Bodkin Boats Time Trial API Documentation

Base URL: `https://files.cablepost.co.uk/bodkinBoats/`

All endpoints return JSON and accept parameters via query string. No authentication is required.

Endpoints

1. Get TT Tracks

Returns a list of all available time trial track names.

GET `/get_tt_tracks.php`

Parameters: None

Example Request:

```
GET https://files.cablepost.co.uk/bodkinBoats/get_tt_tracks.php
```

Example Response:

```
[
  "Hello World",
  "Wind Charges",
  "Sigma",
  "Moving Platforms",
  "Flappy Bodkin Boat",
  "Leads",
  "BodSurf4",
  "BodSurf2",
  "BodSurf1",
  "Invert",
  "Blazes",
  "BodSurf3"
]
```

The response is a plain JSON array of track name strings. These names are used as the `track` parameter in `get_tt_track_times`.

---

2. Get TT Track Times

Returns all recorded time trial times for a given track, sorted from fastest to slowest. Multiple entries per player are included — the full history of runs is returned, not just each player's personal best.

GET `/get_tt_track_times.php`

Parameter	Type	Required	Description
`track`	string	Yes	The track name (from `get_tt_tracks`), URL-encoded

> **Note:** Track names may contain spaces and must be URL-encoded in the query string. For example, `Hello World` should be passed as `Hello%20World`.

Example Request:

```
GET https://files.cablepost.co.uk/bodkinBoats/get_tt_track_times.php?
track=Hello%20World
```

Example Response:

```
[
  { "name": "CoolPink1271", "time": 30300 },
  { "name": "CoolPink1271", "time": 30350 },
  { "name": "NapsterNPT",   "time": 30502 },
  { "name": "CoolPink1271", "time": 30700 }
]
```

Response Fields:

Field	Type	Description
`name`	string	Display name of the player who set the time
`time`	integer	Lap/run time in milliseconds

The array is sorted ascending by `time` (fastest first). All runs for all players are included, so a player may appear multiple times.

Typical Usage Flow

To display a leaderboard showing each player's personal best on a given track:

1. Call `get_tt_tracks` to retrieve the list of available tracks.

2. Call `get_tt_track_times` with your chosen track name (URL-encoded).
3. Filter the results to keep only the fastest time per player.

```
// Example: fetch personal bests for a track and display a leaderboard
const track = "Hello World";
const url = `https://files.cablepost.co.uk/bodkinBoats/get_tt_track_times.php?
track=${encodeURIComponent(track)}`;

const response = await fetch(url);
const times = await response.json();

// Keep only each player's best (array is already sorted fastest-first)
const personalBests = new Map();
for (const entry of times) {
  if (!personalBests.has(entry.name)) {
    personalBests.set(entry.name, entry.time);
  }
}

// Display leaderboard
let position = 1;
for (const [name, time] of personalBests) {
  const seconds = (time / 1000).toFixed(3);
  console.log(`P${position++} - ${name}: ${seconds}s`);
}
```

Example output:

```
P1 - CoolPink1271: 30.300s
P2 - NapsterNPT: 30.502s
P3 - BillBodkin: 31.303s
P4 - reqsery: 32.157s
P5 - muggel_potato: 32.303s
```

Formatting Times

Times are returned in milliseconds as plain integers. Here are some common formatting patterns:

```
// Format as seconds with 3 decimal places (e.g. "30.300s")
const formatTime = ms => (ms / 1000).toFixed(3) + "s";

// Format as MM:SS.mmm (e.g. "00:30.300")
const formatTimeFull = ms => {
  const minutes = Math.floor(ms / 60000);
  const seconds = Math.floor((ms % 60000) / 1000);
  const millis  = ms % 1000;
  return `${String(minutes).padStart(2, "0")}:${String(seconds).padStart(2,
"0")}.${String(millis).padStart(3, "0")}`;
};
```