

DEVELOPER DOCUMENTATION

Sportsbook SSE API

Sports, events, leagues, history and bet slips delivered over Server-Sent Events (SSE) with simple header-based authentication.

VERSION 1

SSE STREAM

20 ENDPOINTS

```
GET https://w7api.com/sports/sse/v1/inplay
?id=1
&country=NG

200 OK text/event-stream
```

API VERSION

v1

BASE URL

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

WEBSITE

bet365soft.com

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

WELCOME

Overview

The **Sportsbook SSE API** delivers sports data over **Server-Sent Events (SSE)**: one long-lived HTTP connection, authenticated with the **x-api-key** header. It covers sports, live and pre-match events, leagues, tournaments, history and bet slips in 20 endpoints, with client code for the **Browser, Node.js, PHP, Kotlin and Java**.

API VERSION

v1

PROTOCOL

SSE

ENDPOINTS

20

AUTH HEADER

x-api-key

Contents

Overview	2
Getting Started	3
Origin Connection	4
Endpoint Overview	5
Endpoints	6
/sports - List sports	6
/inplay - Live events by sport	6
/inplayEvent - Live event details	7
/leagues - Leagues by sport	7
/prematch - Pre-match events	8
/prematchEvent - Pre-match event details	8
/upcoming - Upcoming events by date	9
/topEvents - Top events	9
/search - Search events	10
/tournaments - List tournaments	10
/tournamentWinner - Tournament winner	11
/summary - Game summary	11
/team - Team details	11
/player - Player details	12
/stage - Stage details	12
/getLeagues - Leagues by date range	13
/getEvents - Events by league	13
/getGame - Game details	14
/betSlip - Bet slip	14
/betValidation - Bet validation	15
Bet Slip Data	16
Code Examples	17
Best Practices & Notes	23

BASICS

Getting Started

Base URL

All requests are made to the base URL below. Append the endpoint path, for example `/sports`.

```
BASE URL

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

How SSE works

1 Connect

Send a GET request with the required headers.

2 Stay open

The server keeps the HTTP connection open.

3 Receive events

Messages arrive as data lines, each ended by a blank line.

```
STANDARD SSE MESSAGE FORMAT

: a comment line (optional, ignore it)
data: first line of the message
data: another line (optional)
(blank line = the message is complete)
```

Authentication

Send your API key in the `x-api-key` HTTP header on every request, together with the `Accept` header. The key is not sent as a query parameter.

Header	Value	Required	Description
<code>x-api-key</code>	YOUR_API_KEY	Required	Your API key.
<code>Accept</code>	text/event-stream	Required	Tells the server you want an SSE stream.
<code>Origin</code>	https://yourdomain.com	Browsers	Added automatically by the browser. It must be an allowed origin.
<code>Content-Type</code>	application/json	POST only	For the JSON body of <code>/betValidation</code> .

Quick start

```
TERMINAL - cURL (-N keeps the stream unbuffered)

curl -N "https://w7api.com/sports/sse/v1/inplay?id=1&country=NG" \
-H "x-api-key: YOUR_API_KEY" \
-H "Accept: text/event-stream"
```

BROWSER ACCESS

Origin Connection

For browser-based SSE connections, the requesting **Origin** must be allowed by the API or server configuration. The browser sends the `Origin` header automatically on cross-origin requests, and the server answers with a matching `Access-Control-Allow-Origin` value for an allowed origin.

REQUEST HEADERS SENT BY A BROWSER

```
Origin: https://yourdomain.com
x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

Do not use a wildcard origin

For credential-sensitive browser configurations, do not allow every origin with `*`. Use the exact allowed origin, for example `https://yourdomain.com`, whenever the server is configured to restrict browser access.

EventSource or fetch()

The browser's built-in `EventSource` is the simplest option, but it cannot send custom headers such as `x-api-key`. Use `fetch()` with a readable stream when you must send that header (see the Browser example). Custom headers also trigger a CORS preflight request, so the server must allow them.

Server-side clients

Node.js, PHP, Kotlin and Java clients do not send an `Origin` header automatically. They only need the `x-api-key` and `Accept` headers.

Request format

All endpoints use **GET** with parameters in the URL query string, except `/betValidation`, which uses **POST** with a JSON body. Values with spaces or special characters must be URL-encoded (for example `Premier%20league`).

