Delsys Application Program Interface

User's Guide

MAN-033-1-0

TABLE OF CONTENTS

1 Important Information

1.1 Intended Use

The Delsys Application Program Interface (API) is a software development tool to be used with the Trigno Wireless Biofeedback System. The API is intended to be used as a software component in a finished software product for third-party applications. The function of the API is to manage data transfer and to facilitate communication between the Trigno Biofeedback System and third-party software applications. The Delsys API is designed to work exclusively with the Trigno Biofeedback System and is not intended to be used in diagnostic or safety-critical applications.

The Trigno™ Wireless Biofeedback System is a battery-powered biofeedback device that enables researchers and clinicians to acquire EMG and related signals from subjects for biofeedback and research purposes. They are intended for relaxation training and muscle reeducation. Interpretation of the EMG and supporting signals by a qualified individual is required.

Please refer to the Trigno Wireless Biofeedback System User Guide for additional important information.

⚠ Please refer to the Trigno Wireless Biofeedback System User Guide for additional important information.

1.2 Technical Service and Support

For information and assistance, please visit:

www.delsys.com

Contact us:
E-mail: support@delsys.com
Telephone: (508) 545 8200

1.3 System Requirements

1.3.1 Windows Developer System Requirements:

- Windows 7 and above (64 bit)
- Microsoft Visual Studio 2015 or later
- Xamarin Framework (included in Visual Studio)

1.3.2 Android Developer System Requirements:

- Windows 7 and above (64 bit)
- Microsoft Visual Studio 2015 or later
- Xamarin Framework (included in Visual Studio)
- Android 6.0 and above.

2 Delsys API Overview

The Delsys API is a .NET cross platform library created by Delsys for managing data acquisition and data transfer from Delsys hardware. The API is a tool for creating applications built on top of Delsys hardware technologies. More specifically, the Delsys API allows users to connect, configure, and collect data from Delsys Trigno Sensors. The following communication modes are currently supported:

- RF (Trigno) Mode
    - Allows streaming of data from Trigno Sensors to a PC via Delsys' proprietary RF protocol and the Trigno Base Station.
- Bluetooth (BLE) Mode
    - Allows streaming of data from Trigno Sensors directly to an Android tablet (or other mobile device).

This guide provides a technical overview of the modules constituting the API, and includes information on how to properly utilize the functionality of the API for an RF (base station) setup.

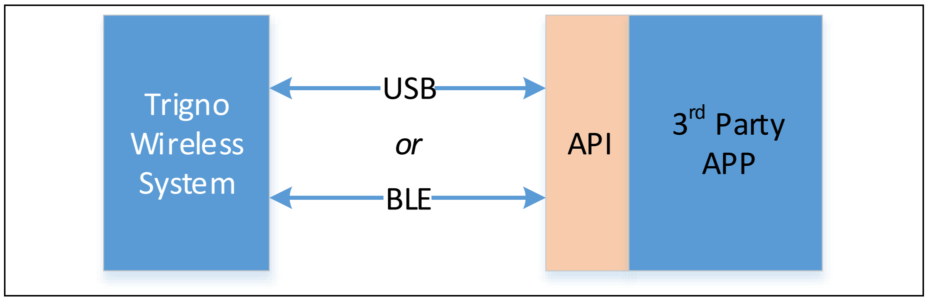

Figure 1: Data flow and API, Trigno System, and third-party app.

User interaction with the API can be represented by the functional block diagram shown below. Software applications use the API as a communication layer to configure various Trigno hardware setups. The main block is the Pipeline Controller, which manages every aspect of the API. Developers interact with the Pipeline Controller through a series of interfaces to send commands, and receive data and system state information.

