Stream SDK for Mac - Development Guide and API Reference

November 10, 2016

NeuroSky®
Brain-Computer Interface Technologies

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.

Contents

Stream SDK for Mac Development Guide and API Reference

Introduction

This guide will teach you how to use NeuroSky's **Stream SDK for Mac** to write OSX applications that can read and use streams of data that conform to NeuroSky's ThinkGear Communication Protocol (binary streams of data that are output from NeuroSky's various hardware sensors). This SDK allows your app to read and use the data either realtime from a Bluetooth (wireless) stream, or from a stream of ThinkGear data that was previously recorded to a file.

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

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

Stream SDK for Mac Contents

- **Development Guide and API Reference**
  - StreamSDKForMac-DevGuide.pdf (this document)
- **API Library files** (in `lib/`)
  - TGStreamMac.framework
    - *TGSEEGPower.h*
    - *TGStreamForMac.h*
    - *TGStreamDelegate.h*
    - *TGStreamEnum.h*
- **Sample Projects** (in `SampleProject/`)
  - TGStreamMacDemo

Supported Neurosky Hardware

Supports data streams output from the following Neurosky hardware:

- MindWave Mobile

May also be able to work with data streams output from the following NeuroSky hardware:

- TGAT
- TGAM
- MindWave
- BMD10x
- Cardiochip Starter Kit Unit

Supported OSX Version

Supports development on:

- Xcode 7 and later.

Supports creation of apps for:

- OSX Version 10.8 "Mountain Lion" and later

SDK Features

- Support Automatic Reference Counting in this release.

Your First Project: TGStreamMacDemo

Important: For how to set up your Xcode development environment, please visit: Mac Developer Library: Start Developing Mac Apps Today.

TGStreamMacDemo is a sample project we've included in the **Stream SDK for Mac** that demonstrates how to setup an app, connect to a NeuroSky data stream, and then handle the data. Add the project to your Xcode environment by following these steps:

1. In Xcode, select **File** —> **Open** —>
2. Browse to the `SampleProject` folder in the **Stream SDK for Mac**, and select `TGStream-MacDemo`
3. Click the **Open** button
4. Update the code signing options in the project target settings
5. Select **Product** —> **Run** to compile, link and run TGStreamMacDemo in Xcode

Important: If you are connecting the app to a Bluetooth wireless device, such as the MindWave Mobile, make sure you have first paired it to your Mac carefully according to the User Manual or Quick Start Guide that came with that device, and that the device is turned on, and that Bluetooth is turned on on your Mac. Otherwise, your app will not be able to find and connect to the device. Refer to the documentation that came with the device (e.g. MindWave Mobile User Guide) for important device setup and usage information.

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 OSX application.

Note: This project is only intended to be an example to illustrate API usage. It may not be completely compliant with Apple's guidelines for building deployable applications.

Develop Your Own NeuroSky-Enabled Apps for OSX

If you desired, it would be possible for you to write a NeuroSky-enabled application entirely without using this SDK, simply by implementing your app according to the `ThinkGear Communication Protocol` document. However, this would take a lot of time, and be quite error-prone.

For most Mac applications, we instead strongly recommend using the **TGStreamMac** framework in this SDK. It reduces the complexity of managing stream connections and handles parsing of the data from NeuroSky hardware streams. To make a NeuroSky-enabled application, all you need to do is to import the **TGStreamMac** 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 TGStreamMac framework include:

- Can only communicate with one attached Neurosky Hardware Enabled accessory at a time

Configure Your Environment

Add the `TGStreamMac` framework to your 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 `TGStreamMac.framework` from the `libs/` folder of the SDK, and click **Add**

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:

```
streamMac=[TGStreamForMac sharedInstance];
streamMac.delegate=self;
```

This sets up the shared `TGStreamForMac` instance. The delegate can be set to any class that conforms to 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 : NSViewController
```

Simply modify the definition in the following way:

