AL ALB365

Integration Guide

Casino API

Seamless-wallet casino aggregation: thousands of slot games from 30+ studios through one API. Player balances stay in your own wallet.

Version	API base	Updated
4.0	https://casino.albet365.xyz	27 November 2026

1. Overview

The integration has two directions:

- **You → ALB365**: create players, list games, open a game (REST, JSON).
- **ALB365 → You**: wallet callbacks (`authenticate` , `balance` , `bet` , `win` , `cancel` , `status`). Your server keeps the balance and answers every callback.

1. Overview
2. Access & security
3. Players
4. Game catalog
5. Launching a game
6. Wallet callback
7. Callback commands
8. Error codes
9. Ready-to-use callback handler
10. Test before go-live
11. Go-live checklist

---

2. Access & security

Item	Value
Base URL	`https://casino.albet365.xyz`
Authentication	Header `Authorization: Bearer <API key>`
Content type	`application/json` , all requests are `POST`
IP whitelist	Only the server IPs you registered with us can call the API. Others receive `IP_NOT_ALLOWED` .
Rate limit	Per account and minute (default 600). Exceeding it returns HTTP 429.

Every response has the same envelope:

```
{ "code": 0, "message": "SUCCESS", "data": { … } }      // code ≠ 0 → error, see §8
```

Important. Never call the API from a browser or mobile app. Your API key must stay on your server.

---

3. Players

`POST` `/v4/user/create`

Creates the game account for one of your players and returns its `user_code` . Idempotent: the same `name` always returns the same `user_code` , so you can call it before every launch.

```
// request
{ "name": "player123" }          // your player id/username, 1–64 chars [A-Za-z0-9_.@-]
// response
{ "code": 0, "message": "SUCCESS", "data": { "user_code": 100245 } }
```

Store the `user_code` with your player. Callbacks identify the player by `account` = this `user_code` .

4. Game catalog

POST `/v4/game/providers`

```
{ "lang": 1 }   →   data: [ { "provider_id": 31, "provider_name": "…", "status": 1 }, … ]
```

POST `/v4/game/games`

```
{ "provider_id": 0, "lang": 1 }   // 0 = all providers
→ data: [ { "provider_id": 31, "game_code": "…", "game_name": "…", "game_image": "https://…", … }, … ]
```

Cache the catalog on your side and refresh it a few times per day. Only the providers enabled for your account are returned.

5. Launching a game

POST `/v4/game/game-url`

```
{
  "user_code": 100245,
  "provider_id": 31,
  "game_symbol": "game_code from the catalog",
  "lang": 1,
  "return_url": "https://your-site.com/casino"
}
→ { "code": 0, "data": { "game_url": "https://…" } }
```

Open `game_url` in an iframe or a new window. Request a new URL for every launch — URLs are session-bound.

POST `/v4/agent/info`

Returns your account card (agent code, currency, number of players, configured callback URL).

6. Wallet callback

ALB365 calls **your callback URL** (registered with your account manager) for every wallet operation:

Item	Value
Method	`POST`, `Content-Type: application/json`
Authentication	Header `callback-token: <your callback token>` — **verify it on every call** and reject anything else.
Source IP	Callbacks come only from `46.4.171.117` — whitelist it on your firewall and reject other sources.
Timeout	**bet** and **balance**: answer within **1.5 seconds**. All other commands: within **3 seconds**. A late answer is treated as failed (a late `bet` is cancelled automatically). Typical answers take 20–200 ms: keep the handler simple — one database transaction, no calls to other services.
Amounts	Decimal currency units, 2 decimals (e.g. `2.50`).

```
// request from ALB365
{ "command": "bet", "data": { "account": 100245, "user_name": "player123", "trans_guid": "8f1c…", "amount": 2.50,
  "provider_id": 31, "game_code": "…", "round_id": "…" } }
// your answer
{ "result": 0, "status": "OK", "data": { "balance": 47.50 } }
```

`account` is the `user_code` returned by `/v4/user/create`; `user_name` is the name you sent there — identify the player by either.

Idempotency is mandatory. The same `trans_guid` can arrive more than once (network retries). Apply it once, and answer every repeat with the same result and the current balance.