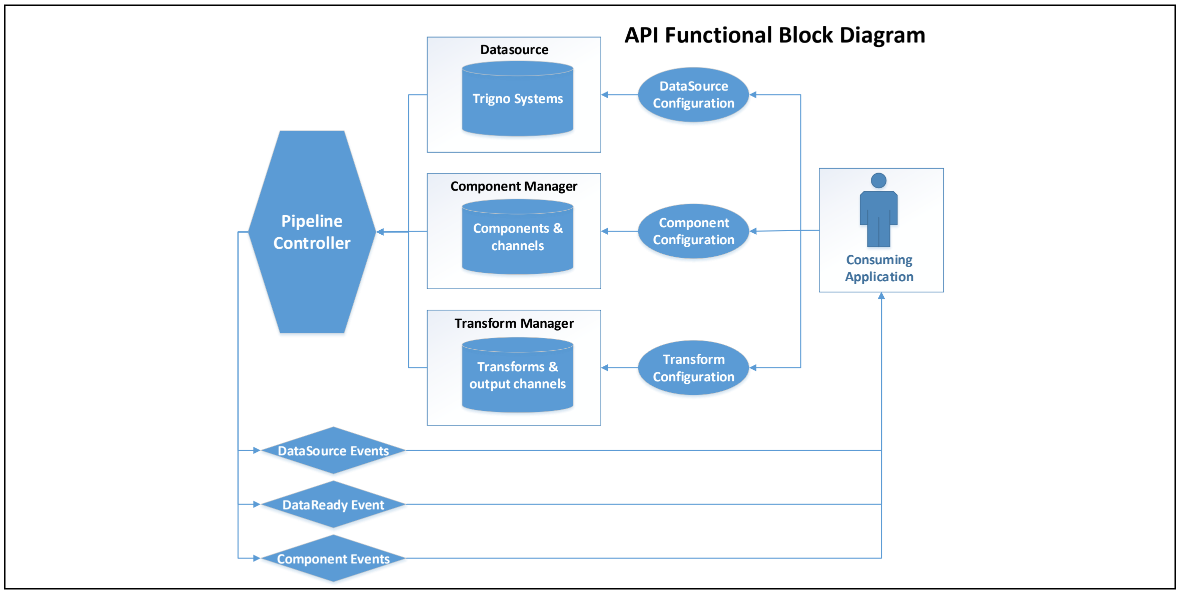

Figure 2: API Function Block Diagram showing key elements with data and communication flow between the software application and the Trigno System (data source).

3 Using the API

Prior to beginning development with the Delsys API, please read the Trigno Wireless Biofeedback System User Guide (MAN-031) for information about using the Trigno hardware and its intended uses. It is also recommended that developers download and install the EMGworks Software Package and familiarize themselves with it to learn how users are expected to interact with the system.

To use the Trigno System via the API, the third-party software application must perform these basic steps:

- Connect to the Trigno System.
- Configure the hardware (pair sensors, etc.). Please see the next section “Configuring the Trigno Hardware” for additional details.
- Start the Trigno System.
- Trigger the system (optional). Send a start trigger to the Trigno Base Station. For more information on this please see the Trigno Wireless Biofeedback System User Guide (MAN-031).
- Receive Data

For a short, practical tutorial on how to perform all of those steps, please follow the API Quick Start Guide (MAN-032) included in the ‘docs’ folder of the API software package.

4 Definitions

4.1 Pipeline

The ‘pipeline’ defines the flow of data from a single Data Source, ending with data being presented to the user. For example, in RF mode, a pipeline would collect data from Trigno Hardware, process the data, and make the output available to the user. The pipeline contains an internal state machine which controls the flow of data.

The pipeline contains various useful events which can be used to collect data, and receive signals on system attributes.

Each pipeline exposes its own Component Manager and Transform manager (see details below).

4.2 Pipeline Controller

The Pipeline Controller is responsible for coordinating and monitoring all available pipelines. **Note:** Delsys API currently supports only one pipeline inside the pipeline controller.

4.3 Pipeline States

Pipeline states are available for viewing, and give developers a chance to see at what stage in the data collection process they are in. The most important states are:

4.3.1 Off

- no sensors have been detected
- subsystem is un-configured

4.3.2 Connected

- sensors are detected, subsystem configured
- sensors and transforms un-configured

4.3.3 Armed

- All configurations have been finalized

4.3.4 Running

- Data collection is underway

The pipeline state machine is shown in the diagram below:

