Real Time Scoring (leader-board) interface for Silent Wings Studio (SWS)

The current OGN <==> SWS interface access to the Scoring engines SoaringSPOT/SeeYOU, Strepla and SGP/CountrySoaring and retrieves the competition information, information such competition name, pilots information, classes,task, way-points, etc, … with that information build the data structures able to comply with the SWS API command **eventgroups**, **event** and **trackpoints**.

We are proposing that the Scoring engines will get the information directly from the OGN, will score in real time (really finding the leader of the race) and make it available to the OGN-SWS interface.

The OGN-SWS interface will provide that data to the SWS program on the following format:

On the EVENT command modify:

event command API endpoint.

This command is used to get information about a given tracking event

Usage `host.url/event?eventid=<eventId>`

Add the following field:

```
    /** Indicate is real time scoring is active or not
      * @type Integer (bool)
      */
    "RealTimeScoring": 0,        0=No 1=yes
```

That will indicate to SWS that it can use the scorepoints command described below.

Add this new command(endpoint) to the SWS API:

scorepoints command

The viewer will do **score** requests for each track in the current event.

The reason we have chosen to do separate requests for each single track is because during a live event, some trackers tend to get lost for some period of time, before coming back again. As the viewer only requests scores for timestamps more recent than the last points that were received, the requests need to be kept separate.

The call to this command, does not need as frequent as the trackpoints command and only requested when the LEADER BOARD is requested by the user of SWS.

Usage

```
host.url/scorepoints?trackid=<trackId>&eventid=<eventId>&begin=1282834064
```

- `trackid` is the unique track ID given for each track in an event.
- `Begin` is optional and indicates that each score returned must have a timestamp equal or newer than the begin value. All timestamps are Unix Time in seconds.

When the viewer has received some scores, it will only request scores that have a timestamp newer than what has received earlier to avoid downloading the same data repeatedly.

If "begin" is omitted then it should be considered as zero, meaning sending all available scores.

The viewer will ignore any scores that it has already received.

The score-points request should return JSON like in the example below.

```
{
    /** The track ID which much match the trackID in the URL request.
      * @type String
      */
    "trackId": "1",

    /** Live data indicator
      * @type Bool
      * If the track still is ongoing (live), this value must be set to true.
      * The viewer will then continue to request new scores
      * at some interval (normally every 15 seconds or more).
      * Once the race is known to be complete, the value should be set to
      * false. This will prevent the viewer from unnecessary polling, and is
      * especially useful for archived events.
      */Array of scores
    "live": false,

    /** Event revision.
      * @type Integer
      * Should match the most recent event revision.
      * If the viewer finds a higher value here than the value previously
      * received from the event description, it will trigger requesting
      * the event description again, to update whatever did change.
      * Most common use case is probably when the start time is delayed.
      */
    "eventRevision": 1,

    /** Array of scores
      * @type Array of objects
      *
      * Note that since these arrays may become quite large, the variable
      * names have been kept as short as possible to save some
      * bandwidth.
      *
      * The array must be sorted on timestamps in increasing order.
      */
    "scores": [
        {
            /** Timestamp in Unix Time seconds
              * @type Double
              */
            "t": 1282834064,

            **Score-leader position
```

```
                         * @type Int
                         */
                       "l": 15,
                       /** Points in this race (optional valid for speed and AAT)
                         * @type Int
                         */
                       "p": 575,
                       /** Time in seconds behind the leader (optional valid for eGlide
comps)
                         * @type Int
                         */
                       "s": 1640,

                       **Score leader position for the whole championship
                         * @type Int
                         */
                       "lc": 22,
                       /** Points in this race (optional valid for speed and AAT)
                            for the whole championship
                         * @type Int
                         */
                       "pc": 575,
                       /** Time in seconds behind the leader (for eGlide comps)
i                            for the whole championship
                         * @type Int
                         */
                       "sc": 23456,

               },
```

In the case of SoaringSpot the OGN-SWS interface will the the results of the real time race at:

GET https://api.soaringspot.com/v1/classes/{id}/results

- Get class results

Requirements

Name	Requirement	Type	Description
domain	soaringspot.com		
id		int	Class id

and the competition results at:

GET https://api.soaringspot.com/v1/contests/{id}/winners

- Competition winners.

Requirements

Name	Requirement	Type	Description
domain	soaringspot.com		
id		int	Contest id

and pass the results back to SWS using the scorepoints API endpoint.