Integration documentation

Everything you need to read the exchange catalog, its live prices and live streams from your own platform.

Overview

The API answers plain web requests with JSON. The catalog has four levels, each linked to the one above by ID:

```
Sport → Tournament → Event → Market (with runners)
```

Base address for every request:

```
https://apimanual.com/api/v1
```

All IDs are strings. Sport, tournament and event IDs are numeric strings such as `1` or `36065329`; market IDs look like `1.262354006`. Runner IDs (`selectionId`) are numbers. All times are UTC in ISO 8601 format.

Every sport, tournament, event and market also carries a `source` field: `betfair` or `stakefair`. Stakefair IDs start with 9 (sports `9001`, tournaments `90001`, events `900001`, markets `9.000001`).

Authentication

Every API call needs two things, sent as headers: your **App Key** and a **session token**. If you have integrated with Betfair before, this works the same way.

HEADER	VALUE
`X-Application`	Your App Key. It is created when you sign up (your first month is free); read it in the client area under App Key, where you can also create a new one.
`X-Authentication`	A session token. You get it by logging in, as shown below.

Server IPs. The API accepts calls only from the IP addresses you have registered in the client area under **Server IPs**: the public IPs of the servers that call the API. A call from any other address, including

the login, is refused with `403` and the code `IP_NOT_WHITELISTED`; the message says which address the call came from. Add your server's IP before your first call.

POST /login

Log in with your App Key in the header and your username and password in the body (JSON or form encoded). The reply contains the session token.

```
curl -X POST "https://apimanual.com/api/v1/login" \
  -H "X-Application: YOUR_APP_KEY" \
  -H "Content-Type: application/json" \
  -d '{"username":"YOUR_USERNAME","password":"YOUR_PASSWORD"}'

{ "token": "p2Qk...the session token...9fA", "status": "SUCCESS", "expiresAt": "2026-10-
08T11:30:00.000Z" }
```

POST /keepAlive

A session lasts 12 hours. Call this (with both headers) to extend it by another 12 hours, so a long-running system never has to log in again. Calling it once an hour is plenty.

```
{ "status": "SUCCESS", "expiresAt": "2026-10-08T12:30:00.000Z" }
```

POST /logout

Ends the session (with both headers). The token stops working.

```
{ "status": "SUCCESS" }
```

Keep the App Key, password and session token on your server; do not put them in a website or mobile app. When a call answers `401` with the code `INVALID_SESSION_INFORMATION`, log in again and repeat the call.

Quick start

Add your server's IP under Server IPs in the client area, log in once, then list sports and walk down to the markets of one event.

curl

```
# 1. Log in and keep the token
curl -X POST "https://apimanual.com/api/v1/login" -H "X-Application: YOUR_APP_KEY" \
  -H "Content-Type: application/json" \
  -d '{"username":"YOUR_USERNAME","password":"YOUR_PASSWORD"}'

# 2. Sports
curl -H "X-Application: YOUR_APP_KEY" -H "X-Authentication: YOUR_TOKEN"
"https://apimanual.com/api/v1/exchange/sports"

# 3. Tournaments of a sport (1 = Soccer)
curl -H "X-Application: YOUR_APP_KEY" -H "X-Authentication: YOUR_TOKEN"
"https://apimanual.com/api/v1/exchange/tournaments?sportId=1"

# 4. Events of a tournament
curl -H "X-Application: YOUR_APP_KEY" -H "X-Authentication: YOUR_TOKEN"
"https://apimanual.com/api/v1/exchange/events?tournamentId=10932509"

# 5. Markets of an event
curl -H "X-Application: YOUR_APP_KEY" -H "X-Authentication: YOUR_TOKEN"
"https://apimanual.com/api/v1/exchange/markets?eventId=36065329"
```