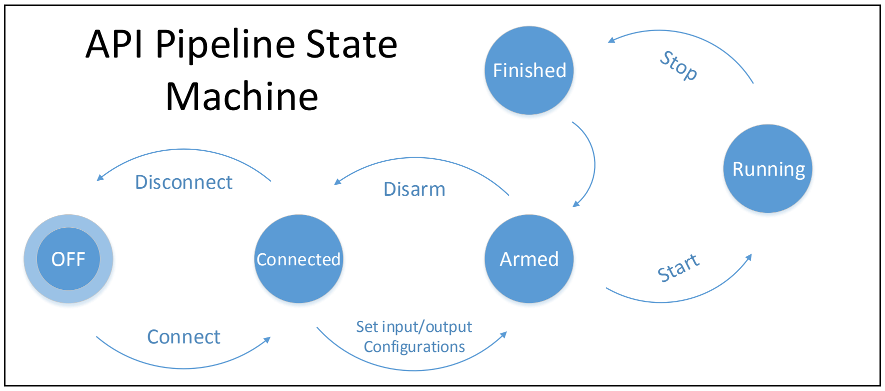

Figure 3: API Pipeline State Machine diagram.

4.4 Component

A component defines a hardware device from which data are collected. A component will contain a variety of attributes, as well as specific data channels.

An example of a component is a Trigno Sensor, which has a set of configuration options, as well as a defined number of channels (EMG, ACC, etc.).

4.5 Channel

A channel defines a singular data stream, which is sent at regular intervals from a component. Channels have specific units (mV,g,etc.).

4.6 Data Source

Data sources define the hardware platforms by which component data are aggregated. Example of this are a Trigno Base Station, or an Android Tablet's Bluetooth subsystem. In order for the Delsys API to function, it must detect, and be connected to a supported Data Source.

4.7 Transform

A transform can be visualized as a "data filter". Data comes in one side via a defined number of input channels, is transformed in some manner, and exists the opposite side via a defined number of output channels. What shape the transformation takes is defined by the transform applied.

Users will be able to apply a pre-existing transform, or define their own.

Note: the Delsys API currently supports only the default transform type (Raw Data).

4.8 Component Manager

The Component Manager manages the configuration and the interaction of components. Two Component Managers are supported:

4.8.1 RF Manager:

- The Component Manager in RF mode.

4.8.2 BT Manager:

- The Component Manager in Bluetooth mode.

The component manager provides useful tools for interacting with components:

- Viewing and setting component configurations
- Viewing component attributes (available sample modes, current sample modes etc.).
- Viewing component channels.
- Viewing component properties (battery life, signal strength etc.).

4.9 Transform Manager

This administers all Transforms and Transform Configurations. It allows users to view active transforms, add new transforms, and remove transforms.

4.10 Configurations

This section can be split up into 3 main configuration types. Each of these types defines how data pass from the Data Source to the user. **Note:** configurations must always be applied before data can be collected.

4.10.1 Data Source Configuration

Defines attributes related to the Datasource. In RF mode, the configuration allows the Trigno Base to be set to specific configurations, as well as the setting and configuring of the Delsys Trigger Module.

4.10.2 Component Configuration

Defines how a component will behave during data collection. Components will be different depending on mode.

- Configures the state of DIO pins on sensor.
- Allows user to select from a number of sampling/channel configurations and layouts. These sampling modes may be different depending on whether operating in RF or Bluetooth mode. Sample modes may also vary depending on sensor type.

4.10.3 Output Configuration

This configuration defines how data appear to the user. Transform output channels are selected for viewing and the order in which they are presented can be configured. This approach provides flexibility in configuring the manner in which data are collected.

5 Technical Overview

The Delsys API uses the following Objects to abstract the hardware layer from Subscribers, and to provide a robust and easy-to-use interface.

5.1 Components

At the highest level, components can be thought of as virtual collections of channels with specific properties and subroutines to modify data within those channels. A component can be formed in a variety of ways.

Components are presented as a generic collection, which can accept various component subsets. For example, if a Trigno Base has been connected, the list can be populated with connected Trigno Sensors. In the case of a generic DAQ device, the list may be populated with Components representing specific groupings of channels.

Many of the properties, attributes and information of components are open to view from the Component List.

The Component List also acts as a configurable interface of the API. Users can set their own components and transforms that are bound to them.

The Component List represents a dynamic collection of components, providing the user with access to the following.

- Type Information: Attributes relative to the sensor types.
- Component Data: Synchronized chunks of acquired data passed through a series of data transformation steps.
- Channel Information: Channel structure, setup, attributes.

