

DEVELOPER DOCUMENTATION

Sportsbook Settlement API

Settle bets by sending the game result as a simple JSON request with your API key, game ID and league ID.

BET SETTLEMENT

REST + JSON

1 ENDPOINT



```
POST https://w7api.com/sports/settle/result
  ?apikey=0
  &id={gameId}

200 OK application/json
```

METHOD

POST

BASE URL

<https://w7api.com/sports/settle/result>

WEBSITE

bet365soft.com

Code examples included for [cURL](#) [Node.js](#) [PHP](#) [Kotlin](#) [Java](#)

WELCOME

Overview

The **Sportsbook Bet Settlement API** lets you send the result of a game so that bets can be settled. You send one **POST** request that contains your API key, the game ID and the league ID in the URL, and the settled results (one item per market) as a **JSON body**. The API replies with JSON that lists the result of every item. This guide describes the request, the JSON body format, the response, and includes client code for **Node.js, PHP, Kotlin and Java**.

METHOD
POST

BODY FORMAT
JSON

ENDPOINTS
1

AUTH
apikey

At a glance

Item	Value
Request URL	<code>https://w7api.com/sports/settle/result</code>
Method	POST
Query parameters	<code>apikey</code> , <code>id</code> (game ID), <code>lid</code> (league ID)
Request body	JSON settle data, sent with <code>Content-Type: application/json</code>
Response	JSON with <code>success</code> , <code>id</code> and <code>data</code>
Website	bet365soft.com

Contents

Overview

Getting Started

Endpoint

/result - Settle bet

Settle Data

Response

Code Examples

Node.js example

PHP example

Kotlin example

Java example

Best Practices & Notes

2

3

4

4

5

6

7

8

9

10

11

12

BASICS

Getting Started

Request URL

Send every request to the URL below. Add your parameters to the query string.

```
REQUEST URL

https://w7api.com/sports/settle/result
```

How a request is built

1 Build the URL

Add apikey, id and lid to the query string.

2 Add the body

Put the settle data in the JSON body.

3 Send POST

Send it with Content-Type application/json.

Authentication

Pass your API key in the `apikey` query parameter. If you do not have your own API key, use `0` as the value.

Quick start

You can try the API with `apikey=0`. Save the example JSON body from the Settle Data section as `settle.json`, then run the request below.

```
TERMINAL - cURL

curl -X POST \
  "https://w7api.com/sports/settle/result?apikey=0&id=GAME_ID&lid=127733" \
  -H "Content-Type: application/json" \
  -d @settle.json
```

Request format

The request uses the **POST** method. Query values with spaces or special characters must be URL-encoded. The body is JSON, so set the `Content-Type` header to `application/json`.

Type	Meaning
text	A string value, such as an API key or a game ID.
number	A numeric value, such as a league ID.
json	A JSON value, sent as the request body.

REFERENCE

Endpoint

The API has one endpoint. It needs the parameters below and the JSON body described in Settle Data, and it returns the JSON response described in Response.

01

POST

/result

Settle bet

Settles the bets of a game. The game ID and league ID go in the query string, and the settled results go in the JSON body.

Parameter	Type	Required	Default	Description
apikey	text	Required	0	Your API key. If you don't have your own key, enter 0 .
id	text	Required	6a49516e5e99bd05c6485682	Game ID.
lid	number	Required	127733	League ID.
data	json	Required	-	The settle data as a JSON body (see Settle Data).

EXAMPLE REQUEST

```
POST https://w7api.com/sports/settle/result
?apikey=0
&id=6a49516e5e99bd05c6485682
&lid=127733
Content-Type: application/json

{
  "appId": "b2923cd03389635cf76815cf92c91622",
  "data": [ ...settle items... ]
}
```

About the request URL

The request URL is <https://w7api.com/sports/settle/result> . The endpoint name `/result` is already part of this URL, so you do not add it again.

REFERENCE

Settle Data

The JSON body has an application ID and a list of settled results. Each item in the list is one market with its result.

Body fields

Field	Type	Required	Description
appId	string	Required	Your application ID, for example b2923cd0...
data	array	Required	List of settled results. See the item fields below.

Item fields (inside data)

Field	Type	Required	Description
id	number	Required	Item number, for example 1.
bid	number	Required	Bet (market) ID, for example 19.
mid	number	Required	Selection ID, for example 180.
name	string	Required	Market name, for example Both Teams To Score.
value	string	Required	Result value, for example Yes or Over 0.5.

