AL ALB365

Integration Guide

# Ticket API

Register every bet slip with ALB365. An accepted ticket receives a guaranteed result for each of its selections, even if a settlement message was missed on the feed.

| API base | Format | Updated |
| --- | --- | --- |
| https://radar.albet365.xyz/v1 | XML | 27 November 2026 |

## 1. Overview

Send every bet slip your customers place to `POST /v1/bets`. Every selection on an **accepted** ticket is guaranteed a result. Results still arrive through the regular `bet_settlement` messages on your feed; the Ticket API adds no new feed message.

1. Your customer places a bet on odds from your ALB365 feed.
2. You send the ticket to `POST /v1/bets` under your own `ticket_id`.
3. The answer is **accepted** with a `bet_id`, or **rejected** with a reason.
4. Selections are settled through `bet_settlement`; `GET /v1/bets/{ticket_id}` shows the same results per ticket.

---

## 2. Base URL and authentication

| Item | Value |
|---|---|
| Base URL | `https://radar.albet365.xyz/v1` (same host and API key as the feed REST API) |
| Auth header | `x-access-token: <API key>` — never put it in a URL |
| Request body | XML, `Content-Type: application/xml` (`text/xml` also accepted) |
| Max request size | 64 KB |
| Response body | XML, `application/xml; charset=utf-8` |
| Client timeout | ≥ **75 seconds** |
| Results | Every selection of an accepted ticket is settled automatically. Results arrive as `bet_settlement` messages on your **Odds Feed** (so the feed product is needed); without the feed, poll `GET /v1/bets/{ticket_id}`. |
| IP whitelist | Only your registered server IPs |

You can send and read only your own tickets.

---

## 3. Sending a ticket

**POST** `/v1/bets`

One request carries one bet slip. Copy every selection value exactly from the `odds_change` message the odds came from.

```
<ticket ticket_id="T-1001" timestamp="1794589825684">
  <stake value="12.50" currency="EUR"/>
  <selection event_id="sr:match:12345678" product="3" market_id="18" specifiers="total=2.5" outcome_id="12"
odds="1.85"/>
  <selection event_id="sr:match:23456789" product="1" market_id="1" outcome_id="1" odds="2.10"/>
</ticket>
```

| Element / attribute | Required | Description |
|---|---|---|
| `ticket/@ticket_id` | Yes | Your unique id for this slip. 1–79 characters: letters, digits and `. _ : -`, starting with a letter or digit. Never reuse it. |
| `ticket/@timestamp` | Yes | When the bet was placed, Unix time in milliseconds. |
| `stake/@value` | Yes | Total stake, decimal with a dot (`12.50`). |
| `stake/@currency` | Yes | ISO 4217 code (`EUR`, `USD`, `TRY` …). |
| `selection` | 1 or more | One element per leg; several make an accumulator. |
| `selection/@event_id` | Yes | Event id from the feed, e.g. `sr:match:12345678`. |
| `selection/@product` | Yes | `1` = live, `3` = pre-match: the product of the message the odds came from. |
| `selection/@market_id` | Yes | Market id from the feed. |
| `selection/@specifiers` | If the market has them | Exactly as in the feed (`total=2.5`, `hcp=-15.5`); omit otherwise. |
| `selection/@outcome_id` | Yes | Outcome id from the feed. |
| `selection/@odds` | Yes | The odds your customer accepted, decimal with a dot. |

The root element must be `<ticket>`; a DOCTYPE is rejected, and unknown elements or attributes are rejected with 400. Send each ticket as soon as the bet is placed — a selection whose event has already closed comes back `missing`.

```
curl -sS -X POST "https://radar.albet365.xyz/v1/bets" \
  -H "x-access-token: YOUR_API_KEY" -H "Content-Type: application/xml" \
  --data-binary @ticket.xml
```