```
@interface ViewController : NSViewController<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 API Reference in a section further below for a full list of the supported data.

```
#pragma mark - TGStream delegate
-(void) onDataReceived:(NSInteger)datatype data:(int)data obj:(NSObject *)obj {

    NSString *str_type = @"default";

    // distinguish different types of mind data here
    switch (datatype)
    {

        case 0xBA:
        case 0xBC:
            NSLog(@"Command-%02lx: %d", (long)datatype, data);
            break;
        case MindDataType_CODE_POOR_SIGNAL:
            str_type = @"MindDataType_CODE_POOR_SIGNAL";
            break;
        case MindDataType_CODE_RAW:
             str_type = @"MindDataType_CODE_RAW";
            break;
        case MindDataType_CODE_ATTENTION:
             str_type = @"MindDataType_CODE_ATTENTION";
            break;
        case MindDataType_CODE_MEDITATION:
             str_type = @"MindDataType_CODE_MEDITATION";
            break;
        case MindDataType_CODE_EEGPOWER:
             str_type = @"MindDataType_CODE_EEGPOWER";
            break;
        default:
            break;
    }

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

    if (obj)// if obj exists, got EEGPower data here
    {
        TGSEEGPower *kEEGPower=(TGSEEGPower *)obj;
        NSString *TGSEEGPowerSrting=[ NSString stringWithFormat:@" 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
",
                                     kEEGPower.delta,
                                     kEEGPower.theta,
                                     kEEGPower.lowAlpha,
                                     kEEGPower.highAlpha,
                                     kEEGPower.lowBeta,
                                     kEEGPower.highAlpha,
                                     kEEGPower.lowGamma,
                                     kEEGPower.lowGamma];
```

```
        NSLog(@"%@",TGSEEGPowerSrting);
    }
}
```

Note: While datatype is "MindDataType CODE POOR SIGNAL", data is the value of poor signal status. 0 means poor and 200 means great.

Handle 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;{

    NSString *connectState;

    switch (connectionState) {

        case STATE_CONNECTED:
            connectState=@"1 - STATE_CONNECTED";
            break;

        case STATE_WORKING:
            connectState=@"2 - STATE_WORKING";
            break;

        case STATE_STOPPED:
            connectState=@"3 - STATE_STOPPED";
            break;

        case STATE_DISCONNECTED:
            connectState=@"4 - STATE_DISCONNECTED";
            break;

        case STATE_COMPLETE:
            connectState=@"5 - STATE_COMPLETE";
            break;

        case STATE_RECORDING_START:
            connectState=@"7 - STATE_RECORDING_START";
            break;

        case STATE_RECORDING_END:
            connectState=@"8 - STATE_RECORDING_END";
            break;

        case STATE_FAILED:
            connectState=@"100 - STATE_FAILED";
            break;

        case STATE_ERROR:
            connectState=@"101 - STATE_ERROR";
```

```
            break;

        default:
            break;
    }

    NSLog(@"Connection States:%@",connectState);

}
```

Start and Stop the Data Stream

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

```
- (IBAction)initBT:(id)sender {
    [streamMac startStream];

}

- (IBAction)initFile:(id)sender {

    NSBundle *mainBundle = [NSBundle mainBundle];

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

    [streamMac initStreamWithFile:filePath];
    checkSum=0;
}
```

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

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

    [streamMac stopStream];

}
```

Log Messages

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

```
    [streamMac enableLog:true];
```

Further Considerations

- Your app should provide a consistent user experience to your users by following the guidelines described in NeuroSky Developer Application Standards. This will allow your users to quickly become familiar with using the NeuroSky technology with your app.

Stream Mac API Reference

Overview

The `TGStreamForMac` class handles connections between a Neurosky Hardware accessory and Mac.

Configuration

Get the Shared Stream Manager

- `+ (TGStreamForMac *)sharedInstance`
- `+ (TGStreamForMac *)sharedInstance:(NSString *)portPath`

Start Stream

- `- (void) initStreamWithFile:(NSString *)filePath;`
- `- (void) startStream;`

Stop Stream

- `- (void) stopStream;`

Send Command

- `-(void)sendCommand:(SendCMDType)cmdType`

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 TGStreamForMac.

```
@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 `TGStreamForMac` object for the OSX-based device.

```
// default portPath is "/dev/tty.MindWaveMobile-SerialPo".
+ (TGStreamForMac *)sharedInstance

//init with port path
+ (TGStreamForMac *) sharedInstance:(NSString *)portPath;
```

Return Value

The shared `TGStreamForMac` object.

Note

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

Instance Methods

initStreamWithFile

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

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

Note

The purpose of initConnectWithFile 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 `stopStream` method.

startStream

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

```
- (void) startStream;
```

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 `stopStream` method.

stopStream

Close the data stream that is opening.