Example JSON body

JSON - settle data with 4 results

```
1  {
2    "appId": "b2923cd03389635cf76815cf92c91622",
3    "data": [
4      { "id": 1, "bid": 1, "mid": 1, "name": "Result", "value": "Barcelona" },
5      { "id": 2, "bid": 2, "mid": 7, "name": "Handicap", "value": "Freiburg @ 1" },
6      { "id": 3, "bid": 19, "mid": 180, "name": "Both Teams To Score", "value": "Yes" },
7      { "id": 4, "bid": 17, "mid": 9, "name": "Total", "value": "Over 0.5" }
8    ]
9  }
```

Example reading

The first item says: market **Result** (bid 1), selection 1, result **Barcelona** . The second item says: market **Handicap** (bid 2), selection 7, result **Freiburg @ 1** .

REFERENCE

Response

The API returns JSON. It repeats the game ID and lists the result of every item you sent.

Response fields

Field	Type	Description
success	boolean	Shows whether the request succeeded, for example true.
id	string	The game ID from your request.
data	array	List of results, one item for each item you sent. See the item fields below.

Item fields (inside data)

Field	Type	Description
id	number	Item number, the same as in your request.
bid	number	Bet (market) ID.
mid	number	Selection ID.
bet	string	Market name, for example Result. In the request this field is called <code>name</code> .
value	string	The value you sent, for example Barcelona.
data	string	Result detail, for example Winner Team - Real Madrid.
result	string	Result of the item. Allowed values: <code>Pending</code> , <code>Lost</code> , <code>Won</code> and <code>Refund</code> .

Allowed result values

The `result` field of every item is one of: `Pending`, `Lost`, `Won` or `Refund`.

Example response

```
JSON - response with 4 results

1  {
2    "success": true,
3    "id": "6a49516e5e99bd05c6485682",
4    "data": [
5      {
6        "id": 1, "bid": 1, "mid": 1,
7        "bet": "Result", "value": "Barcelona",
8        "data": "Winner Team - Real Madrid", "result": "Lost"
9      },
10     {
11       "id": 2, "bid": 2, "mid": 7,
12       "bet": "Handicap", "value": "Freiburg @ 1",
13       "data": "Winner value - 1 - 2", "result": "Lost"
14     },
15     {
16       "id": 3, "bid": 19, "mid": 180,
17       "bet": "Both Teams To Score", "value": "Yes",
18       "data": "Winner value - 1 - 2", "result": "Won"
19     },
20     {
21       "id": 4, "bid": 17, "mid": 9,
22       "bet": "Total", "value": "Over 0.5",
23       "data": "Winner value - 3", "result": "Won"
24     }
25   ]
26 }
```

INTEGRATION

Code Examples

Each example sends the settle data from the Settle Data page with a POST request and reads the response. Replace the game ID and league ID with your own values, and set your key in the `SPORTSB00K_API_KEY` environment variable (or keep `0`).

Node.js

Node.js 18 or newer. No packages to install.

PHP

PHP 7.4 or newer with the cURL extension enabled.

Kotlin

Kotlin with the OkHttp library (Gradle dependency shown in the file).

Java

Java 17 or newer (text blocks). Uses the built-in `java.net.http` client.

Call the API from your server

Settlement requests should come from your backend. Do not put your API key or settlement code in a public website or a mobile app.

CODE EXAMPLE

Node.js

 Node.js 18 or newer. No packages to install.Run: `node settle-client.js`

```
1 // settle-client.js - Node.js 18+ (built-in fetch, no packages needed)
2 const SETTLE_URL = "https://w7api.com/sports/settle/result";
3
4 const settleData = {
5   appId: "b2923cd03389635cf76815cf92c91622",
6   data: [
7     { id: 1, bid: 1, mid: 1, name: "Result", value: "Barcelona" },
8     { id: 2, bid: 2, mid: 7, name: "Handicap", value: "Freiburg @ 1" },
9     { id: 3, bid: 19, mid: 180, name: "Both Teams To Score", value: "Yes" },
10    { id: 4, bid: 17, mid: 9, name: "Total", value: "Over 0.5" },
11  ],
12 };
13
14 /** POST the settle data. gameId = "id", leagueId = "lid". */
15 async function settleBet({ apiKey = "0", gameId, leagueId, body }) { // returns the JSON response
16   const query = new URLSearchParams({ apikey: apiKey, id: gameId, lid: leagueId });
17   const res = await fetch(`${SETTLE_URL}?${query}`, {
18     method: "POST",
19     headers: { "Content-Type": "application/json" },
20     body: JSON.stringify(body),
21   });
22   if (!res.ok) throw new Error(`HTTP ${res.status}`);
23   return res.json(); // { success, id, data: [...] }
24 }
25
26 // ----- usage -----
27 settleBet({
28   apiKey: process.env.SPORTSB00K_API_KEY || "0",
29   gameId: "6a49516e5e99bd05c6485682",
30   leagueId: 127733,
31   body: settleData,
32 })
33 .then((result) => console.log(result.success, result.data))
34 .catch(console.error);
```