Type	Meaning
<code>text</code>	A string value, such as an ID, a date, a country code or a search keyword.
<code>number</code>	A numeric value, such as a sport ID, league ID or event ID.
<code>json</code>	A JSON value. In query strings it must be URL-encoded.

Typical workflows

Browse live events	/sports	›	/inplay	›	/inplayEvent		
Browse pre-match events	/sports	›	/leagues	›	/prematch	›	/prematchEvent
Look at past games	/getLeagues	›	/getEvents	›	/getGame		

REFERENCE

Endpoint Overview

A summary of all 20 endpoints. Every request also needs the headers described in Authentication.

#	Method	Endpoint	Description	Required parameters
1	GET	/sports	List sports	none
2	GET	/inplay	Live events by sport	id
3	GET	/inplayEvent	Live event details	id
4	GET	/leagues	Leagues by sport	id
5	GET	/prematch	Pre-match events	id, sid
6	GET	/prematchEvent	Pre-match event details	id
7	GET	/upcoming	Upcoming events by date	id, fromDate, toDate
8	GET	/topEvents	Top events	status
9	GET	/search	Search events	status, text
10	GET	/tournaments	List tournaments	none
11	GET	/tournamentWinner	Tournament winner	id
12	GET	/summary	Game summary	id
13	GET	/team	Team details	id
14	GET	/player	Player details	id
15	GET	/stage	Stage details	id
16	GET	/getLeagues	Leagues by date range	sid, fromDate, toDate
17	GET	/getEvents	Events by league	id, fromDate, toDate
18	GET	/getGame	Game details	id, lid, fromDate, toDate
19	GET	/betSlip	Bet slip	id, data
20	POST	/betValidation	Bet validation	id, data

Tip: finding IDs

Sport IDs come from [/sports](#) , league IDs from [/leagues](#) or [/getLeagues](#) , and event IDs from list endpoints such as [/inplay](#) , [/prematch](#) and [/getEvents](#) .

REFERENCE

Endpoints

Each endpoint lists its parameters and a ready-to-copy request. All requests need the `x-api-key` and `Accept` headers.

01

GET

/sports

List sports

Returns every sport supported by the API. Use the sport IDs from this response in other endpoints.

No parameters. This endpoint only needs the request headers.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/sports
```

```
x-api-key: YOUR_API_KEY
```

```
Accept: text/event-stream
```

02

GET

/inplay

Live events by sport

Returns the live (in-play) events for a sport, with an optional country filter.

Parameter	Type	Required	Default	Description
id	number	Required	1	Sport ID. Get valid IDs from /sports .
country	text	Optional	NG	Country code, for example NG. Optional.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/inplay
```

```
?id=1
```

```
&country=NG
```

```
x-api-key: YOUR_API_KEY
```

```
Accept: text/event-stream
```

03

GET

/inplayEvent

Live event details

Returns full details of a single live event.

Parameter	Type	Required	Default	Description
id	number	Required	-	Event ID, for example one returned by <code>/inplay</code> .

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/inplayEvent
?id={eventId}

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

04

GET

/leagues

Leagues by sport

Returns the leagues available for a sport, with an optional country filter.

Parameter	Type	Required	Default	Description
id	number	Required	1	Sport ID. Get valid IDs from <code>/sports</code> .
country	text	Optional	NG	Country code, for example NG. Optional.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/leagues
?id=1
&country=NG

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

05

GET

/prematch

Pre-match events

Returns the pre-match events of a league within a sport.

Parameter	Type	Required	Default	Description
id	number	Required	-	League ID. Get valid IDs from /leagues .
sid	number	Required	1	Sport ID. Get valid IDs from /sports .
country	text	Optional	NG	Country code, for example NG. Optional.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/prematch
?id={leagueId}
&sid=1
&country=NG

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

06

GET

/prematchEvent

Pre-match event details

Returns full details of a single pre-match event.

Parameter	Type	Required	Default	Description
id	number	Required	-	Event ID, for example one returned by /prematch .

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/prematchEvent
?id={eventId}

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

07

GET

/upcoming

Upcoming events by date

Returns upcoming events for a sport within a date range, with an optional country filter.

