Btraced Protocol

March 2013

Protocol Version 1.1.4

By:

mushapps

Btraced Protocol

The communication used by Btraced and the server side is described in this document. Basically the iPhone application will send the information of the trips to the server in XML format and the webservice will give back an acknoledge or error messages in JSON decribed as follows.

What's new on protocol version 1.1.4?

The detault value for the Altitude field is now set to -32000.0 instead of -1.0. Anyway to actualy check if the altitude value is valid, the user should check for the Vertical Accurracy sign. If this is negative, then Altitude is not valid. Else it is a valid altitude value.

What's new on protocol version 1.1?

The new version of the procol include a couple of new fields in the XML sent to the server, these are:

- TimeOffset: Return the local time offset in seconds from GMT
- GetTripUrl: It is a flag to ask the server the URL of the trip so it can later be used for Facebook or Twitter

Additionally, there is one more optional field in the JSON answer of the server. This is the trip URL and it is named as "tripURL".

All these new fields will be detailed in the next sections.

iPhone Application

The iPhone application, Btraced, can either send the information of the trips while tracking or in offline mode. No matter the moment, the XML format is unique in both cases.

The information to the server is sent in the body of a POST message to the webservice URL specified in the settings menu.

The sctructure of the XML is as follows.

```
<bwiredtravel>
    <model>iPhone</model>
    <devId>BC041607-1111-1111-1111-111111111111</devId>
    <username>demo</username>
    <password>xml</password>
    <timeOffset>-10800</timeOffset>
    <travel>
        <id>3</id>
        <getTripUrl>1</getTripUrl>
```

```
        <description>turn back</description>
        <length>6565.13</length>
        <time>2272</time>
        <tpoints>180</tpoints>
        <uplpoints>150</uplpoints>
        <point>
            <id>353</id>
            <date>1331843920.964860</date>
            <lat>-34.740783</lat>
            <lon>-58.229860</lon>
            <speed>-1.000000</speed>
            <course>-1.000000</course>
            <haccu>100.000000</haccu>
            <bat>0.65</bat>
            <vaccu>-1.000000</vaccu>
            <altitude>0.000000</altitude>
            <continous>1</continous>
            <tdist>6301.71</tdist>
            <rdist>16.62</rdist>
            <ttime>1961</ttime>
        </point>
        [...]
        <point>
            <id>382</id>
            <date>1331844232.556679</date>
            <lat>-34.740686</lat>
            <lon>-58.229675</lon>
            <speed>0.000000</speed>
            <course>100.704447</course>
            <haccu>10.000000</haccu>
            <bat>0.65</bat>
            <vaccu>19.000000</vaccu>
            <altitude>30.426458</altitude>
            <continous>1</continous>
            <tdist>6565.13</tdist>
            <rdist>0.34</rdist>
            <ttime>2272</ttime>
        </point>
    </travel>
</bwiredtravel>
```

The full XML starts with the tag “bwiredtravel”. There we have general information of the user described by these fields:

Model: Actually it will always contains “iPhone” as the application is designed for iOS. Maybe later it would be switch to Android.

devId: Unique device identification. This can be used to identify the device for that user, or even create an account using just the device ID instead username or password.

Username: If the user set the username in settings, then it’s sent using this flag in plain text.

Password: If the user set the password in settings, then it's sent using this flag in plain text.

Time Offset: [New on version 1.1] Sends the local device time offset in seconds from GMT time.

The there would be information for the selected trip (or current trip). It starts with the "travel" tag and there is general information for that trip as follows.

ID: Unique ID of the trip for that device. It's the internal identification for that trip in the Btraced application.

GetTripUrl: [New on version 1.1] If this key is present and it is set to 1, it asks to the server what's the escaped URL of the current trip. This should be a direct URL to view the full trace of the current trip. Btraced will use this information to post the trip URL into Facebook or Twitter if the user asks for it.

Description: Name of the trip

Length: Total travel length in meters

Time. Total travel time in seconds

Tpoints: The total amount of points recorded for that trip

Uplpoints: Internal btraced counter of the uploaded points for that trip

And then, there are list of all the points that Btraced request to store in the server database. They are one "point" tag for each point to be saved. Each point is described with the following tags.