5.1.1 Component Types

Component Types are groupings of "bound" channels, and as such are able to be created, and extended on the fly. These channels may or may not belong to discrete devices, such as Trigno Sensors, or they could be a combination of discrete devices.

There are specific preset types of components created for the first production version of the API.

5.1.1.1 SensorTrignoRf

This class is derived from the Component base class, and serves as a base class for all Trigno Sensor types.

5.1.1.2 Trigno Sensor Types

Each Trigno Sensor type is a class derived from the SensorTrignoRf class. They are as follows:

Class Name	Sensor Type
SensorAvanti	14
SensorEMGLegacy	0
SensorSpringContactLegacy	1
SensorSnapLeadLegacy	2
SensorStandardLegacy	3
SensorFSRLegacy	4
SensorEKGLegacy	5
SensorLoadCellLegacy	6
SensorGoniometerLegacy	7
SensorMiniLegacy	9
SensorAnalogInputLegacy	10
SensorImLegacy	11
SensorDRLegacy	12
SensorTriggerLegacy	13

5.1.2 Trigno Sensor Properties and Methods

Each instantiation of a Trigno Sensor as a component contains properties and methods which are used to configure the sensor and to inform the consumer of certain sensor properties.

5.1.3 Component Channels

These are channels form the building blocks out of which Components are defined, with each channel defining the properties of a specific stream of data, along with the parameters that govern the stream. Channels are divided into the following hierarchy, which covers a wide variety of current types, while also being flexible and allowing for extensions and new channel types to be created and added.

5.1.3.1 Digital Channels

These represent the channels of a digital device. These channels will not require operations such as voltage scaling and offset calculations.

5.1.3.2 Analog Channels

These channels are voltage based and may need to have scaling and offsets applied. Currently most smart sensor channels, especially EMG, fall under this category.

5.2 Pipeline Controller

This is an instance of the Pipeline Control module. Consuming applications can configure Data Sources, and control the Dataflow Pipeline from this instance.

5.2.1 Collection Data Ready Event

The data ready event is the main sink by which a subscriber can get data after initiating a data collection. This event represents the output of the live data portion of the API pipeline.

5.3 Pipeline Controller Module (PCM)

The pipeline controller represents the control center for the API, issuing commands and interfacing with hardware devices.

Most importantly, the PCM is responsible for configuring Data Sources, which define the precise interaction between software buffers, and data being received from hardware. The PCM will allow the API to access the hardware that will communicate with the API and the supporting application. This process is abstracted to the user, with the exception of a few commands.

5.3.1 Pipeline State Machine

To collect data, the PCM is responsible for managing the data collection pipeline, with data undergoing various transformations. A state machine allows the PCM to transition the pipeline to various states of activity. In this way the PCM is able to go from initialization, transitioning until data collection is achieved.

For more information about the Pipeline State Machine, see the state machine design diagram.

5.3.2 Input Configurations

Subscribers wishing to utilize the API pipeline must specify an input configuration. A configuration consists of collections of properties that define all stages of DataSource functionality. Without specifying a configuration, a subscriber will not be able to enter data collection mode.

5.3.3 Output Configurations

Subscribers wishing to utilize the API pipeline must specify an output configuration. The internal pipeline transforms input data into various forms, and the number of output streams may no longer correspond to the number of input channels. An output configuration is a mapping that allows the system to successfully package data for the user in an intelligible fashion.

5.3.4 External Commands

The PCM will receive external API commands issued by the consumer. These commands will configure, activate, deactivate, and start/stop data acquisition. It will set up any basic structures needed by the API such as the Component Manager, and the Transform Manager. These commands can include:

5.3.4.1 Command List

1. Get Datasource
2. Configure Datasource
3. Get Components
4. Transition Pipeline
5. Subscribe to DataReady event.
6. Set or unset triggers.

6 Programming the API

Essential methods and properties of the API are described below. The list is not exhaustive – see the Sandcastle documentation for an exhaustive description of all API elements – but contains information necessary for proper understanding of the fundamental flow and operation of the API.