### Accepted

```
<ticket_response ticket_id="T-1001" status="accepted" bet_id="10708198" received_at="2026-11-13T13:14:00.000Z"
stake="12.5" currency="EUR">
  <selection event_id="sr:match:12345678" product="3" market_id="18" specifiers="total=2.5" outcome_id="12"
odds="1.85" status="ok"/>
  <selection event_id="sr:match:23456789" product="1" market_id="1" outcome_id="1" odds="2.10" status="ok"/>
</ticket_response>
```

### Rejected

```
<ticket_response ticket_id="T-1002" status="rejected" reason="SELECTION_UNAVAILABLE" received_at="…" stake="5"
currency="EUR">
  <selection event_id="sr:match:34567890" … status="missing" reason="EVENT_CLOSED"/>
  <selection event_id="sr:match:23456789" … status="ok"/>
</ticket_response>
```

Reason codes — ticket: `SELECTION_UNAVAILABLE`, `SOURCE_REJECTED`; selection: `EVENT_CLOSED`, `NOT_AVAILABLE`. New codes may appear: treat any unknown reason on a rejected ticket as a rejection.

## 4. Reading a ticket

```
GET /v1/bets/{ticket_id}
```

Returns the same `<ticket_response>`; each selection gains its result once settled:

```
<selection … status="ok" result="1" settlement="won" result_source="general"/>
<selection … status="ok" result="0" settlement="lost" result_source="general"/>
```

| Attribute | On | Meaning |
|---|---|---|
| `status` | ticket_response | `accepted` registered · `rejected` not registered · `error` not processed now, resend |
| `bet_id` | ticket_response | Id of the registered bet (accepted only) |
| `reason` | both | Why rejected / failed / missing |
| `received_at`, `stake`, `currency` | ticket_response | Arrival time (UTC) and your stake, echoed |
| `status` | selection | `ok` matched · `missing` could not be matched |
| `result` | selection | `1` won, `0` lost |
| `settlement` | selection | `won`, `lost`, `refund` … |
| `void_factor` | selection | Share of the stake refunded (`1`, `0.5`), when present |
| `result_source` | selection | `general` — same result as in `bet_settlement` |

The ticket's own status stays accepted or rejected; it never becomes won or lost — work out the slip result from its selections. Not found → HTTP 404 `<response response_code="NOT_FOUND">`.

## 5. Errors, retries and idempotency

Resending the same ticket (same `ticket_id`, identical XML) is always safe: it is never registered twice and the resend gets the original answer.

| HTTP | Body | Meaning | What to do |
|---|---|---|---|
| 400 | `BAD_REQUEST` | Invalid root / ticket_id, DOCTYPE, unknown content | Fix the XML; do not resend unchanged |
| 401 / 403 | `FORBIDDEN` | Missing / wrong key, IP not whitelisted | Check key and IP |
| 409 | `CONFLICT` | ticket_id already used with different content | Give a new slip a new ticket_id |
| 413 | `BAD_REQUEST` | Body larger than 64 KB | Send a smaller body |
| 415 | `BAD_REQUEST` | Content-Type is not XML | Send `application/xml` |
| 500 | `INTERNAL_ERROR` | No answer; the ticket may or may not have arrived | Resend the same ticket |
| 502 | `status="error" reason="SOURCE_ERROR"` | Ticket service error | Resend the same ticket |
| 503 | `<ticket_response status="error" reason="SOURCE_TIMEOUT\|SOURCE_UNAVAILABLE"/>` | Busy or down | Resend after `Retry-After` |
| 429 | `TOO_MANY_REQUESTS` | Your per-minute limit | Resend the identical ticket after `Retry-After` |

Back off between resends (1 s, 2 s, 5 s, 10 s …). Never change the XML or the ticket_id of a ticket you are resending. Branch on the HTTP status and on `response_code` / `reason`; the message text may change.

---

## 6. Integration checklist

- Unique `ticket_id` per slip (≤ 79 chars), stored with the bet.

- Selection values copied exactly from the feed.
- Ticket sent as soon as the bet is placed.
- `accepted` → store `bet_id`.
- 429 / 500 / 502 / 503 / timeout → resend the identical ticket, honouring `Retry-After`.
- Settle from `bet_settlement`; reconcile with `GET /v1/bets/{ticket_id}`.