7. Callback commands

command	data	Your action	Answer data
`authenticate`	`account`	Check the player exists and may play	`{account, balance}`
`balance`	`account`	Return the balance	`{balance}`
`bet`	`account, trans_guid, amount, round_id, provider_id, game_code`	Debit. Insufficient funds → `2006`	`{balance}`
`win`	same as bet	Credit (may be 0). Always accept.	`{balance}`
`cancel`	`account, trans_guid, cancel_trans_guid`	Refund the bet `cancel_trans_guid` once. Unknown bet → answer OK, refund nothing.	`{balance}`
`status`	`account, trans_guid`	Was this transaction processed? `trans_status` = `OK` or `NOT_FOUND`	`{trans_status}`

Extra fields you may receive and can store for your reports: `round_finished` (true on the last transaction of a round), `game_type`, `gplay_id`, `type` (1 bet · 2 win · 16 cancel · 32 bonus win), `call_id`, `time_stamp`. Unknown fields must be ignored.

`win` and `cancel` are re-sent until you answer `result: 0` (for up to 72 hours, with growing pauses). A `bet` that is not answered OK within the timeout is not placed: ALB365 then sends you a `cancel` for it (`cancel_trans_guid` = that bet), so a debit you made too late is always refunded. A cancel for a bet you never booked → answer OK, refund nothing.

8. Error codes

Code	Meaning
`0`	Success
`1`	Game or studio under maintenance — show "temporarily unavailable" and try again later
`1001`	Internal error — retry later
`1002`	Validation error (missing / wrong field)
`1009`	Invalid token, IP not allowed, account disabled
`1012`	Parameters invalid (callback)
`1013`	Provider not enabled for your account
`1014`	Callback URL not configured for your account
`1429`	Rate limit exceeded
`1503`	Service temporarily unavailable
`2002`	User not found
`2006`	Insufficient funds (your callback answer)

9. Ready-to-use callback handler

Copy one of these into your project, connect the four wallet functions to your own player table, and your casino is live. Both handle token check, idempotency and every command.

PHP (8.x, PDO)

```php
<?php // casino-callback.php — register its public HTTPS address as your callback URL
const CALLBACK_TOKEN = 'your callback token';      // from your integration pack
header('Content-Type: application/json');
$answer = fn($result, $status, $data = null) => exit(json_encode(['result' => $result, 'status' => $status, 'data' => $data]));

// 1. Only ALB365 may call you
if (!hash_equals(CALLBACK_TOKEN, $_SERVER['HTTP_CALLBACK_TOKEN'] ?? '')) $answer(1009, 'TOKEN_INVALID');
$in = json_decode(file_get_contents('php://input'), true);
$cmd = $in['command'] ?? ''; $d = $in['data'] ?? [];

$pdo = new PDO('mysql:host=localhost;dbname=site', 'user', 'pass', [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);
// players(user_code, balance)   casino_tx(trans_guid UNIQUE, type, user_code, amount, balance_after)
$player = function ($uc, $lock = false) use ($pdo) {
  $q = $pdo->prepare('SELECT user_code, balance FROM players WHERE user_code=?' . ($lock ? ' FOR UPDATE' : ''));
  $q->execute([$uc]); return $q->fetch(PDO::FETCH_ASSOC);
};
$p = $player($d['account'] ?? 0);
if (!$p) $answer(2002, 'USER_NOT_FOUND');
$bal = fn($v) => round((float)$v, 2);

switch ($cmd) {
  case 'authenticate': $answer(0, 'OK', ['account' => $d['account'], 'balance' => $bal($p['balance'])]);
  case 'balance':      $answer(0, 'OK', ['balance' => $bal($p['balance'])]);
  case 'status':
    $q = $pdo->prepare('SELECT type FROM casino_tx WHERE trans_guid=?'); $q->execute([$d['trans_guid'] ?? '']);
    $t = $q->fetchColumn();
    $answer(0, 'OK', ['trans_status' => $t ? 'OK' : 'NOT_FOUND']);
  case 'bet': case 'win': case 'cancel':
    $pdo->beginTransaction();
    $p = $player($d['account'], true);                                  // row lock: no double spend
    $q = $pdo->prepare('SELECT balance_after FROM casino_tx WHERE trans_guid=?'); $q->execute([$d['trans_guid']]);
    if (($done = $q->fetchColumn()) !== false) { $pdo->commit(); $answer(0, 'OK', ['balance' => $bal($done)]); } // repeat → same answer
    $amount = (float)($d['amount'] ?? 0);
    if ($cmd === 'bet') {
      if ($amount < 0 || $p['balance'] < $amount) { $pdo->rollBack(); $answer(2006, 'INSUFFICIENT_FUNDS', ['balance' => $bal($p['balance'])]); }
      $delta = -$amount;
    } elseif ($cmd === 'win') { $delta = max(0, $amount); }
    else { // cancel: refund the bet cancel_trans_guid once; unknown bet → OK, nothing refunded
      $ref = $d['cancel_trans_guid'] ?? '';
      $q = $pdo->prepare("SELECT amount FROM casino_tx WHERE trans_guid=? AND type='bet'"); $q->execute([$ref]);
      $c = $pdo->prepare("SELECT 1 FROM casino_tx WHERE ref_guid=? AND type='cancel'"); $c->execute([$ref]);
      $delta = $c->fetchColumn() ? 0 : (float)($q->fetchColumn() ?: 0);   // already refunded → 0
    }
    $nb = $p['balance'] + $delta;
    $pdo->prepare('UPDATE players SET balance=? WHERE user_code=?')->execute([$nb, $p['user_code']]);
    $pdo->prepare('INSERT INTO casino_tx(trans_guid,type,user_code,amount,ref_guid,balance_after) VALUES(?,?,?,?,?,?)')
        ->execute([$d['trans_guid'], $cmd, $p['user_code'], abs($delta), $d['cancel_trans_guid'] ?? null, $nb]);
    $pdo->commit();
    $answer(0, 'OK', ['balance' => $bal($nb)]);
  default: $answer(1012, 'PARAMETERS_INVALID');
}
```

