

DEVELOPER DOCUMENTATION

Sportsbook History API

Historical games, teams, players, tournaments, stages, leagues and events through one simple, JSON-based HTTP API.

VERSION 1

REST + JSON

8 ENDPOINTS



```
GET https://w7api.com/sports/history/v1/summary
    ?apikey=0
    &id={gameId}

200 OK application/json
```

API VERSION

v1

BASE URL

<https://w7api.com/sports/history/v1>

WEBSITE

bet365soft.com

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

WELCOME

Overview

The **Sportsbook History API** gives your application programmatic access to historical sports data: game summaries, teams, players, tournaments, stages, and past leagues and events by date range. Every endpoint is a simple HTTP **GET** request that returns **JSON**, so it works from any language or platform. This guide documents all 8 endpoints and includes ready-to-use client code for **Node.js**, **PHP**, **Kotlin** and **Java**.

API VERSION
v1

RESPONSE FORMAT
JSON

ENDPOINTS
8

AUTH
API key

Contents

Overview

Getting Started

Endpoint Overview

Endpoints

/summary - Game summary

/team - Team details

/player - Player details

/tournament - Tournament by season

/stage - Stage details

/getLeagues - Leagues by date range

/getEvents - Events by league

/getGame - Game details

Code Examples

Node.js example

PHP example

Kotlin example

Java example

Best Practices & Notes

2

3

4

5

5

5

6

6

7

7

8

8

9

10

11

12

13

14

BASICS

Getting Started

Base URL

All requests are made to the following base URL. Append the endpoint path (for example `/sports`) and your query parameters.

```
BASE URL

https://w7api.com/sports/history/v1
```

Authentication

Every endpoint requires the `apikey` query parameter. If you do not have your own API key, pass `0` as the value.

Quick start

No registration is needed to try the API. Use `apikey=0`, copy the request below into a terminal, and you will receive the leagues that had events in that date range.

```
TERMINAL - cURL

curl -G "https://w7api.com/sports/history/v1/getLeagues" \
  -d "apikey=0" -d "sid=1" \
  -d "fromDate=2026-10-16" -d "toDate=2026-10-18"
```

Request format

All endpoints use the **GET** method. Parameters are sent in the URL query string. Values with spaces or special characters must be URL-encoded (for example `Premier%20league`).

Parameter types

Type	Meaning
text	A string value, such as an API key, an ID or a date.
number	A numeric value, such as a sport ID, league ID or event ID.

Required and optional parameters

Each parameter in this guide is tagged with a **Required** or **Optional** badge. In this API every parameter is required.

Typical workflows

Most integrations chain a few endpoints together. These are the common paths:

Browse past matches	<code>/getLeagues</code>	>	<code>/getEvents</code>	>	<code>/getGame</code>
Tournament details	<code>/tournament</code>	>	<code>/stage</code>		

REFERENCE

Endpoint Overview

A summary of all 8 endpoints. Click an endpoint in the Contents page for full details.

#	Endpoint	Description	Required parameters
1	/summary	Game summary	apikey, id
2	/team	Team details	apikey, id
3	/player	Player details	apikey, id
4	/tournament	Tournament by season	apikey, id, seasonId
5	/stage	Stage details	apikey, id
6	/getLeagues	Leagues by date range	apikey, sid, fromDate, toDate
7	/getEvents	Events by league	apikey, id, fromDate, toDate
8	/getGame	Game details	apikey, id, lid, fromDate, toDate

Tip: finding IDs

League IDs come from [/getLeagues](#) and event IDs come from [/getEvents](#). Game, team, player, tournament, season and stage IDs are passed as text.

Date ranges

The [/getLeagues](#), [/getEvents](#) and [/getGame](#) endpoints use [fromDate](#) and [toDate](#) to select a past period, for example 2026-10-16 to 2026-10-18.

REFERENCE

Endpoints

Each endpoint below lists its parameters and a ready-to-copy example request.

01

GET

/summary

Game summary

Returns the summary of a game.

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	-	Game ID.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/history/v1/summary
?apikey=0
&id={gameId}
```

02

GET

/team

Team details

Returns historical data for a team.

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	-	Team ID.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/history/v1/team
?apikey=0
&id={teamId}
```

03

GET

/player

Player details

Returns historical data for a player.

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	-	Player ID.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/history/v1/player
?apikey=0
&id={playerId}
```

04

GET

/tournament

Tournament by season

Returns tournament data for a specific season.

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	-	Tournament ID.
seasonId	text	Required	-	Season ID.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/history/v1/tournament
?apikey=0
&id={tournamentId}
&seasonId={seasonId}
```

05

GET

/stage

Stage details

Returns data for a tournament stage.