JavaScript (Node.js 18+)

```
const BASE = "https://apimanual.com/api/v1";
const APP_KEY = process.env.APP_KEY;
let token;

async function login() {
  const res = await fetch(BASE + "/login", {
    method: "POST",
    headers: { "X-Application": APP_KEY, "Content-Type": "application/json" },
    body: JSON.stringify({ username: process.env.API_USERNAME, password: process.env.API_PASSWORD }),
  });
  const data = await res.json();
  if (!res.ok) throw new Error(data.error);
  token = data.token;
}

async function get(path, retry = true) {
  if (!token) await login();
  const res = await fetch(BASE + path, {
    headers: { "X-Application": APP_KEY, "X-Authentication": token },
  });
  const data = await res.json();
  // The session expired: log in again once and repeat the call.
  if (res.status === 401 && data.code === "INVALID_SESSION_INFORMATION" && retry) {
    token = null;
    return get(path, false);
  }
  if (!res.ok) throw new Error(data.error);
  return data;
}

const sports = await get("/exchange/sports");
const events = await get("/exchange/events?sportId=" + sports.data[0].id);
console.log(events.total, "events");
```

Python

```
import os, requests

BASE = "https://apimanual.com/api/v1"
APP_KEY = os.environ["APP_KEY"]

def login():
    res = requests.post(BASE + "/login", headers={"X-Application": APP_KEY},
                        json={"username": os.environ["API_USERNAME"], "password":
os.environ["API_PASSWORD"]},
                        timeout=30)
    res.raise_for_status()
    return res.json()["token"]

token = login()

def get(path, **params):
    headers = {"X-Application": APP_KEY, "X-Authentication": token}
    res = requests.get(BASE + path, headers=headers, params=params, timeout=30)
    res.raise_for_status()
    return res.json()

sports = get("/exchange/sports")
markets = get("/exchange/markets", eventId="36065329")
print(markets["total"], "markets")
```

Responses and paging

Every list has the same shape:

```
{
  "data": [ ... ],   // the rows
  "total": 1944,     // rows matching your request, across all pages
  "limit": 100,
  "offset": 0
}
```

PARAMETER	MEANING
`limit`	Rows to return. Default 100, maximum 1000.
`offset`	Rows to skip. Default 0.

To read everything, keep asking with `offset` increased by `limit` until you have `total` rows.

Sports

GET /exchange/sports

All sports available to you.

```
{
  "data": [
    { "id": "1", "name": "Soccer" },
    { "id": "2", "name": "Tennis" },
    { "id": "7", "name": "Horse Racing" }
  ],
  "total": 22, "limit": 100, "offset": 0
}
```

Tournaments

GET /exchange/tournaments

PARAMETER	MEANING
sportId	Only tournaments of this sport. Optional.
limit	Rows to return. Default 100, maximum 1000.
offset	Rows to skip. Default 0.

```
{
  "data": [
    { "id": "10932509", "name": "English Premier League", "region": "GBR", "sportId": "1" }
  ],
  "total": 179, "limit": 100, "offset": 0
}
```

Events

GET /exchange/events

PARAMETER	MEANING
`sportId`	Only events of this sport. Optional.
`tournamentId`	Only events of this tournament. Use `none` for events that belong to no tournament. Optional.
`limit`	Rows to return. Default 100, maximum 1000.
`offset`	Rows to skip. Default 0.

Events are ordered by start time.

```
{
  "data": [
    {
      "id": "36065329",
      "name": "Liverpool v Man City",
      "countryCode": "GB",
      "timezone": "Europe/London",
      "venue": null,
      "startTime": "2026-10-11T15:30:00.000Z",
      "sportId": "1",
      "tournamentId": "10932509",   // null when the event has no tournament
      "totalMatched": 113638.36     // sum over the event's markets
    }
  ],
  "total": 21, "limit": 100, "offset": 0
}
```