Parameter	Type	Required	Default	Description
id	number	Required	1	Sport ID. Get valid IDs from /sports .
fromDate	text	Required	today	Start of the date range, e.g. 2026-10-18.
toDate	text	Required	tomorrow	End of the date range, e.g. 2026-10-19.
country	text	Optional	NG	Country code, for example NG. Optional.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/upcoming
?id=1
&fromDate=2026-10-18
&toDate=2026-10-19
&country=NG

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

08

GET

/topEvents

Top events

Returns the featured top events, either live or pre-match.

Parameter	Type	Required	Default	Description
status	text	Required	Line	Event status. Allowed values: Live or Line .
country	text	Optional	NG	Country code, for example NG. Optional.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/topEvents
?status=Line
&country=NG

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

09

GET

/search

Search events

Searches events by name, for example a league or a team.

Parameter	Type	Required	Default	Description
status	text	Required	Line	Event status. Allowed values: <code>Live</code> or <code>Line</code> .
text	text	Required	Premier league	Search keyword. URL-encode spaces (e.g. <code>Premier%20league</code>).
country	text	Optional	NG	Country code, for example NG. Optional.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/search
?status=Line
&text=Premier%20league
&country=NG

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

10

GET

/tournaments

List tournaments

Returns the list of tournaments available through the API.

No parameters. This endpoint only needs the request headers.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/tournaments

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

11

GET

/tournamentWinner

Tournament winner

Returns the tournament-winner (outright) data for a championship.

Parameter	Type	Required	Default	Description
id	number	Required	1	Championship (league) ID.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/tournamentWinner
?id=1

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

12

GET

/summary

Game summary

Returns the summary of a game.

Parameter	Type	Required	Default	Description
id	text	Required	-	Game ID.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/summary
?id={gameId}

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

13

GET

/team

Team details

Returns historical data for a team.

Parameter	Type	Required	Default	Description
id	text	Required	-	Team ID.



EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/team
?id={teamId}

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

14

GET

/player

Player details

Returns historical data for a player.

Parameter	Type	Required	Default	Description
id	text	Required	-	Player ID.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/player
?id={playerId}

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

15

GET

/stage

Stage details

Returns data for a tournament stage.

Parameter	Type	Required	Default	Description
id	text	Required	-	Stage ID.

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/stage
?id={stageId}

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

16

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
sid	number	Required	-	Sport ID. Get valid IDs from <code>/sports</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/sse/v1/getLeagues
?sid=1
&fromDate=2026-10-16
&toDate=2026-10-18

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

17

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
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/sse/v1/getEvents
?id={leagueId}
&fromDate=2026-10-16
&toDate=2026-10-18

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

18

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
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/sse/v1/getGame
?id={eventId}
&lid={leagueId}
&fromDate=2026-10-16
&toDate=2026-10-18

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

19

GET

/betSlip

Bet slip

Sends bet slip data with a GET request. Pass the slip JSON array URL-encoded in the data parameter (see Bet Slip Data).

Parameter	Type	Required	Default	Description
id	text	Required	-	Your own reference ID for the slip.
data	json	Required	-	Bet slip JSON array, URL-encoded with <code>encodeURIComponent</code> .

EXAMPLE REQUEST

```
GET https://w7api.com/sports/sse/v1/betSlip
?id={yourSlipId}
&data={urlEncodedSlipJson}

x-api-key: YOUR_API_KEY
Accept: text/event-stream
```

20

POST

/betValidation

Bet validation

Validates a bet slip. Send a JSON request body that contains your own ID and the slip data array (see Bet Slip Data).

Body field	Type	Required	Default	Description
id	any	Required	-	Your own server or database ID. Any value is accepted.
data	json array	Required	-	The bet slip JSON array.

EXAMPLE REQUEST

```
POST https://w7api.com/sports/sse/v1/betValidation
x-api-key: YOUR_API_KEY
Content-Type: application/json

{ "id": 1, "data": [ ...bet slip array... ] }
```

REFERENCE

Bet Slip Data

The `data` parameter of `/betSlip` and the `data` field of `/betValidation` use the same JSON array. Each item in the array is one selection on the bet slip.

