Smart Jacket App

BLE Service and Protocol Specification version 1.3.1

This file contains a succinct specification for the BLE service implemented by the Smart Jacket and supported in the app. As complex data is passed between service and app, a JSON format for the exchange has also been defined, with a subset supported in the app.

Smart Jacket BLE Service

The following is a description of the BLE service supported by the app. Following are the UUIDs for the service and each characteristic.

Name	**Type**	**Perms.**	**Behaviour**
Timestamp	4 bytes	read	The current time on the jacket, for plan synchronization.
Plan Name	String	read	The name of the current plan, if any.
State	5 bytes	read	Current status of the Jacket, including training. Each byte corresponds to the following:
	Plan State		The current state of training plan: None, Ready, Started, Paused, Ended as values from 0 to 4, respectively
	Current Phase		The current phase within the plan, starting at 1 (not 0)
	Phase Progress		The current progression within the phase, as percentage 0-100
	Controller Mode		Indicates the current function of the buttons: None, Music, Call as values from 0 to 3, respectively
	Battery Level		Indicates the percentage of battery remaining.
Client Command	JSON string	write	Commands sent from client to server, such as: start plan, start phase, set controller to music...
Server Message	JSON string	notify	Messages from the server, such as: Acknowledge, emergency on, emergency off, battery low...

Service/Characteristic	UUID
Smart Jacket Service	`E708EB00-AD98-4158-B7C2-A748744694AB`
Timestamp	`E708EB01-AD98-4158-B7C2-A748744694AB`
Plan Name	`E708EB02-AD98-4158-B7C2-A748744694AB`
Plan State	`E708EB03-AD98-4158-B7C2-A748744694AB`
Client Command	`E708EB04-AD98-4158-B7C2-A748744694AB`
Server Message	`E708EB05-AD98-4158-B7C2-A748744694AB`

Smart Jacket JSON Protocol

The following is the supported subset of JSON messages between app and jacket.

Upload a Training Plan

To upload a new training plan, the app sends a message to the server (jacket) containing plan data.

```
{ "co": "npl", "pl": <plan> }
```

A **plan** is composed by a name (*na*), an array of phases (*ph*) and number of repetitions or iterations (*it*).

```
{
 "na": <string>,
 "ph": [ <list of Phase> ],
 "it": <integer>
}
```

A **phase** within a plan is composed by a name (*na*), color (*co*), blink direction (*bd*), duration (*du*), blink intensity (*bi*) and training intensity (*in*).

```
{
 "na": <string>,
 "co": <string>,
 "bd": <string>,
 "du": <long>,
 "bi": <long>,
 "in": <long>
}
```

The server's reply acknowledges the plan or indicates an error. A positive reply is as follows, with a *true* acknowledgement (*ack*) and fixed status code (*rs*).

```
{ "ack": true, "rs": 200 }
```

In case of an error, the acknowledgement (ack) is negative and the code indicates the type of error.

```
{ "ack": false, "rs": <int> }
```

Tactile Control Signal

Activating the jacket's tactile controls will send a signal message to the app. The signal message contains the signal type (*si*). Only signals for buttons (*"btn"*) are supported. These contain a button type (*bt*) indicating which button/action was taken by the wearer.

```
{ "si": "btn", "bt": <string> }
```

The button/action string does not necessarily follow a convention. Rather, it is matched by the app to the "buttonActions" Imapping in the app's settings file.