Markets

GET `/exchange/markets`

PARAMETER	MEANING
`eventId`	Only markets of this event. Optional, but recommended.
`limit`	Rows to return. Default 100, maximum 1000.
`offset`	Rows to skip. Default 0.

```
{
  "data": [
    {
      "id": "1.262354006",
      "name": "Match Odds",
      "marketType": "MATCH_ODDS",
      "bettingType": "ODDS",
      "startTime": "2026-10-11T15:30:00.000Z",
      "eventId": "36065329",
      "totalMatched": 72859.78,
      "runners": [
        { "selectionId": 56323, "runnerName": "Liverpool", "handicap": 0, "sortPriority": 1,
"metadata": { "runnerId": "56323" } },
        { "selectionId": 47999, "runnerName": "Man City",  "handicap": 0, "sortPriority": 2,
"metadata": { "runnerId": "47999" } },
        { "selectionId": 58805, "runnerName": "The Draw",  "handicap": 0, "sortPriority": 3,
"metadata": { "runnerId": "58805" } }
      ],
      "description": {
        "marketTime": "2026-10-11T15:30:00.000Z",
        "suspendTime": "2026-10-11T15:30:00.000Z",
        "bettingType": "ODDS",
        "marketType": "MATCH_ODDS",
        "turnInPlayEnabled": true,
        "persistenceEnabled": true,
        "bspMarket": true,
        "marketBaseRate": 2,
        "rules": "..."
      }
    }
  ],
  "total": 9, "limit": 100, "offset": 0
}
```


FIELD	MEANING
`runners[].selectionId`	The runner ID. Use it to match a runner to its result.
`runners[].handicap`	The line, for handicap markets; otherwise 0.
`runners[].metadata`	Extra runner facts where they exist, for example jockey and trainer in horse racing.
`totalMatched`	Total amount matched on the market.
`description`	Market settings and rules.

For a market that appeared within the last 10 minutes, `description` can be `null` and each runner has only `selectionId` and `name`. The full details follow on the next refresh.

Live prices

Back and lay prices of Exchange markets, from the Betfair Exchange Stream API. Ask for the current prices of a few markets, or keep a stream open and receive every change as it happens. Both take the market IDs from the markets list, up to 500 at a time, and are part of the Exchange service.

GET `/exchange/prices?marketIds=1.262354006,1.262354010`

The current prices of the markets, in the same shape as Betfair's listMarketBook.


```
{
  "data": [
    {
      "marketId": "1.262354006",
      "status": "OPEN",
      "inplay": true,
      "totalMatched": 228748.85,
      "updatedAt": "2026-10-11T15:41:07.512Z",
      "runners": [
        {
          "selectionId": 56323, "handicap": 0, "status": "ACTIVE",
          "lastPriceTraded": 1.16, "totalMatched": 216567.51,
          "ex": {
            "availableToBack": [ { "price": 1.14, "size": 2263.86 }, { "price": 1.13, "size": 6554.47
}, { "price": 1.12, "size": 4680.25 } ],
           "availableToLay":  [ { "price": 1.15, "size": 189.23 },  { "price": 1.16, "size": 2621.54
}, { "price": 1.17, "size": 1655.98 } ]
          }
        }
      ]
    }
  ],
  "pending": ["1.262354010"],
  "unavailable": []
}
```

FIELD	MEANING
`status`	`OPEN`, `SUSPENDED` or `CLOSED`.
`inplay`	Whether the event has started and the market is trading in play.
`ex.availableToBack`	The best three prices to back, best first, with the amount available at each.
`ex.availableToLay`	The best three prices to lay, best first.
`lastPriceTraded`	The price of the last matched bet on the runner.
`updatedAt`	When Betfair published these prices.
`pending`	Markets that were not being streamed yet. They start now; ask again in a few seconds.
`unavailable`	IDs that are unknown, blocked for your account, or of markets without Betfair prices.

A market is streamed from Betfair while you ask for it. One that nobody has asked for yet is listed under `pending` and has prices a few seconds later; a stream keeps its markets live for as long as it is open, and a market asked for here stays live for a minute after your last request.