6.1 Pipelines

The pipeline is the most fundamental object of the API. To simplify the calls described below, the following reference is made.

```
var myPipeline = PipelineController.Instance.PipelineIds[0].
```

6.1.1 Connecting and Disconnecting

Connecting and disconnecting the Pipeline is achieved as follows:

```
// Reference the sample programs for when these operations should occur.
myPipeline.Connect();
myPipeline.Disconnect();
```

6.1.2 Data Collection

Data collection is handled via a delegate method. The following code demonstrates the operation to initiate data collection, and is followed by a delegate method which gives access to the collected data.

```
// Set up the delegate for when data is collected and ready, the runtime of the data
// collection (120 seconds), and then begin data collection.
myPipeline.CollectionDataReady += DataReadyDelegate;
myPipeline.RunTime = Convert.ToDouble(120);
myPipeline.Start();
// . . .
// The delegate that fires whenever data is ready.

void DataReadyDelegate(object sender, ComponentDataReadyEventArgs e)
{
    // Prints out the number of channels of data we received.
    Console.WriteLine(e.Data.Length);
}
```

6.1.3 Data Source Information Dictionary

There are several properties that can be extracted from the Pipeline. These properties are accessible via a string-to-string dictionary in the Pipeline. Each dictionary key and a description of its associated value are listed in the table below. Accessing this information can be done by the call below (after a Pipeline has been set up), replacing **Key** with whatever key string you wish to query the value of.

```
var myDSInfo = myPipeline.DataSourceInfo[Key];
```

Key	Value Description
"Base ID"	The base identification number.
"Hardware Name"	The name of the base as identified by Windows.
"Hardware Address"	The USB port address of the base.
"Firmware Version"	The firmware version of the base.

6.2 Sensors

Sensors have a robust selection of settable properties and attributes, accessible by using a reference to a sensor such as the one below (which assumes at least one component has been allocated.)

```
var mySensor = PipelineController.Instance.PipelineIds[0].TrignoRfManager.Components[0];
```

6.2.1 Sensor Type

The sensor's type (e.g 14 is an Avanti sensor.)

```
int sensorType = sensor.Properties.Type;
```

6.2.2 Sensor Serial

The unique Serial ID of the sensor.

```
int sensorSID = sensor.Properties.Sid;
```

6.2.3 Sensor Firmware

The firmware version of the sensor.

```
string sensorFW = sensor.Properties.Fw;
```

6.2.4 Sensor Mode

The configured mode of the sensor.

```
int sensorMode = sensor.Configuration.SampleMode;
```

6.2.5 Sensor Channels

The sensor's channels each have properties that may be queried.

```
var myChan = mySensor.TrignoChannels[0];
```

6.2.5.1 Sampling Rate

The sample rate is the quotient of the samples per frame and frame interval. All three of these properties can be retrieved.

```
int sampleSize = myChan.SamplesPerFrame;
float frameInterval = myChan.FrameInterval;
// Equivalent to sampleSize/frameInterval
float sampleRate = myChan.SampleRate;
```

6.2.5.2 Units

The units of the values being output by the channel

```
var units = myChan.Units;
```

6.2.6 Allocation/Deallocation

Allocating a sensor and deallocating a sensor is done via the Pipeline's TrignoRfManager.

```
var componentManager = myPipeline.TrignoRfManager;
// Allocates the sensor.
componentManager.SelectComponentAsync(mySensor);
// Deallocates the sensor.
componentManager.DeselectComponentAsync(mySensor);
```

6.3 References

6.3.1 Sandcastle Documentation

The Sandcastle Documentation included with the API package contains auto-generated descriptions of individual classes, methods, and properties. The sources indicated below are convenient starting points, but the Sandcastle Documentation provides a full guide to API functions and features.

6.3.2 API Quick Start Guide

The APi Quick Start Guide provides a short introduction and example of how the API can be implemented. This is an RF example, and requires a base station to run. It illustrates connecting and disconnecting the pipeline, pairing, configuring, and getting data from sensors. It is recommended to begin by following this guide, and referencing the other documentation (this User Guide included) to gain a fuller understanding of the API.

6.3.3 APIExamples Solution

The APIExamples Solution is a Visual Studio Solution which contains the BasicExample project. This project is a separate, but similar program to that described in the API Quick Start Guide PDF. The Solution will include more examples in future releases.