Field	Type	Required	Description
eid	string	Required	Event ID.
gid	string	Required	Game ID. It is the same as <code>eid</code> in the examples.
vs	string	Required	Match name, for example Barnaul Vs Novosibirsk.
bid	string	Required	Bet (market) ID, for example 101.
bkey	string	Required	Bet (market) name, for example Result 2-Way.
mid	string	Required	Selection ID, for example 401.
mkey	string	Required	Selection name, for example Barnaul.
rate	number	Required	Odds value, for example 1.93.
status	string or number	Required	Visibility: <code>Show</code> or <code>0</code> to show, <code>Hide</code> or <code>1</code> to hide.
match	string	Required	Match type: <code>Line</code> or <code>Live</code> .
sport	string	Optional	Sport name, for example Basketball.
league	string	Optional	League name, for example IPBL Pro Division.

Additional optional fields can be added to each item as needed.

Example bet slip

```
JSON - bet slip with 2 selections

1  [
2    {
3      "eid": "757541132", "gid": "757541132",
4      "vs": "Barnaul Vs Novosibirsk",
5      "bid": "101", "bkey": "Result 2-Way",
6      "mid": "401", "mkey": "Barnaul",
7      "rate": 1.93, "status": "Show", "match": "Live",
8      "sport": "Basketball", "league": "IPBL Pro Division"
9    },
10   {
11     "eid": "757541956", "gid": "757541956",
12     "vs": "Severodvinsk Women Vs Orenburg Women",
13     "bid": "101", "bkey": "Result 2-Way",
14     "mid": "401", "mkey": "Severodvinsk Women",
15     "rate": 2.39, "status": "Show", "match": "Live",
16     "sport": "Basketball", "league": "IPBL Pro Division Women"
17   }
18 ]
```

Sending the slip with GET

For `/betSlip`, convert the array to a string with `JSON.stringify` and URL-encode it with `encodeURIComponent` before adding it to the query string. For `/betValidation`, send the array as normal JSON inside the request body.

INTEGRATION

Code Examples

Each example opens an SSE stream on `/inplay`, prints every message, and includes a `/betValidation` helper (except the browser example). Replace `YOUR_API_KEY` with your key, or set the `SPORTSB00K_API_KEY` environment variable for server-side code.

Browser Any modern browser. EventSource or fetch().	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

A key placed in browser JavaScript is visible to every visitor. For public websites, restrict access with the allowed-origin setting, or call the API from your own backend. Never commit keys to a public repository.

BROWSER SSE REQUEST EXAMPLE

Browser

 Any modern browser. EventSource or fetch().

Run: Load the script from your allowed origin and open the browser console.

Option 1: EventSource

```
BROWSER - sse-eventsourc.js

1 // sse-eventsourc.js - browser, built-in EventSource
2 const source = new EventSource(
3   "https://w7api.com/sports/sse/v1/inplay?id=1&country=NG"
4 );
5
6 source.onmessage = (event) => {
7   const data = JSON.parse(event.data);
8   console.log(data);
9 };
10
11 source.onerror = (error) => {
12   console.error("SSE connection error", error);
13 };
```

Option 2: fetch() with the x-api-key header

```
BROWSER - sse-browser.js

1 // sse-browser.js - runs in the browser (fetch + ReadableStream)
2 // Native EventSource cannot send custom headers such as x-api-key, so we use fetch().
3 const BASE_URL = "https://w7api.com/sports/sse/v1";
4 const API_KEY = "YOUR_API_KEY";
5
6 const tryJson = (t) => { try { return JSON.parse(t); } catch { return t; } };
7
8 async function streamSSE(endpoint, params, onEvent, signal) {
9   const url = `${BASE_URL}/${endpoint}?${new URLSearchParams(params)}`;
10   const res = await fetch(url, {
11     headers: { "x-api-key": API_KEY, "Accept": "text/event-stream" },
12     signal, // lets you stop the stream (AbortController)
13   });
14   if (!res.ok) throw new Error(`HTTP ${res.status} on /${endpoint}`);
15
16   const reader = res.body.pipeThrough(new TextDecoderStream()).getReader();
17   let buffer = "";
18   while (true) {
19     const { value, done } = await reader.read();
20     if (done) break;
21     buffer += value.replace(/\r\n/g, "\n");
22     const messages = buffer.split("\n\n");
23     buffer = messages.pop(); // keep the unfinished part
24     for (const msg of messages) {
25       const data = msg.split("\n")
26         .filter((line) => line.startsWith("data:"))
27         .map((line) => line.slice(5).trim())
28         .join("\n");
29       if (data) onEvent(tryJson(data));
30     }
31   }
32 }
33
34 // ----- usage -----
35 const controller = new AbortController();
36 streamSSE("inplay", { id: 1, country: "NG" },
37   (event) => console.log("event:", event), controller.signal)
38   .catch(console.error);
39
40 // controller.abort(); // call this to close the connection
```