GET /exchange/stream?marketIds=1.262354006,1.262354010

A stream of Server-Sent Events that stays open. It starts with each market's whole book, then sends only what changed. A changed side of a runner's prices is sent whole, so it simply replaces what you have. You can keep up to 5 streams open (your account manager can allow more; /me shows your limit), each with up to 500 markets; put many markets in one stream rather than one stream per market.

EVENT	DATA
`subscribed`	Sent first: `markets` streamed, `pending` (their book follows as soon as it arrives) and `unavailable`.
`image`	One market's whole book, the same as in /exchange/prices. Replace what you have for that market.
`change`	What changed in one market: `status`, `inplay`, `totalMatched` and the `runners` that changed, each with only its changed fields.
`end`	The stream is about to close, with a `code`: for example `INVALID_SESSION_INFORMATION` when the session token expires. Log in again and reopen.

```
event: subscribed
data: {"markets":["1.262354006","1.262354010"],"pending":[],"unavailable":[]}

event: image
data:
{"marketId":"1.262354006","status":"OPEN","inplay":true,"totalMatched":228748.85,"updatedAt":"...","r
unners":[...]}

event: change
data: {"marketId":"1.262354006","updatedAt":"...","totalMatched":228761.2,"runners":
[{"selectionId":56323,"handicap":0,"lastPriceTraded":1.15,"totalMatched":216579.86,"ex":
{"availableToBack":[{"price":1.15,"size":310.4},{"price":1.14,"size":2263.86},
{"price":1.13,"size":6554.47}]}}]}

: keep-alive
```

A line starting with a colon arrives every 15 seconds and can be ignored. If the connection drops, open the stream again: it starts with fresh images.

JavaScript (Node.js 18+)

```
const res = await fetch(BASE + "/exchange/stream?marketIds=1.262354006", {
  headers: { "X-Application": APP_KEY, "X-Authentication": token },
});
const books = new Map();
let buffer = "";
for await (const chunk of res.body.pipeThrough(new TextDecoderStream())) {
  buffer += chunk;
  let end;
  while ((end = buffer.indexOf("\n\n")) >= 0) {
    const block = buffer.slice(0, end);
    buffer = buffer.slice(end + 2);
    const event = block.match(/^event: (.*)$/m)?.[1];
    const data = block.match(/^data: (.*)$/m)?.[1];
    if (!event || !data) continue; // a keep-alive
    const message = JSON.parse(data);
    if (event === "image") books.set(message.marketId, message);
    if (event === "change") applyChange(books.get(message.marketId), message);
  }
}

// Merge a change into the book: market fields, then each changed runner's fields.
function applyChange(book, change) {
  if (!book) return;
  for (const key of ["status", "inplay", "totalMatched", "updatedAt"]) if (key in change) book[key] =
change[key];
  for (const update of change.runners ?? []) {
    const runner = book.runners.find((r) => r.selectionId === update.selectionId && r.handicap ===
update.handicap);
    if (!runner) { book.runners.push(update); continue; }
    const { ex, ...rest } = update;
    Object.assign(runner, rest);
    if (ex) runner.ex = { ...runner.ex, ...ex };
  }
}
```

Results

`GET` `/exchange/results`

Results of finished markets, newest first. A result is kept for 15 days and then deleted, so store what you need on your side.

PARAMETER	MEANING
`marketId`	The result of one market. Optional.
`eventId`	All market results of one event. Optional.
`tournamentId`	All results in one tournament. Optional.
`sportId`	All results in one sport. Optional.
`limit`	Rows to return. Default 100, maximum 1000.
`offset`	Rows to skip. Default 0.


```
{
  "data": [
    {
      "marketId": "1.263353090",
      "marketName": "Over/Under 1.5 Goals",
      "marketType": "OVER_UNDER_15",
      "totalMatched": 6172.27,
      "marketStartTime": "2026-10-07T22:00:00.000Z",
      "eventId": "36154409",
      "eventName": "Antofagasta v Universidad de Chile",
      "eventStartTime": "2026-10-07T22:00:00.000Z",
      "sportId": "1",
      "sportName": "Soccer",
      "tournamentId": "961354",
      "tournamentName": "Chilean Cup",
      "winners": [ { "selectionId": 1221386, "name": "Over 1.5 Goals" } ],
      "runners": [
        { "selectionId": 1221385, "name": "Under 1.5 Goals", "status": "LOSER" },
        { "selectionId": 1221386, "name": "Over 1.5 Goals",  "status": "WINNER" }
      ],
      "settledAt": "2026-10-07T23:22:21.207Z"
    }
  ],
  "total": 35, "limit": 100, "offset": 0
}
```