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	-	Stage ID.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/history/v1/stage
?apikey=0
&id={stageId}
```

06

GET

/getLeagues

Leagues by date range

Returns the leagues for a sport within a past date range. League IDs are used in [/getEvents](#) and [/getGame](#) .

Parameter	Type	Required	Default	Description
apikey	text	Required	0	Your API key. If you don't have your own key, enter 0.
sid	number	Required	-	Sport ID.
fromDate	text	Required	-	Start of the date range, e.g. 2026-10-16.
toDate	text	Required	today	End of the date range, e.g. 2026-10-18.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/history/v1/getLeagues
?apikey=0
&sid=1
&fromDate=2026-10-16
&toDate=2026-10-18
```

07

GET

/getEvents

Events by league

Returns the events of a league within a past date range. Event IDs are used in `/getGame`.

Parameter	Type	Required	Default	Description
apikey	text	Required	0	Your API key. If you don't have your own key, enter 0.
id	number	Required	-	League ID, for example one returned by <code>/getLeagues</code> .
fromDate	text	Required	-	Start of the date range, e.g. 2026-10-16.
toDate	text	Required	today	End of the date range, e.g. 2026-10-18.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/history/v1/getEvents
?apikey=0
&id={leagueId}
&fromDate=2026-10-16
&toDate=2026-10-18
```

08

GET

/getGame

Game details

Returns the details of a single past event (game) in a league within a date range.

Parameter	Type	Required	Default	Description
apikey	text	Required	0	Your API key. If you don't have your own key, enter 0.
id	number	Required	-	Event ID, for example one returned by <code>/getEvents</code> .
lid	number	Required	-	League ID.
fromDate	text	Required	-	Start of the date range, e.g. 2026-10-16.
toDate	text	Required	today	End of the date range, e.g. 2026-10-18.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/history/v1/getGame
?apikey=0
&id={eventId}
&lid={leagueId}
&fromDate=2026-10-16
&toDate=2026-10-18
```

INTEGRATION

Code Examples

Each example below is a complete, self-contained client with one method per endpoint, plus a short usage section at the bottom. Copy it into your project, set your API key (or keep `0`) and run.

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 11 or newer. Uses the built-in `java.net.http` client.

Keep your API key private

If you use your own API key, do not hardcode it in a public repository or ship it inside a mobile app. Load it from an environment variable or call the API from your backend.

CODE EXAMPLE

Node.js

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

NODE.JS - SportsbookHistoryClient

```
1  ///  
2  const BASE_URL = "https://w7api.com/sports/history/v1";  
3  
4  class SportsbookHistoryClient {  
5      constructor(apiKey = "0") {          // "0" = no personal key  
6          this.apiKey = apiKey;  
7      }  
8  
9      async #get(endpoint, params = {}) {  
10         const query = new URLSearchParams({ apiKey: this.apiKey, ...params });  
11         const res = await fetch(`${BASE_URL}/${endpoint}?${query}`);  
12         if (!res.ok) throw new Error(`HTTP ${res.status} on /${endpoint}`);  
13         return res.json();  
14     }  
15  
16     summary(id)                { return this.#get("summary", { id }); }  
17     team(id)                   { return this.#get("team", { id }); }  
18     player(id)                 { return this.#get("player", { id }); }  
19     tournament(id, seasonId)   { return this.#get("tournament", { id, seasonId }); }  
20     stage(id)                  { return this.#get("stage", { id }); }  
21     getLeagues(sid, fromDate, toDate) { return this.#get("getLeagues", { sid, fromDate, toDate }); }  
22     getEvents(id, fromDate, toDate) { return this.#get("getEvents", { id, fromDate, toDate }); }  
23     getGame(id, lid, fromDate, toDate) {  
24         return this.#get("getGame", { id, lid, fromDate, toDate });  
25     }  
26 }  
27  
28 // ----- usage -----  
29 (async () => {  
30     const api = new SportsbookHistoryClient("0");  
31     console.log(await api.getLeagues(1, "2026-10-16", "2026-10-18"));  
32     console.log(await api.getEvents(12345, "2026-10-16", "2026-10-18")); // 12345 = a league ID  
33     console.log(await api.summary("GAME_ID")); // replace with a real game ID  
34 })().catch(console.error);
```

CODE EXAMPLE

PHP

 PHP 7.4 or newer with the cURL extension enabled.Run: `php SportsbookHistoryClient.php`