```
- (void) stopStream;
```

Note

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

sendCommand

Send command to MWM to configure it.

```
typedef NS_ENUM(NSUInteger, SendCMDType) {
    SendCMDTypeNone = 0,
    SendCMDTypeReadBaud,
    SendCMDTypeChangeBaudTo_576000,
    SendCMDTypeChangeBaudTo_1152000,
    SendCMDTypeReadNotch,
    SendCMDTypeChangeNotchTo_50,
    SendCMDTypeChangeNotchTo_60
};

-(void)sendCommand:(SendCMDType)cmdType;
```

Note: Set the filter type with "SendCMDTypeChangeNotchTo 50" or "SendCMDType-ChangeNotchTo 60". The value depends on the alternating current frequency(or AC Frequency) of your country or territory, for examle, USA 60HZ, China 50HZ. If you are not sure about it, please google “alternating current frequency YOUR LOCATION”. This function only support MindWave Mobile1.5. This is an asynchronous call, it will return immediately. Please confirm the state is connected before calling this funciton.

If call this function success, MindWave Mobile 1.5 will stop send out data for 0.8 second. If you want to check

Note

This method is supported by MWM1.5 and later.

setRecordStreamFilePath

set the record stream with default path. The default path is Documents/TGS_log/yyyy-MM-dd-HH-MM-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 TGStreamForMac.

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

enableLog

Enable log of TGStreamForMac,SDK log has prefix `Stream SDK MAC—` .

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

Enum

Declared in `TGStreamEnum.h`

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,

};
```

The MindDataType is used to distinguish different types of mind data which are used in the **onDataReceived** delegate function. And it corresponds to the EEG data. The follow code shows how to use it.

```
#pragma mark - TGStream delegate
-(void) onDataReceived:(NSInteger)datatype data:(int)data obj:(NSObject *)obj {

    NSString *str_type = @"default";

    // distinguish different types of mind data here
    switch (datatype)
    {
        case MindDataType_CODE_POOR_SIGNAL:
            str_type = @"MindDataType_CODE_POOR_SIGNAL";
            break;
        case MindDataType_CODE_RAW:
             str_type = @"MindDataType_CODE_RAW";
            break;
        case MindDataType_CODE_ATTENTION:
             str_type = @"MindDataType_CODE_ATTENTION";
            break;
        case MindDataType_CODE_MEDITATION:
             str_type = @"MindDataType_CODE_MEDITATION";
            break;
        case MindDataType_CODE_EEGPOWER:
```

```
            str_type = @"MindDataType_CODE_EEGPOWER";
            break;

        case 0xBA:
        case 0xBC:
            str_type = [NSString stringWithFormat:@"Command-%02lx",(long)datatype];
            break;
        default:
            break;
    }

    NSLog(@"%@:%d data:%d",str_type,(int)datatype,data);
}
```

ParserType

```
typedef NS_ENUM(NSUInteger, ParserType){

    PARSER_TYPE_DEFAULT = 0, //sampleRate 1024 for default Parser

};
```

Connection Status

```
typedef NS_ENUM(NSUInteger,ConnectionStates){

    STATE_CONNECTED = 1,       // Neurosky Hardware Connected

    STATE_WORKING = 2,        // Neurosky Hardware Stream Or File Stream is Processing

    STATE_STOPPED = 3,        // when you tear down Neurosky Hardware Stream Or File Stream

    STATE_DISCONNECTED = 4,   // Neurosky Hardware DisConnected

    STATE_COMPLETE = 5,      // 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 `TGStreamForMac` 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 OSX-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` — playload byte data
- `length` — length of playload
- `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;
```

Parameters

- `datatype` — type of data, reference to **MindDataType**
- `data` — value of data
- `obj` — TGSEEGPower Object

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

- `delta` — value of Delta wave
- `theta` — value of Theta wave
- `lowAlpha` — value of Low Alpha wave
- `highAlpha` — value of High Alpha wave
- `lowBeta` — value of Low Beta wave
- `highBeta` — value of High Beta wave
- `lowGamma` — value of Low Gamma wave
- `middleGamma` — value of Middle Gamma wave