CODE EXAMPLE

PHP

 PHP 7.4 or newer with the cURL extension enabled.Run: `php settle-client.php`

```
PHP - settle-client.php

1  <?php
2  // settle-client.php - PHP 7.4+ (requires the cURL extension)
3
4  const SETTLE_URL = 'https://w7api.com/sports/settle/result';
5
6  $settleData = [
7      'appId' => 'b2923cd03389635cf76815cf92c91622',
8      'data' => [
9          ['id' => 1, 'bid' => 1, 'mid' => 1, 'name' => 'Result', 'value' => 'Barcelona'],
10         ['id' => 2, 'bid' => 2, 'mid' => 7, 'name' => 'Handicap', 'value' => 'Freiburg @ 1'],
11         ['id' => 3, 'bid' => 19, 'mid' => 180, 'name' => 'Both Teams To Score', 'value' => 'Yes'],
12         ['id' => 4, 'bid' => 17, 'mid' => 9, 'name' => 'Total', 'value' => 'Over 0.5'],
13     ],
14 ];
15
16 /** POST the settle data. $gameId = "id", $leagueId = "lid". */
17 function settleBet(string $apiKey, string $gameId, int $leagueId, array $body): array
18 {
19     $url = SETTLE_URL . '?' . http_build_query(['apikey' => $apiKey, 'id' => $gameId, 'lid' => $leagueId]);
20
21     $ch = curl_init($url);
22     curl_setopt_array($ch, [
23         CURLOPT_POST => true,
24         CURLOPT_POSTFIELDS => json_encode($body),
25         CURLOPT_HTTPHEADER => ['Content-Type: application/json'],
26         CURLOPT_RETURNTRANSFER => true,
27         CURLOPT_TIMEOUT => 15,
28     ]);
29     $response = curl_exec($ch);
30     $code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
31     $error = curl_error($ch);
32     curl_close($ch);
33
34     if ($response === false) throw new RuntimeException("cURL error: $error");
35     if ($code !== 200) throw new RuntimeException("HTTP $code");
36     return json_decode($response, true); // ['success' => ..., 'id' => ..., 'data' => [...]]
37 }
38
39 // ----- usage -----
40 print_r(settleBet(getenv('SPORTSB00K_API_KEY') ?: '0', '6a49516e5e99bd05c6485682', 127733, $settleData));
```

CODE EXAMPLE

Kotlin

 Kotlin with the OkHttp library (Gradle dependency shown in the file).**Run:** Run main() from your IDE or Gradle.

```

KOTLIN - SettleClient.kt

1  // SettleClient.kt - Kotlin + OkHttp 4.x
2  // Gradle: implementation("com.squareup.okhttp3:okhttp:4.12.0")
3  // Android: call settle() from a background thread / coroutine (Dispatchers.IO).
4  import okhttp3.HttpUrl.Companion.toHttpUrl
5  import okhttp3.MediaType.Companion.toMediaType
6  import okhttp3.OkHttpClient
7  import okhttp3.Request
8  import okhttp3.RequestBody.Companion.toRequestBody
9  import java.io.IOException
10
11 class SettleClient(private val apiKey: String = "0") {      // "0" = no personal key
12
13     private val http = OkHttpClient()
14     private val settleUrl = "https://w7api.com/sports/settle/result"
15
16     /** POSTs the settle data (JSON text) and returns the JSON response as text. */
17     fun settle(gameId: String, leagueId: Long, jsonBody: String): String {
18         val url = settleUrl.toHttpUrl().newBuilder()
19             .addQueryParameter("apikey", apiKey)
20             .addQueryParameter("id", gameId)
21             .addQueryParameter("lid", leagueId.toString())
22             .build()
23         val request = Request.Builder().url(url)
24             .post(jsonBody.toRequestBody("application/json".toMediaType()))
25             .build()
26
27         http.newCall(request).execute().use { res ->
28             if (!res.isSuccessful) throw IOException("HTTP ${res.code}")
29             return res.body!!.string()
30         }
31     }
32 }
33
34 // ----- usage -----
35 fun main() {
36     val settleJson = """
37     {
38         "appId": "b2923cd03389635cf76815cf92c91622",
39         "data": [
40             { "id": 1, "bid": 1, "mid": 1, "name": "Result", "value": "Barcelona" },
41             { "id": 2, "bid": 2, "mid": 7, "name": "Handicap", "value": "Freiburg @ 1" },
42             { "id": 3, "bid": 19, "mid": 180, "name": "Both Teams To Score", "value": "Yes" },
43             { "id": 4, "bid": 17, "mid": 9, "name": "Total", "value": "Over 0.5" }
44         ]
45     }
46     """.trimIndent()
47
48     val api = SettleClient(System.getenv("SPORTSB00K_API_KEY") ?: "0")
49     println(api.settle("6a49516e5e99bd05c6485682", 127733, settleJson))
50 }

