AL ALB365

Integration Guide

Odds Feed

Real-time pre-match and live odds, fixtures, results and settlements for all sports — delivered over AMQP, fully compatible with the Unified Odds Feed (UOF) protocol and its official SDKs.

Protocol
UOF · AMQP 0.9.1 over TLS

Messaging host
mq.albet365.xyz:5671

API
https://radar.albet365.xyz/v1

Updated
27 November 2026

1. Your credentials

Item	Value
Messaging host	`mq.albet365.xyz`, port `5671`, TLS required
TLS certificate	If your client verifies server certificates and the connection is refused, trust our certificate: `https://radar.albet365.xyz/docs/amqp-ca.pem` (the UOF SDKs and most AMQP clients connect without extra settings).
Virtual host	`/unifiedfeed/<your bookmaker id>`
Username / password	Your feed credentials (from your account manager)
Exchange	`unifiedfeed` (topic)
API base	`https://radar.albet365.xyz/v1`
API authentication	Header `x-access-token: <API key>`
IP whitelist	Messaging and API connections are accepted only from your registered server IPs. Connections from other IPs are closed automatically.

2. Using the official SDK (recommended)

The UOF SDKs (Java, .NET) work unchanged — configure a custom environment:

```
// .NET
var config = UofSdk.GetConfigurationBuilder()
    .SetAccessToken("<API key>")
    .SelectCustom()

.SetMessagingHost("mq.albet365.xyz").SetMessagingPort(5671).UseMessagingSsl(true)
        .SetMessagingUsername("<feed username>").SetMessagingPassword("<feed
password>")
        .SetVirtualHost("/unifiedfeed/<bookmaker id>")
        .SetApiHost("radar.albet365.xyz").UseApiSsl(true)
        .SetNodeId(1)
    .SetDefaultLanguage(CultureInfo.GetCultureInfo("en"))
    .Build();
```

```
// Java
UofConfiguration config = UofSdk.getUofConfigurationBuilder()
    .setAccessToken("<API key>").selectCustom()

.setMessagingHost("mq.albet365.xyz").setMessagingPort(5671).useMessagingSsl(true)
    .setMessagingUsername("<feed username>").setMessagingPassword("<feed password>")
    .setMessagingVirtualHost("/unifiedfeed/<bookmaker id>")
    .setApiHost("radar.albet365.xyz").useApiSsl(true).setNodeId(1)
    .setDefaultLanguage(Locale.ENGLISH).build();
```

---

3. Plain AMQP clients

1. Connect with TLS to `mq.albet365.xyz:5671`, virtual host `/unifiedfeed/<bookmaker id>`, heartbeat 30 s.
2. Declare a **server-named, exclusive, auto-delete** queue (empty name).
3. Bind it to exchange `unifiedfeed` with routing key `#` (or narrower keys, always ending in `#`).
4. Consume; each message body is UOF XML.

```
// Node.js (amqplib)
const conn = await amqp.connect({ protocol: 'amqps', hostname: 'mq.albet365.xyz',
port: 5671,
  username: USER, password: PASS, vhost: '/unifiedfeed/<bookmaker id>', heartbeat:
30 });
const ch = await conn.createChannel();
const { queue } = await ch.assertQueue('', { exclusive: true, autoDelete: true,
durable: false });
await ch.bindQueue(queue, 'unifiedfeed', '#');
ch.consume(queue, (m) => handle(m.fields.routingKey, m.content.toString()), { noAck:
true });
```

Routing key

```
<priority>.<pre>.<live>.<message_type>.<sport_id>.<urn_type>.<event_id>.<node_id>
hi·–.live.odds_change.1.sr:match.12345678·–
```

Message types: `odds_change`, `bet_stop`, `bet_settlement`, `bet_cancel`, `rollback_bet_settlement`, `rollback_bet_cancel`, `fixture_change`, `alive`, `snapshot_complete`.

