSEEGPowerInstance.lowBeta,
                                TGSEEGPowerInstance.highAlpha,
                                TGSEEGPowerInstance.lowGamma,
                                TGSEEGPowerInstance.lowGamma];

        NSLog(@"%@", TGSEEGPowerSrting);

    }
}
```

Handle Accessory Connection and Disconnection

The `TGStreamDelegate` protocol also specifies `onStatesChanged:` for the delegate object to handle accessory connection and disconnection . Add the following method definitions to the header file:

```
-(void) onStatesChanged: (ConnectionStates) connectionState;
```

In the implementation file, implement these methods:

```
-(void) onStatesChanged: (ConnectionStates) connectionState{

    NSLog(@"Connection States: %lu", (unsigned long) connectionState);

}
```

Start and Stop the Data Stream

When your application is ready to receive the EEG/ECG data, call the `initConnectWithFile` or `initConnectWithAccessorySession` method in `TGStream`. In the sample project, this is done by click button `InitFile` or `InitBT`.

```
-(IBAction) InitFile: (id) sender {

    NSBundle *mainBundle = [ NSBundle mainBundle];
    NSString *filePath = [ mainBundle pathForResource:@"sample_data" ofType:@"txt"];
```

```
[TGStreamInstance initWithFile:filePath];  
  
}  
  
- (IBAction)InitBT:(id)sender {  
  
    [TGStreamInstance initWithAccessorySession];  
  
}
```

You will also need a matching call to `tearDownAccessorySession` click button `TearDown`.

```
- (IBAction)TearDown:(id)sender {  
  
    [TGStreamInstance tearDownAccessorySession];  
  
}
```

Log Messages

The TGStream library will emit some debug messages through `NSLog()` to help you develop and debug your application. These messages will be prefixed with "Stream SDK —".

```
[TGStreamInstance enableLog:true];
```

Further Considerations

- Provide a consistent user experience by adhering to the guidelines set by the [NeuroSky Developer Application Standards](#) document.

References

- [Communicating with External Accessories](#) (Apple documentation)
- [EAAccessoryManager Class Reference](#)
- [EAAccessory Class Reference](#)
- [EASession Class Reference](#)

Stream API Reference

Overview

The `TGStream` class handles connections between a Neurosky Hardware accessory and an iOS device.

Configuration

Get the Shared Stream Manager

- `+(TGStream *) sharedInstance`

Init Accessory Session

- `-(void) initConnectWithFile: (NSString *) path`
- `-(void) initConnectWithAccessorySession`

Tear Down Accessory Session

- `-(void) tearDownAccessorySession`

Set Parser

- `-(void) setParserWith: (ParserType) parserType SampleRate: (int) sampleRate`

Set path for recording the Data Stream

- `-(void) setRecordStreamFilePath`
- `-(void) setRecordStreamFilePath: (NSString *) filePath`

Start and Stop recording the Data Stream

- `-(void) startRecordRawData`
- `-(void) stopRecordRawData`

Get SDK Version

- `-(NSString *) getVersion`

Enable SDK Log

- `-(void) enableLog: (BOOL) enabled`

Property

delegate

The object that acts as the delegate of the `TGStream`.

```
@property (nonatomic, weak) id<TGStreamDelegate> delegate;
```

Note

The delegate receives notifications about changes to the status of the Neurosky Hardware Enabled accessory, as well as data received notifications. The delegate must adopt the `TGStreamDelegate` protocol.

Class Methods

sharedInstance

Return the shared `TGStream` object for the iOS-based device.

```
+ (TGStream *) sharedInstance
```

Return Value

The shared `TGStream` object.

Note

You should always use this method to obtain the `TGStream` object, rather than creating an instance directly.

Instance Methods

initWithConnectWithFile

Open up a data stream from an input file. After calling this method, the delegate method `onDataReceived` and `onStatesChanged` will be called back.

```
-(void) initWithConnectWithFile: (NSString *) path;
```

Note

The purpose of `initWithConnectWithFile` is to help the developers to read `rawData` from files. Sometimes the developer wants to develop a demo but don't have the hardware. So, he can read data from `rawData` files. The `rawData` files is a binary file which stored the stream reading from Neurosky Hardware.

When you no longer want to receive `onDataReceived`, you should call the matching `tearDownAccessorySession` method.

`initConnectWithAccessorySession`

Open up a data stream from a ThinkGear accessory that connects via Bluetooth or 30-pin connector. After calling this method, the delegate method `onDataReceived` and `onStatesChanged` will be called back.

```
-(void) initConnectWithAccessorySession;
```

Note

First of All, you should make sure Neurosky Hardware was paired to iPhone/iPad. When you no longer want to receive `onDataReceived`, you should call the matching `tearDownAccessorySession` method.

`tearDownAccessorySession`

Close the data stream that is opening.

```
-(void) tearDownAccessorySession;
```

Note

Calls to this method must be balanced with a preceding call to the `initConnectWithAccessorySession` or `initConnectWithFile`

`setParser`

Parser setting for different Neurosky Hardware

```
-(void) setParserWith: (ParserType) parserType SampleRate: (int) sampleRate;
```

Parameters

- `parserType` — Default Parser
- `sampleRate` — 1024 for default Parser

Note

We can set `parserType` with `PARSER_TYPE_DEFAULT`. When the `parserType` is `PARSER_TYPE_DEFAULT`, the `sampleRate` will be ignored.

`setRecordStreamFilePath`

set the record stream with default path. The default path is `Documents/TGS_log/yyyy-MM-dd-HH-ii-ss-RawPackageRecording.txt`.

```
-(void) setRecordStreamFilePath;
```

setRecordStreamFilePath:

set the record stream with customized path.