Table `casino_tx` needs a **UNIQUE** index on `trans_guid`: even two simultaneous copies of the same transaction can then be booked only once.

Node.js (Express)

```
// npm i express — wallet.* are your own database functions (one DB transaction each)
import express from 'express'; import { timingSafeEqual, createHash } from 'node:crypto';
const TOKEN = process.env.CASINO_CALLBACK_TOKEN;
const same = (a, b) => timingSafeEqual(createHash('sha256').update(String(a)).digest(),
createHash('sha256').update(String(b)).digest());
const app = express(); app.use(express.json());
app.post('/casino/callback', async (req, res) => {
  const send = (result, status, data = null) => res.json({ result, status, data });
  if (!same(req.get('callback-token') ?? '', TOKEN)) return send(1009, 'TOKEN_INVALID');
  const { command, data: d = {} } = req.body ?? {};
  const p = await wallet.player(d.account);
  if (!p) return send(2002, 'USER_NOT_FOUND');
  switch (command) {
    case 'authenticate': return send(0, 'OK', { account: d.account, balance: p.balance });
    case 'balance':      return send(0, 'OK', { balance: p.balance });
    case 'status': { const t = await wallet.findTx(d.trans_guid); return send(0, 'OK', { trans_status: t ? 'OK' : 'NOT_FOUND'
}); }
    case 'bet': case 'win': case 'cancel': {
      // applyOnce: inside ONE transaction — if trans_guid exists return its balance_after, else apply + store.
      // bet: debit (insufficient → { error: 'INSUFFICIENT_FUNDS' }) · win: credit · cancel: refund bet d.cancel_trans_guid
once (unknown → 0)
      const r = await wallet.applyOnce(command, d);
      return r.error ? send(2006, r.error, { balance: r.balance }) : send(0, 'OK', { balance: r.balance });
    }
    default: return send(1012, 'PARAMETERS_INVALID');
  }
});
app.listen(3000);
```

10. Test before go-live

1. Tell your account manager your callback URL. They press **Callback test**: ALB365 calls your URL with your token and tells you the result (reachable, token accepted, answer format, response time).
2. Create a test player with `/v4/user/create`, give it a test balance in your database, open a cheap slot with `/v4/game/game-url`.
3. Play a few rounds: check that every `bet` / `win` moved the balance exactly once.
4. Send the same `bet` body twice to your own URL (e.g. with curl): the balance must change only once.

```
curl -X POST https://your-site.com/casino/callback -H "Content-Type: application/json" -H "callback-token: YOUR_TOKEN" \
  -d '{"command":"balance","data":{"account":100245}}'
# → {"result":0,"status":"OK","data":{"balance":50}}
```

11. Go-live checklist

- Server IPs sent to ALB365 and whitelisted.
- Callback URL is HTTPS, publicly reachable, answers `bet` / `balance` in well under 1.5 s (aim for < 300 ms).
- `callback-token` verified on every request (constant-time compare).
- Idempotency on `trans_guid` implemented and tested (send the same bet twice).
- `cancel` refunds only once; `win` without a prior bet is accepted (free spins).
- Wallet operations are atomic (database transaction), balances never go negative.