CODE EXAMPLE

Node.js

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

```
1 // sse-client.js - Node.js 18+ (built-in fetch, no packages needed)
2 const BASE_URL = "https://w7api.com/sports/sse/v1";
3 const API_KEY = process.env.SPORTSB00K_API_KEY || "YOUR_API_KEY";
4 const HEADERS = { "x-api-key": API_KEY };
5
6 const tryJson = (t) => { try { return JSON.parse(t); } catch { return t; } };
7
8 /** Async generator: yields one parsed message per SSE event. */
9 async function* sse(endpoint, params = {}, signal) {
10   const url = `${BASE_URL}/${endpoint}?${new URLSearchParams(params)}`;
11   const res = await fetch(url, { headers: { ...HEADERS, Accept: "text/event-stream" }, signal });
12   if (!res.ok) throw new Error(`HTTP ${res.status} on /${endpoint}`);
13
14   const decoder = new TextDecoder();
15   let buffer = "";
16   for await (const chunk of res.body) {
17     buffer += decoder.decode(chunk, { stream: true }).replace(/\r\n/g, "\n");
18     let end;
19     while ((end = buffer.indexOf("\n\n")) !== -1) {
20       const raw = buffer.slice(0, end);
21       buffer = buffer.slice(end + 2);
22       const data = raw.split("\n").filter((l) => l.startsWith("data:"))
23         .map((l) => l.slice(5).trim()).join("\n");
24       if (data) yield tryJson(data);
25     }
26   }
27 }
28
29 /** POST /betValidation (id = your own ID, data = the bet slip array) */
30 async function betValidation(id, data) {
31   const res = await fetch(`${BASE_URL}/betValidation`, {
32     method: "POST",
33     headers: { ...HEADERS, "Content-Type": "application/json" },
34     body: JSON.stringify({ id, data }),
35   });
36   if (!res.ok) throw new Error(`HTTP ${res.status} on /betValidation`);
37   return res.text();
38 }
39
40 // ----- usage -----
41 (async () => {
42   const controller = new AbortController();
43   setTimeout(() => controller.abort(), 60_000); // stop after 60 seconds
44
45   for await (const event of sse("inplay", { id: 1, country: "NG" }, controller.signal)) {
46     console.log("event:", event);
47   }
48 })().catch((err) => { if (err.name !== "AbortError") console.error(err); });
```

CODE EXAMPLE

PHP

PHP 7.4 or newer with the cURL extension enabled.