```
-(void)setRecordStreamFilePath: (NSString *)filePath;
```

Parameters

- `filePath` — customized path

startRecordRawData

Start to record raw data

```
-(void)startRecordRawData;
```

stopRecordRawData

Stop recording raw data

```
-(void)stopRecordRawData;
```

Note

Calls to this method must be balanced with a preceding call `startRecordRawData`

getVersion

Return the version string of TGStream.

```
-(NSString *) getVersion;
```

enableLog

Enable log of TGStream, SDK log has prefix `Stream SDK`.

```
-(void)enableLog: (BOOL)enabled;
```

Enum

Declared in `TGStreamEnum.h`

DeviceType

```
typedef NS_ENUM(NSUInteger, DEVICE_TYPE){  
  
    DEVICE_TYPE_UNKNOWN = 0,  
    DEVICE_TYPE_MindWaveMobile = 1, //EEG  
    DEVICE_TYPE_CardioChipStarterKit = 2, //ECG  
  
};
```

Enum

July 23, 2015 | © 2009-2015 [NeuroSky, Inc.](#) All Rights Reserved.

MindDataType

Data types correspond to EEG

```
typedef NS_ENUM(NSUInteger, MindDataType)
{
    MindDataType_CODE_POOR_SIGNAL = 2,

    MindDataType_CODE_RAW = 128,

    MindDataType_CODE_ATTENTION = 4,

    MindDataType_CODE_MEDITATION = 5,

    MindDataType_CODE_EEGPOWER = 131,

};
```

BodyDataType

Data type correspond to ECG

```
typedef NS_ENUM(NSUInteger, BodyDataType)
{
    BodyDataType_CODE_POOR_SIGNAL = 2,

    BodyDataType_CODE_RAW = 128,

    BodyDataType_CODE_HEARTRATE = 3,

};
```

ParserType

```
typedef NS_ENUM(NSUInteger, ParserType){
    PARSER_TYPE_DEFAULT = 0,

};
```

Connection Status

```
typedef NS_ENUM(NSUInteger, ConnectionStates){

    STATE_INIT = 0,                // Neurosky Hardware Stream Init

    STATE_CONNECTING = 1,          // Neurosky Hardware Stream Connecting

    STATE_CONNECTED = 2,           // Neurosky Hardware Connected

    STATE_WORKING = 3,             // Neurosky Hardware Stream Or File Stream is Processing

    STATE_STOPPED = 4,            // when you tear down Neurosky Hardware Stream Or File Stream

};
```

```
STATE_DISCONNECTED = 5,    // Neurosky Hardware DisConnected

STATE_COMPLETE = 6,        // Neurosky Hardware Stream Or File Stream Processed Complete

STATE_RECORDING_START = 7, // start record raw data

STATE_RECORDING_END = 8,   // stop record raw data

STATE_FAILED = 100,        // File Stream Init Fail

STATE_ERROR = 101          // Neurosky Hardware Stream Init Fail
};
```

Raw Data Record Error

```
typedef NS_ENUM(NSUInteger, RecrodError){

    RECORD_ERROR_FILE_PATH_NOT_READY =1,        // set record stream file path fail

    RECORD_ERROR_RECORD_IS_ALREADY_WORKING =2, // Execute startRecordRawData twice continous
    without stopRecordRawData

    RECORD_ERROR_RECORD_OPEN_FILE_FAILED =3,    // Open record stream file fail

    RECORD_ERROR_RECORD_WRITE_FILE_FAILED =4    // Write record stream file fail

};
```

Stream Delegate Protocol Reference

Overview

The `TGStreamDelegate` protocol defines methods for handling accessory event notifications dispatched from a `TGStream` object.

Protocol Definition

Connection Status call back

- - `onStatesChanged:`

Check Sum Fail call back

- - `onChecksumFail:`

Record Fail call back

- - `onRecordFail:`

Data Received Events call back

- - `onDataReceived:` *required method*

Instance Methods

onStatesChanged:

Tells the delegate that accessory Connect Status with iOS-based device.

```
-(void) onStatesChanged: (ConnectionStates) connectionState;
```

Parameters

- `connectionState` — Connect Status declared in `TGStreamEnum.h`

onChecksumFail:

Tells the delegate that check sum failed

```
-(void) onChecksumFail: (Byte *)payload length: (NSUInteger)length checksum: (NSInteger)checksum;
```

Parameters

- `payload` — payload byte data
- `length` — length of payload
- `checksum` — the checksum value

onRecordFail:

Tells the delegate that some error happens when record raw data

```
-(void) onRecordFail: (RecrodError) flag;
```

Parameters

- `flag` — `RecrodError` declared in `TGStreamEnum.h`

onDataReceived:

Tells the delegate that data was received from the accessory

```
-(void) onDataReceived: (NSInteger)datatype data: (int)data obj: (NSObject *)obj  
deviceType: (DEVICE_TYPE) deviceType;
```

Parameters

- `datatype` — type of data
- `data` — value of data
- `obj` — `TGSEEGPower` Object
- `deviceType` — device Type declared in `TGStreamEnum.h`

if the `obj` is not `nil`, it will return an `TGSEEGPower` Object. The `TGSEEGPower` object has below property

- `delta` — the delta EEG power band
- `theta` — the theta EEG power band
- `lowAlpha` — the lowAlpha EEG power band
- `highAlpha` — the highAlpha EEG power band
- `lowBeta` — the lowBeta EEG power band
- `highBeta` — the highBeta EEG power band
- `lowGamma` — the lowGamma EEG power band
- `middleGamma` — the middleGamma EEG power band