FIELD	MEANING
`winners`	The winning runners. Empty when the market was voided.
`runners[].status`	Outcome of each runner: `WINNER`, `LOSER` or `REMOVED`.
`settledAt`	When the result was recorded.

Blocking

Block anything you do not want delivered to you. A blocked item disappears from every list above, together with everything beneath it: blocking a sport also hides its tournaments, events, markets and results. Blocks apply to your account only. You can manage them in the client area or with these requests.

GET /exchange/blocks

Everything you have blocked.


```
{ "data": [ { "level": "sport", "id": "2", "blockedAt": "2026-10-07T23:09:43.221Z" } ] }
```


POST /exchange/blocks

Block one item. `level` is `sport`, `tournament`, `event` or `market`.

```
curl -X POST "https://apimanual.com/api/v1/exchange/blocks" \
  -H "X-Application: YOUR_APP_KEY" -H "X-Authentication: YOUR_TOKEN" \
  -H "Content-Type: application/json" -d '{"level":"sport","id":"2"}'

{ "blocked": true, "level": "sport", "id": "2" }
```

DELETE /exchange/blocks?level=sport&id=2

Remove a block.


```
{ "blocked": false, "level": "sport", "id": "2" }
```

To see blocked items in a list, flagged instead of hidden, add `includeBlocked=true` to the sports, tournaments, events or markets request. Each row then carries `"blocked": true` or `false`.

Live streaming

Live video of matches, for the Live Streaming service. Events are identified by their Betfair event ID, the same ID as in the Exchange events list. The video plays only on your registered websites: tell your account manager the website where you show it, for example `watch.yoursite.com`.

GET /live-streaming/events

Live and upcoming matches that have video. Optional filters: `eventId`, `sportType` (`FOOTBALL`, `CRICKET`, `TENNIS`, …) and `status`.


```
{
  "data": [
    {
      "eventId": "36134279",
      "title": "India v Australia",
      "sportType": "CRICKET",
      "competition": "ICC World Cup",
      "startTime": "2026-10-02T09:30:00Z",
      "status": "LIVE",
      "exchangeEvent": { "id": "36134279", "name": "India v Australia", "sportId": "4", "sportName":
"Cricket", "tournamentId": "12345", "tournamentName": "ICC World Cup", "startTime": "2026-10-
02T09:30:00.000Z" }
    }
  ],
  "total": 1,
  "updatedAt": "2026-10-02T09:41:00.000Z"
}
```


FIELD	MEANING
`status`	`LIVE`, `UPCOMING` or `OFFLINE`. Show the player only while it is `LIVE`; video usually starts about 2 minutes before the start.
`exchangeEvent`	The same event in the Exchange catalog, or null when it is not there.
`updatedAt`	When the list was last read. It is refreshed every minute.

POST /live-streaming/links

A watch link for one viewer. Ask for a new one every time a viewer presses play, from your server: the link must be opened within a minute, plays on one device only and cannot be reused.

```
curl -X POST "https://apimanual.com/api/v1/live-streaming/links" \
  -H "X-Application: YOUR_APP_KEY" -H "X-Authentication: YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"eventId":"36134279","domain":"watch.yoursite.com","viewerRef":"user-
123","viewerIp":"203.0.113.7"}'

{ "eventId": "36134279", "url": "https://tv.ltve.live/w/Xb3kP9...", "expiresAt": "2026-10-
02T09:42:00.000Z", "singleUse": true }
```

FIELD	MEANING
`eventId`	The Betfair event ID. Required.
`domain`	The website where the video plays. Required; one of your registered websites (www. and other forms of it work too).
`viewerRef`	Your own reference for the viewer, for example a user ID. Recommended: it is shown as a watermark on the video.
`viewerIp`	The viewer's IP address. Optional.