```

CODE EXAMPLE

Java

Java 17 or newer (text blocks). Uses the built-in `java.net.http` client.

Run: `javac SettleClient.java && java SettleClient`

```

1  // SettleClient.java - Java 17+ (text blocks, no external dependencies)
2  import java.io.IOException;
3  import java.net.URI;
4  import java.net.URLEncoder;
5  import java.net.http.HttpClient;
6  import java.net.http.HttpRequest;
7  import java.net.http.HttpResponse;
8  import java.nio.charset.StandardCharsets;
9
10 public class SettleClient {
11     private static final String SETTLE_URL = "https://w7api.com/sports/settle/result";
12     private final HttpClient http = HttpClient.newHttpClient();
13     private final String apiKey;
14
15     public SettleClient(String apiKey) { this.apiKey = apiKey; } // "0" = no personal key
16
17     private static String enc(String s) { return URLEncoder.encode(s, StandardCharsets.UTF_8); }
18
19     /** POSTs the settle data (JSON text) and returns the JSON response as text. */
20     public String settle(String gameId, long leagueId, String jsonBody)
21         throws IOException, InterruptedException {
22         String query = "apikey=" + enc(apiKey) + "&id=" + enc(gameId) + "&lid=" + leagueId;
23         HttpRequest req = HttpRequest.newBuilder(URI.create(SETTLE_URL + "?" + query))
24             .header("Content-Type", "application/json")
25             .POST(HttpRequest.BodyPublishers.ofString(jsonBody))
26             .build();
27         HttpResponse<String> res = http.send(req, HttpResponse.BodyHandlers.ofString());
28         if (res.statusCode() != 200) throw new IOException("HTTP " + res.statusCode());
29         return res.body();
30     }
31
32     // ----- usage -----
33     public static void main(String[] args) throws Exception {
34         String settleJson = ""
35         {
36             "appId": "b2923cd03389635cf76815cf92c91622",
37             "data": [
38                 { "id": 1, "bid": 1, "mid": 1, "name": "Result", "value": "Barcelona" },
39                 { "id": 2, "bid": 2, "mid": 7, "name": "Handicap", "value": "Freiburg @ 1" },
40                 { "id": 3, "bid": 19, "mid": 180, "name": "Both Teams To Score", "value": "Yes" },
41                 { "id": 4, "bid": 17, "mid": 9, "name": "Total", "value": "Over 0.5" }
42             ]
43         }
44         """;
45         String key = System.getenv().getOrDefault("SPORTSB00K_API_KEY", "0");
46         System.out.println(new SettleClient(key).settle("6a49516e5e99bd05c6485682", 127733, settleJson));
47     }
48 }

```

GUIDANCE

Best Practices & Notes

- 1 Keep your API key private**
Send settlement requests from your server and load the key from an environment variable.
- 2 Send the right IDs**
Use the game ID in `id` and the league ID in `lid`. A wrong ID settles the wrong game.
- 3 Send valid JSON**
Set `Content-Type: application/json` and check that the body is valid JSON before sending it.
- 4 Avoid duplicate settlements**
If a request fails or times out, check whether it was processed before you send it again.
- 5 Check the status and success**
Treat anything other than `200` as a failed request, and check that `success` is true in the response. Handle all four `result` values: Pending, Lost, Won and Refund.
- 6 Keep a log**
Store each request body and response, including the `result` of every item, so you can check a settlement later.

HTTP status handling

Status	Meaning	What to do
200	The request succeeded.	Read the response body.
4xx	The request was rejected, for example a missing parameter, a wrong key or invalid JSON.	Check the parameters and the JSON body.
5xx	A temporary problem on the server side.	Wait a moment, then check and retry.

Ready to settle?

Send a POST request to the settlement URL with your `apikey`, game ID and league ID. For more information visit bet365soft.com.