ID: Btraced point ID of the internal database

Date: Date of the point stored in EPOC linux time with milliseconds

Lat: Latitude of the point

Lon: Longitured of the point

Speed: Speed at that point, in m/s. Value would be -1.0 if the speed is not available

Course: Trip course at that point in degrees. Value would be -1.0 if the course is not available

Haccu: Horizontal accuracy for that point in meters

Bat: Baterry level of the iPhone (value is 0 to 1)

VAccu: Vertical accurracy at that point in meters. A negative value means vertical accuracy cannot be determinated additionally, in this case, Altitude value is not valid.

Altitude: Elevation at that point. If VAccu is negative, this value is not valid.

Continous: It's set to 1 if the trip was not stopped since the previous saved point. It will be 0 if the trip tracking has been stopped and started again. Can be used for determining each trip segment

Tdist: Total trip's distance at that point in meters

Rdist: Relative distance in meters from previous point

TTime: Total time of the trip at that point in seconds

Webservice

Every time the webservice receives a post message from Btraced, it should answer with a response in JSON format. There are multiple type of answers that can be used. They are described here.

In general lines, here are all the possible fields:

ID: Answer type (a value to identify the type of answer)

TripID: The internal btraced trip number which the server answer is related to

Points: An array with the internal brtaced points ID which were successfully saved into the server

Valid: Should be always set “true” so Btraced can idenfity a valid JSON message

Error: set to true if there was an error. Ignore this field or set to no if there wasn’t any error

Message: Here there can be any custom message string for an error

ExtraData: An array of extra data needed for special types of messages

tripURL: [New on version 1.1] If requested by the app in the xml, the server can return an URL for the current trip. This URL will later used in order to publish the trip into Twitter or Facebook. This parameter is optional, and should be sent back only if it was requested by the application to avoid lot of traffic.

Message Type 0: Success

Example:

```
{
 "id":0,
 "tripid":3,
 "points":[301,   302,   303,   304,   305,   306,   307,   308],
 "valid":true,
"tripURL":"http%3A%2F%2Fbtraced%2Ecom%2Ftripdetail%2Easp%3Ftweet
id%3DInfinite%2BLoop%21bc041607de815c048df36286b6b616f000000000”
}
```

The message type for this message is 0. It’s used to send an acknoledge signal of the saved points in the server.

In this example, the webservice is confirming that the points 301 to 308 for trip ID 3 were successfully stored in the server database.

This message will make Btraced to update the trip status and flag those points as “Uploaded” state. It should be an array of integer values, even if it’s just one point.

Here the new tripURL parameter can be seen. That’s the trip’s URL in the server. This URL will be later used to post the trip in Facebook and Twitter if the user request it.

This parameter should be only sent if the “getTripUrl” parameter appears in the XML and its value is 1.

Message Type 1: Empty username or password

Example:

```
{
 "id":1,
 "error":true,
 "valid":true
}
```

This message is used to notify an invalid username or password in case the webservice requires an user validation. If message field is set, it will be ignored as it’s a known message type. For custom messages see next messages.

Message Type 3: Upload limit reached

Example:

```
{
 "id":3,
 "extradata":[300],
 "tripid":3,
 "points":[301,  302,  303,  304,  305,  306,  307,  308],
 "error":true,
 "valid":true
}
```

In this case, it will notify Btraced that the server has an upload limit on points for each trip. In extradata array, the first value is the amount of points permitted by trip in the server.

This will show the limitation message with the limit value sent in the message.

As the limit can occur with a set of points of the upload, the trip id and points should be set to identify those points that were successfully added to the server datase. For example if the limit of points are 300, we already upload 290 points, and the XML sent by Btraced are for 30 points, then the first 10 points should be stored and reported as “Uploaded”, while the remaining 20 will be not saved in the server and neither should be in the “points” array.

Message Type >900: Custom messages

All the messages with ID number over 900, are used for custom messages not included in Btraced and that the server need to report. These messages can include any combination of the above fields. If the tripid and points field are shown, then those points will be updated to “uploaded” state in Btraced.

If the message and error fields are set to a string and true value, then the custom message will be shown in the Btraced application.

Example:

```
{
 "id":905,
 "tripid":3,
 "points":[301,  302,  303,  304,  305,  306,  307,  308],
 "error":true,
 "message":”This    is    a    custom    server    message”,
 "valid":true
}
```

With this message, Btraced will set points 301 to 308 of trip ID 3 to “uploaded” state. Also it will popup a message with the string in the JSON answer (or show the warning sign).

All the custom messages must have the same ID for the same message. Different messages should have a different ID.