PHP - SportsbookHistoryClient

```

1  "<?php
2  // SportsbookHistoryClient.php - PHP 7.4+ (requires the cURL extension)
3
4  class SportsbookHistoryClient
5  {
6      private const BASE_URL = 'https://w7api.com/sports/history/v1';
7      private string $apiKey;
8
9      public function __construct(string $apiKey = '0')    // '0' = no personal key
10     {
11         $this->apiKey = $apiKey;
12     }
13
14     private function get(string $endpoint, array $params = []): array
15     {
16         $params = ['apikey' => $this->apiKey] + $params;
17         $params = array_filter($params, fn($v) => $v !== null && $v !== '');
18         $url = self::BASE_URL . '/' . $endpoint . '?' . http_build_query($params);
19
20         $ch = curl_init($url);
21         curl_setopt_array($ch, [CURLOPT_RETURNTRANSFER => true, CURLOPT_TIMEOUT => 15]);
22         $body = curl_exec($ch);
23         $code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
24         $err = curl_error($ch);
25         curl_close($ch);
26
27         if ($body === false) throw new RuntimeException("cURL error: $err");
28         if ($code !== 200) throw new RuntimeException("HTTP $code on /$endpoint");
29         return json_decode($body, true, 512, JSON_THROW_ON_ERROR);
30     }
31
32     public function summary(string $id): array { return $this->get('summary', compact('id')); }
33     public function team(string $id): array { return $this->get('team', compact('id')); }
34     public function player(string $id): array { return $this->get('player', compact('id')); }
35     public function tournament(string $id, string $seasonId): array
36     { return $this->get('tournament', compact('id', 'seasonId')); }
37     public function stage(string $id): array { return $this->get('stage', compact('id')); }
38     public function getLeagues(int $sid, string $fromDate, string $toDate): array
39     { return $this->get('getLeagues', compact('sid', 'fromDate', 'toDate')); }
40     public function getEvents(int $id, string $fromDate, string $toDate): array
41     { return $this->get('getEvents', compact('id', 'fromDate', 'toDate')); }
42     public function getGame(int $id, int $lid, string $fromDate, string $toDate): array
43     { return $this->get('getGame', compact('id', 'lid', 'fromDate', 'toDate')); }
44 }
45
46 // ----- usage -----
47 $api = new SportsbookHistoryClient('0');
48 print_r($api->getLeagues(1, '2026-10-16', '2026-10-18'));
49 print_r($api->getEvents(12345, '2026-10-16', '2026-10-18'));    // 12345 = a league ID
50 print_r($api->summary('GAME_ID'));                               // replace with a real game ID

```

CODE EXAMPLE

Kotlin

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

Run: Run main() from your IDE or Gradle.

```
KOTLIN - SportsbookHistoryClient

1  """ SportsbookHistoryClient.kt - Kotlin + OkHttp 4.x
2  // Gradle: implementation("com.squareup.okhttp3:okhttp:4.12.0")
3  // Android: call these methods from a background thread / coroutine (Dispatchers.IO).
4  import okhttp3.HttpUrl.Companion.toHttpUrl
5  import okhttp3.OkHttpClient
6  import okhttp3.Request
7  import java.io.IOException
8
9  class SportsbookHistoryClient(private val apiKey: String = "0") {    // "0" = no personal key
10
11     private val http = OkHttpClient()
12     private val baseUrl = "https://w7api.com/sports/history/v1"
13
14     /** Returns the raw JSON string. Parse it with Gson, Moshi or kotlinx.serialization. */
15     private fun get(endpoint: String, params: Map<String, Any?> = emptyMap()): String {
16         val url = "$baseUrl/$endpoint".toHttpUrl().newBuilder().apply {
17             addQueryParameter("apikey", apiKey)
18             params.forEach { (k, v) -> if (v != null) addQueryParameter(k, v.toString()) }
19         }.build()
20
21         http.newCall(Request.Builder().url(url).build()).execute().use { res ->
22             if (!res.isSuccessful) throw IOException("HTTP ${res.code} on /$endpoint")
23             return res.body!!.string()
24         }
25     }
26
27     fun summary(id: String) = get("summary", mapOf("id" to id))
28     fun team(id: String) = get("team", mapOf("id" to id))
29     fun player(id: String) = get("player", mapOf("id" to id))
30     fun tournament(id: String, seasonId: String) =
31         get("tournament", mapOf("id" to id, "seasonId" to seasonId))
32     fun stage(id: String) = get("stage", mapOf("id" to id))
33     fun getLeagues(sid: Int, fromDate: String, toDate: String) =
34         get("getLeagues", mapOf("sid" to sid, "fromDate" to fromDate, "toDate" to toDate))
35     fun getEvents(id: Long, fromDate: String, toDate: String) =
36         get("getEvents", mapOf("id" to id, "fromDate" to fromDate, "toDate" to toDate))
37     fun getGame(id: Long, lid: Long, fromDate: String, toDate: String) =
38         get("getGame", mapOf("id" to id, "lid" to lid, "fromDate" to fromDate, "toDate" to toDate))
39 }
40
41 // ----- usage -----
42 fun main() {
43     val api = SportsbookHistoryClient()
44     println(api.getLeagues(1, "2026-10-16", "2026-10-18"))
45     println(api.getEvents(12345, "2026-10-16", "2026-10-18"))    // 12345 = a league ID
46     println(api.summary("GAME_ID"))                             // replace with a real game ID
47 }
```

