

# ThinkGear Connector Development Guide

## Introduction

The ThinkGear Connector (TGC) is an executable that provides a daemon-like service that manages communications with ThinkGear devices, such as the MindWave and MindWave Mobile, that are connected to the computer. The TGC runs continuously in the background, and keeps an open socket on the local user's computer, allowing applications to connect to it and receive information from the connected ThinkGear devices. This means that any application in any language that can open and read from sockets (such as Flash's ActionScript3 and other scripting languages) can connect and receive data from a headset.

This update indicates that TGC implements the ThinkGear Socket Protocol.

## Getting Started

The first step is to follow the standard end-user instructions for installing and running the TGC as described in the [ThinkGear Connector User Guide](#). After following the instructions there, return here for more information about developing.

## Connecting to TGC

Once the TGC is up and running on the system, the next step is to open a socket connection to it. Refer to the socket API documentation for your application's platform/programming language for details on how to open a socket connection.

Open a socket connection to the TGC using the following configuration:

<b>Host address</b>	127.0.0.1
<b>Port</b>	13854
<b>Protocol</b>	TCP

# Reading Data from TGC

Once your application has opened a socket connection to the TGC, it should be able to read MindWave and MindWave Mobile data from the socket connection. In order to parse and understand the data stream, please refer to the [ThinkGear Socket Protocol](#) document. Once your application is properly parsing the data stream according to the ThinkGear Socket Protocol, it has access to all the data coming from the ThinkGear headset! Congratulations, your application can now respond to brainwave data and is BCI-enabled!

By default, **Flash applications** that are run from the user's local filesystem (e.g. as a download) are run in a security sandbox which prevents them from accessing Flash Sockets and thus the ThinkGear Connector.

Developers deploying downloadable Flash applications should include instructions in a user manual showing users how to exclude folders or files from being run in the security sandbox. This has to be done via a Flash applet on Adobe's website, which can be [found here](#).

## Connecting to ThinkGear Connector using .NET

### Prerequisites

- Visual Studio 2010 or greater
- Jayrock JSON library <https://code.google.com/p/jayrock/>
- ThinkGear Connector (TGC)

### HelloTGC Example

This example will show the following:

1. Create a TCP socket connection to the ThinkGear Connector
2. Configure the the TGC to output JSON and raw EEG
3. Parse the JSON output

Create a new Visual C# console application

Add references to Jayrock.dll and Jayrock.Json.dll to the project and import the following namespaces:

```
using System.IO;
using System.Net;
using System.Net.Sockets;
using Jayrock.Json;
using Jayrock.Json.Conversion
```

Create the following objects

```
TcpClient client;
Stream stream;
byte[] buffer = new byte[2048];
int bytesRead;
// Building command to enable JSON output from ThinkGear Connector (TGC)
byte[] myWriteBuffer = Encoding.ASCII.GetBytes(@"{"enableRawOutput": true,
"format": "Json"}");
```

Create a TcpClient and try to connect

```
try {
    client = new TcpClient("127.0.0.1", 13854);
    stream = client.GetStream();

    System.Threading.Thread.Sleep(5000);
    client.Close();
} catch(SocketException se) {}
```

Send the configuration packet to TGC

```
try {
    client = new TcpClient("127.0.0.1", 13854);
    stream = client.GetStream();

    // Sending configuration packet to TGC
    if(stream.CanWrite) {
        stream.Write(myWriteBuffer, 0, myWriteBuffer.Length);
    }

    System.Threading.Thread.Sleep(5000);
    client.Close();
} catch(SocketException se) {}
```

Keep reading packets and parse the packets

```
try {
    client = new TcpClient("127.0.0.1", 13854);
    stream = client.GetStream();

    // Sending configuration packet to TGC
    if(stream.CanWrite) {
```

```
        stream.Write(myWriteBuffer, 0, myWriteBuffer.Length);
    }

    if(stream.CanRead) {
        Console.WriteLine("reading bytes");

        // This should really be in it's own thread
        while(true) {
            bytesRead = stream.Read(buffer, 0, 2048);
            string[] packets = Encoding.UTF8.GetString(buffer, 0,
bytesRead).Split('\r');
            foreach(string s in packets) {
                ParseJSON(s.Trim());
            }
        }

        System.Threading.Thread.Sleep(5000);

        client.Close();

    } catch(SocketException se) {}
```

## Miscellany

### Policy Files

When SWF files open a socket connection, SWF will typically automatically request a `crossdomain.xml` file from the TGC by sending the following XML to the TGC:

```
<policy-file-request/>
```

In response, TGC will automatically write the following XML to the socket to complete the handshake:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <allow-access-from domain="*" to-ports="*" />
</cross-domain-policy>
```

This all usually occurs automatically when the socket is opened in SWF, and requires nothing special

on the SWF programmer's part. The SWF program can then communicate over the socket (read and write) as normal.

If no policy file is requested by the client, then no policy file will be sent by TGC; TGC will still function as normal in this case.

By default, the TGC allows *any* remotely-running SWF file to access it, though it will only listen on a specific address and port. For details on Flash's policy file implementation, refer to [this link](#).

From:

<http://developer.neurosky.com/docs/> - **NeuroSky Developer - Docs**

Permanent link:

[http://developer.neurosky.com/docs/doku.php?id=thinkgear\\_connector\\_development\\_guide](http://developer.neurosky.com/docs/doku.php?id=thinkgear_connector_development_guide)

Last update: **2014/07/01 20:09**



### **Warnings and Disclaimer of Liability**

THE ALGORITHMS MUST NOT BE USED FOR ANY ILLEGAL USE, OR AS COMPONENTS IN LIFE SUPPORT OR SAFETY DEVICES OR SYSTEMS, OR MILITARY OR NUCLEAR APPLICATIONS, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE ALGORITHMS COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. YOUR USE OF THE SOFTWARE DEVELOPMENT KIT, THE ALGORITHMS AND ANY OTHER NEUROSKY PRODUCTS OR SERVICES IS "AS-IS," AND NEUROSKY DOES NOT MAKE, AND HEREBY DISCLAIMS, ANY AND ALL OTHER EXPRESS AND IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTIES ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL NEUROSKY BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS OR INCOME, WHETHER OR NOT NEUROSKY HAD KNOWLEDGE, THAT SUCH DAMAGES MIGHT BE INCURRED.