---

4. Producers, alive & recovery

- Each producer (1 = live odds, 3 = pre-match, …) sends `alive` every ~10 s. No alive for **more than 15 s** → the producer is down: stop accepting bets on its live / starting-soon markets (pre-match events starting later than 5 min: after 5 min).
- `<alive … subscribed="0"/>` means messages were lost (e.g. a connection break on our side): **start a recovery for that producer at once**. The UOF SDKs do this automatically.
- After connecting and after every producer-down, request a recovery with `after` = the timestamp of the last message you received (max 72 h back; without `after` you get current odds only). Keep the markets suspended until `snapshot_complete` with your `request_id` arrives.
- Delivery is at-least-once: the same message can arrive twice — process idempotently. Bind with patterns ending in `.#` and include `–·–·–.#` to get `alive` and `snapshot_complete`.
- Producers and their recovery addresses: `GET /v1/descriptions/producers.xml` (`api_url`). Virtual producers each need their own recovery.

```
POST /v1/{product}/recovery/initiate_request?after=<ms>&request_id=
<n>&node_id=<n>
```

`{product}` = `liveodds`, `pre`, … Recovery answers are delivered **only to your feed**, with your own `node_id`. A full recovery per producer is allowed once every 2 minutes per account. The feed's recovery quota is shared, so recoveries reaching further back are rarer (≥ 30 min back: a few per hour; ≥ 1 day back: 2 per 30 min). If the quota is used up, or the producer is not alive at that moment, you get HTTP 429 / 503 with `Retry-After` — wait and retry; never loop without pause.

```
POST /v1/{product}/odds/events/{event_urn}/initiate_request
```

```
POST /v1/{product}/stateful_messages/events/{event_urn}/initiate_request
```

Single-event recoveries (current odds of a not-started / live event; stateful messages up to 30 days back): max 30 per 10 minutes per account. Uncached REST calls are limited to 120 per minute per account; answer 429 → wait for `Retry-After`.

---

5. REST API

All Sports API and descriptions endpoints are available under `https://radar.albet365.xyz/v1` with header `x-access-token`. Examples:

Endpoint	Purpose
`GET /v1/users/whoami.xml`	Your bookmaker id, virtual host, expiry
`GET /v1/descriptions/en/markets.xml?include_mappings=true`	Market / outcome names
`GET /v1/sports/en/sport_events/{urn}/summary.xml`	Event summary, scores, results
`GET /v1/sports/en/sport_events/{urn}/fixture.xml`	Fixture, teams, TV / tracker info
`GET /v1/sports/en/schedules/live/schedule.xml`	Live schedule
`GET /v1/sports/en/schedules/{date}/schedule.xml`	Schedule for a day
`GET /v1/sports/en/competitors/{urn}/profile.xml`	Team profile

Responses are cached for a short time on our side (live data ~2 s). After a `fixture_change` message always fetch the fixture again — you get the new version. Descriptions change rarely — load them at start-up and refresh daily. Not available: replay server, integration (staging) environment, `users/*` except `whoami.xml`.

---

6. WebSocket alternative

For systems without AMQP: `wss:// radar.albet365.xyz/feed/ws` with header `x-access-token: <API key>`. Every frame:

```
{ "routing_key": "hi·–.live.odds_change.1.sr:match.123·–", "payload": "<odds_change …/>", "timestamp": 1732350000000 }
```

Same messages and routing keys as AMQP. Max 5 concurrent connections. The first frame (`–·–·–.welcome·–·–·–·–`) only confirms the connection — ignore it. Reconnect with back-off

and run a recovery after every reconnect.

7. Errors

```
<response response_code="FORBIDDEN"><action>IP_NOT_ALLOWED</action><message>…
</message></response>
```

HTTP	Meaning
401 / 403	Invalid key, IP not whitelisted, service not enabled, account suspended
404	Resource not found
429	Rate / recovery limit — honour `Retry-After`
503	Temporarily unavailable — retry with back-off