Run: `php sse-client.php`

```

1  <?php
2  // sse-client.php - PHP 7.4+ (requires the cURL extension)
3
4  const BASE_URL = 'https://w7api.com/sports/sse/v1';
5
6  function apiKey(): string { return getenv('SPORTSB00K_API_KEY') ?: 'YOUR_API_KEY'; }
7
8  /** Opens an SSE stream and calls $onEvent for every message. Runs until the server closes it. */
9  function sseStream(string $endpoint, array $params, callable $onEvent): void
10 {
11     $url = BASE_URL . '/' . $endpoint . '?' . http_build_query($params);
12     $buffer = '';
13
14     $sch = curl_init($url);
15     curl_setopt_array($sch, [
16         CURLOPT_HTTPHEADER => ['x-api-key: ' . apiKey(), 'Accept: text/event-stream'],
17         CURLOPT_TIMEOUT     => 0, // 0 = keep the stream open
18         CURLOPT_WRITEFUNCTION => function ($sch, $chunk) use (&$buffer, $onEvent) {
19             $buffer .= str_replace("\r\n", "\n", $chunk);
20             while (($pos = strpos($buffer, "\n\n")) !== false) {
21                 $raw = substr($buffer, 0, $pos);
22                 $buffer = substr($buffer, $pos + 2);
23                 $lines = [];
24                 foreach (explode("\n", $raw) as $line) {
25                     if (strpos($line, 'data:') === 0) $lines[] = trim(substr($line, 5));
26                 }
27                 if ($lines) {
28                     $text = implode("\n", $lines);
29                     $onEvent(json_decode($text, true) ?? $text);
30                 }
31             }
32             return strlen($chunk); // cURL requires the byte count
33         },
34     ]);
35     curl_exec($sch);
36     $error = curl_error($sch);
37     $code = curl_getinfo($sch, CURLINFO_HTTP_CODE);
38     curl_close($sch);
39     if ($error) throw new RuntimeException($error);
40     if ($code >= 400) throw new RuntimeException("HTTP $code on /$endpoint");
41 }
42
43 /** POST /betValidation ($id = your own ID, $data = the bet slip array) */
44 function betValidation($id, array $data): string
45 {
46     $sch = curl_init(BASE_URL . '/betValidation');
47     curl_setopt_array($sch, [
48         CURLOPT_POST          => true,
49         CURLOPT_POSTFIELDS    => json_encode(['id' => $id, 'data' => $data]),
50         CURLOPT_HTTPHEADER    => ['x-api-key: ' . apiKey(), 'Content-Type: application/json'],
51         CURLOPT_RETURNTRANSFER => true,
52     ]);
53     $body = curl_exec($sch);
54     curl_close($sch);
55     return $body;
56 }
57
58 // ----- usage -----
59 sseStream('inplay', ['id' => 1, 'country' => 'NG'], function ($event) {
60     print_r($event);
61 });

```

CODE EXAMPLE

Kotlin

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

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

```

KOTLIN - SportsbookSseClient.kt

1  // SportsbookSseClient.kt - Kotlin + OkHttp 4.x
2  // Gradle: implementation("com.squareup.okhttp3:okhttp:4.12.0")
3  // Android: run stream() on 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 import java.util.concurrent.TimeUnit
11
12 class SportsbookSseClient(private val apiKey: String) {
13
14     private val baseUrl = "https://w7api.com/sports/sse/v1"
15     private val http = OkHttpClient.Builder()
16         .readTimeout(0, TimeUnit.MILLISECONDS) // 0 = no read timeout, keeps the stream open
17         .build()
18
19     /** Blocks the calling thread and calls onEvent for every SSE message. */
20     fun stream(endpoint: String, params: Map<String, Any> = emptyMap(), onEvent: (String) -> Unit) {
21         val url = "$baseUrl/$endpoint".toHttpUrl().newBuilder().apply {
22             params.forEach { (k, v) -> addQueryParameter(k, v.toString()) }
23         }.build()
24         val request = Request.Builder().url(url)
25             .header("x-api-key", apiKey)
26             .header("Accept", "text/event-stream")
27             .build()
28
29         http.newCall(request).execute().use { res ->
30             if (!res.isSuccessful) throw IOException("HTTP ${res.code} on /$endpoint")
31             val source = res.body!!.source()
32             val data = StringBuilder()
33             while (!source.exhausted()) {
34                 val line = source.readUtf8Line() ?: break
35                 when {
36                     line.startsWith("data:") ->
37                         data.append(line.removePrefix("data:").trim()).append('\n')
38                     line.isEmpty() && data.isNotEmpty() -> {
39                         onEvent(data.toString().trim()); data.clear()
40                     }
41                 }
42             }
43         }
44     }
45
46     /** POST /betValidation (id = your own ID, dataJson = the bet slip JSON array) */
47     fun betValidation(id: Long, dataJson: String): String {
48         val body = "{\"id\":$id,\"data\":$dataJson}".toRequestBody("application/json".toMediaType())
49         val request = Request.Builder().url("$baseUrl/betValidation")
50             .header("x-api-key", apiKey).post(body).build()
51         http.newCall(request).execute().use { return it.body!!.string() }
52     }
53 }
54
55 // ----- usage -----
56 fun main() {
57     val api = SportsbookSseClient(System.getenv("SPORTSB00K_API_KEY") ?: "YOUR_API_KEY")
58     api.stream("inplay", mapOf("id" to 1, "country" to "NG")) { event ->
59         println("event: $event")
60     }
61 }