CODE EXAMPLE

Java

 Java 11 or newer. Uses the built-in java.net.http client.**Run:** `javac SportsbookHistoryClient.java && java SportsbookHistoryClient`**JAVA - SportsbookHistoryClient**

```

1  ///  
2  import java.io.IOException;  
3  import java.net.URI;  
4  import java.net.URLEncoder;  
5  import java.net.http.*;  
6  import java.nio.charset.StandardCharsets;  
7  import java.time.Duration;  
8  import java.util.*;  
9  public class SportsbookHistoryClient {  
10     private static final String BASE_URL = "https://w7api.com/sports/history/v1";  
11     private final HttpClient http = HttpClient.newBuilder()  
12         .connectTimeout(Duration.ofSeconds(10)).build();  
13     private final String apiKey;  
14     public SportsbookHistoryClient() { this(""); }           // "" = no personal key  
15     public SportsbookHistoryClient(String apiKey) { this.apiKey = apiKey; }  
16  
17     private static String enc(String s) { return URLEncoder.encode(s, StandardCharsets.UTF_8); }  
18     /** Returns the raw JSON string. Parse it with Jackson, Gson or org.json. */  
19     private String get(String endpoint, Object... kv) {       // kv = name, value, name, value ...  
20         StringJoiner query = new StringJoiner("&");  
21         query.add("apiKey=" + enc(apiKey));  
22         for (int i = 0; i < kv.length; i += 2)  
23             query.add(kv[i] + "=" + enc(String.valueOf(kv[i + 1])));  
24         try {  
25             HttpRequest req = HttpRequest  
26                 .newBuilder(URI.create(BASE_URL + "/" + endpoint + "?" + query))  
27                 .timeout(Duration.ofSeconds(15)).GET().build();  
28             HttpResponse<String> res = http.send(req, HttpResponse.BodyHandlers.ofString());  
29             if (res.statusCode() != 200) throw new IOException("HTTP " + res.statusCode());  
30             return res.body();  
31         } catch (IOException | InterruptedException e) {  
32             throw new RuntimeException("Request failed: /" + endpoint, e);  
33         }  
34     }  
35     public String summary(String id) { return get("summary", "id", id); }  
36     public String team(String id) { return get("team", "id", id); }  
37     public String player(String id) { return get("player", "id", id); }  
38     public String tournament(String id, String seasonId) {  
39         return get("tournament", "id", id, "seasonId", seasonId);  
40     }  
41     public String stage(String id) { return get("stage", "id", id); }  
42     public String getLeagues(int sid, String from, String to) {  
43         return get("getLeagues", "sid", sid, "fromDate", from, "toDate", to);  
44     }  
45     public String getEvents(long id, String from, String to) {  
46         return get("getEvents", "id", id, "fromDate", from, "toDate", to);  
47     }  
48     public String getGame(long id, long lid, String from, String to) {  
49         return get("getGame", "id", id, "lid", lid, "fromDate", from, "toDate", to);  
50     }  
51  
52     public static void main(String[] args) {           // usage  
53         SportsbookHistoryClient api = new SportsbookHistoryClient();  
54         System.out.println(api.getLeagues(1, "2026-10-16", "2026-10-18"));  
55         System.out.println(api.getEvents(12345, "2026-10-16", "2026-10-18")); // league ID  
56         System.out.println(api.summary("GAME_ID"));   // use a real game ID  
57     }  
58 }

```

GUIDANCE

Best Practices & Notes

- 1 Cache historical data**
Past games, teams and players rarely change. Cache responses from `/summary` , `/team` and `/player` instead of requesting them again.
- 2 Keep date ranges small**
Request only the period you need with `fromDate` and `toDate` on `/getLeagues` , `/getEvents` and `/getGame` .
- 3 Always URL-encode values**
Use your language's query builder (as in the examples) so every value is encoded correctly.
- 4 Handle errors and timeouts**
Check the HTTP status code, set a request timeout and show a friendly message (or retry once) when a request fails.
- 5 Treat text IDs as strings**
The `id` parameter is text on `/summary` , `/team` , `/player` , `/tournament` and `/stage` . Keep it as a string in your code.
- 6 Send every required parameter**
`/tournament` needs both `id` and `seasonId` . `/getGame` needs `id` , `lid` , `fromDate` and `toDate` .

HTTP status handling

Status	Meaning	What to do
200	Request succeeded.	Parse the JSON body.
4xx	The request is invalid, for example a missing or wrong parameter.	Check the parameters and the endpoint path.
5xx	A temporary problem on the server side.	Wait a moment and retry.

Ready to build?

Start with `/getLeagues` using `apikey=0` . For more information visit [bet365soft.com](https://w7api.com/sports/history/v1).