Show the link in an iframe on that website, served over HTTPS:

```
<div style="position:relative;aspect-ratio:16/9;max-width:100%">
  <iframe src="URL_FROM_THE_LINK"
          style="position:absolute;inset:0;width:100%;height:100%;border:0"
          allow="autoplay; fullscreen; encrypted-media" allowfullscreen></iframe>
</div>
```

The video starts muted (a browser rule); viewers tap the speaker to unmute. Do not set `referrerpolicy="no-referrer"` on the iframe or the page: the video needs it to know your website. Do not record, re-stream or pass the video through your own servers. At most 30 links a minute can be asked for.

Account

GET `/me`

The account behind your App Key and session. Useful as a connection test.

{ "name": "Your Company", "company": null, "username": "yourname", "plan": "Standard", "services": ["exchange"], "status": "ACTIVE", "expiresAt": null, "limits": { "streams": 5, "marketsPerStream": 500 } }

Data freshness

DATA	REFRESHED
Sports	Every 12 hours
Tournaments	Every 6 hours
Events	Every 3 hours
Markets, runners, total matched	Every 10 minutes
Live prices	As they change at Betfair, through /exchange/stream
Results	Recorded within 10 minutes of a market finishing

Asking more often than these intervals returns the same data. A finished market leaves the markets list and appears under results; an event leaves the events list once all of its markets have finished.

Errors

Errors come back as JSON with a message and, for sign-in problems, a code: `{ "error": "...", "code": "..." }`

STATUS	MEANING
200	Success.
400	A request that is missing something, for example a block without a level and id.
401	A problem with the App Key, the session token or the login. See the codes below.
403	The account is suspended or has expired, or the call came from an IP that is not on your Server IPs list.
404	Unknown address, or a live stream that does not exist for the event.
429	Too many failed logins (wait 15 minutes), too many live price streams open, or too many watch links.
503	Live prices or live streaming are not available for a moment. Try again shortly.

CODE	WHAT TO DO
NO_APP_KEY	Add the X-Application header.
NO_SESSION	Add the X-Authentication header. Log in first to get a token.
INVALID_APP_KEY	The App Key is wrong, or you created a newer one. Use the current key.
INVALID_SESSION_INFORMATION	The session token is wrong or has expired. Log in again.
INVALID_USERNAME_OR_PASSWORD	Check the username and password.
TEMPORARILY_LOCKED	Five failed logins in a row. Wait 15 minutes.
ACCOUNT_NOT_ACTIVE	The account is suspended. Contact your account manager.
SERVICE_NOT_ENABLED	The service you called is not part of your account. Choose it under Services in the client area.
SUBSCRIPTION_EXPIRED	Your free or paid month has ended. Renew under Services in the client area.
IP_NOT_WHITELISTED	The call came from an IP that is not on your list. Add your server's IP under Server IPs in the client area.
INVALID_MARKET_IDS	Send marketIds: one to 500 market IDs, separated by commas.
TOO_MANY_STREAMS	All the live price streams your account allows are open (5 unless agreed otherwise). Close one, or put more markets in one stream.
PRICES_UNAVAILABLE	Live prices cannot be read right now. Try again shortly.
NO_STREAM	This event has no live video. Show no player.
DOMAIN_NOT_REGISTERED	The website is not registered for your live streams. Ask your account manager to add it.
VIEWER_LIMIT_REACHED	The limit of viewers watching at once is reached. Try again later.
RATE_LIMITED	Too many watch links in a minute. Slow down.
STREAMING_UNAVAILABLE	Live streaming cannot be reached right now. Try again shortly.