```

CODE EXAMPLE

Java

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

```

1  // SportsbookSseClient.java - Java 11+ (no external dependencies)
2  import java.net.URI;
3  import java.net.URLEncoder;
4  import java.net.http.*;
5  import java.nio.charset.StandardCharsets;
6  import java.util.*;
7  import java.util.function.Consumer;
8  import java.util.stream.Stream;
9
10 public class SportsbookSseClient {
11     private static final String BASE_URL = "https://w7api.com/sports/sse/v1";
12     private final HttpClient http = HttpClient.newHttpClient();
13     private final String apiKey;
14
15     public SportsbookSseClient(String apiKey) { this.apiKey = apiKey; }
16
17     private static String enc(Object o) {
18         return URLEncoder.encode(String.valueOf(o), StandardCharsets.UTF_8);
19     }
20
21     /** Blocks the calling thread and calls onEvent for every SSE message. */
22     public void stream(String endpoint, Map<String, Object> params, Consumer<String> onEvent)
23         throws Exception {
24         StringJoiner query = new StringJoiner("&");
25         params.forEach((k, v) -> query.add(k + "=" + enc(v)));
26         HttpRequest req = HttpRequest
27             .newBuilder(URI.create(BASE_URL + "/" + endpoint + "?" + query))
28             .header("x-api-key", apiKey)
29             .header("Accept", "text/event-stream")
30             .GET().build();
31
32         HttpResponse<Stream<String>> res = http.send(req, HttpResponse.BodyHandlers.ofLines());
33         if (res.statusCode() != 200) throw new RuntimeException("HTTP " + res.statusCode());
34
35         StringBuilder data = new StringBuilder();
36         res.body().forEach(line -> {                                // lines arrive as the server sends them
37             if (line.startsWith("data:")) {
38                 data.append(line.substring(5).trim()).append('\n');
39             } else if (line.isEmpty() && data.length() > 0) {
40                 onEvent.accept(data.toString().trim());
41                 data.setLength(0);
42             }
43         });
44     }
45
46     /** POST /betValidation (id = your own ID, dataJson = the bet slip JSON array) */
47     public String betValidation(long id, String dataJson) throws Exception {
48         String body = "{\"id\": " + id + ", \"data\": " + dataJson + "}";
49         HttpRequest req = HttpRequest.newBuilder(URI.create(BASE_URL + "/betValidation"))
50             .header("x-api-key", apiKey)
51             .header("Content-Type", "application/json")
52             .POST(HttpRequest.BodyPublishers.ofString(body)).build();
53         return http.send(req, HttpResponse.BodyHandlers.ofString()).body();
54     }
55
56     // ----- usage -----
57     public static void main(String[] args) throws Exception {
58         String key = Optional.ofNullable(System.getenv("SPORTSB00K_API_KEY")).orElse("YOUR_API_KEY");
59         new SportsbookSseClient(key).stream("inplay", Map.of("id", 1, "country", "NG"),
60             event -> System.out.println("event: " + event));
61     }
62 }

```

GUIDANCE

Best Practices & Notes

1

Close streams you no longer need

Abort the request (or close the connection) when the user leaves the screen. Open streams keep using connections on both sides.

2

Reconnect with a delay

If the connection drops, wait a moment and reconnect. Increase the delay after repeated failures instead of reconnecting in a tight loop.

3

Keep your API key private

Keys in browser code are visible to visitors. Restrict the allowed origin, or use your own backend as a proxy for public websites.

4

Use the exact allowed origin

Do not use a wildcard origin for credential-sensitive browser setups. Allow only the exact origin, for example `https://yourdomain.com`.

5

Parse messages by blank lines

A message ends with a blank line. Join multiple `data:` lines with a newline, and ignore comment lines that start with a colon.

6

URL-encode your values

Use your language's query builder (as in the examples). For `/betSlip`, encode the slip JSON with `encodeURIComponent`.

HTTP status handling

Status	Meaning	What to do
200	The request succeeded and the stream opens.	Read the messages as they arrive.
4xx	The request was rejected, for example a missing parameter, a wrong key or a blocked origin.	Check the headers, parameters and allowed origin.
5xx	A temporary problem on the server side.	Wait a moment and reconnect.

Ready to build?

Open a stream on `/inplay` with your `x-api-key` header. For more information visit bet365soft.com.