MEGANODES

# CASINO API INTEGRATION GUIDE

Professional Technical Documentation

| | |
|---|---|
| **Document Type** | Partner Integration Manual |
| **Audience** | Technical Partners, Developers, Project Managers |
| **Reference** | https://api-prd-v2.meganodes.net/swagger/index.html |

# Contents

1. Introduction ..... 3
2. API Overview ..... 3
3. Base URL / Swagger Reference ..... 3
4. Authentication / Access Requirements ..... 3
5. Main API Endpoints Detail Guide ..... 3
5.1 Agent Information ..... 4
5.2 Change Agent RTP ..... 4
5.3 Create User ..... 5
5.4 Get User Information ..... 6
5.5 Deposit User Wallet ..... 7
5.6 Withdraw User Wallet ..... 7
5.7 Withdraw All User Wallet ..... 8
5.8 Get Providers ..... 9
5.9 Get Games ..... 10
5.10 Get Game Launch URL ..... 11
5.11 Start Bonus Call ..... 12
5.12 Stop Bonus Call ..... 13
5.13 Get Round Detail Link ..... 14
5.14 Get User Transaction List ..... 15
6. Request / Response Format ..... 17
7. Error Handling ..... 17
8. Callback Integration ..... 18
9. Callback Request and Response Examples ..... 18
9.1 GetBalance Callback ..... 18
9.2 UTransaction Callback ..... 19

# 1. Introduction

Welcome, and thank you for choosing MegaNodes. This guide is prepared to help your team complete integration smoothly and confidently. The document explains each API in practical language, while still maintaining technical accuracy for implementation.

# 2. API Overview

MegaNodes provides a RESTful JSON API for casino platform integration. The API covers agent operations, user lifecycle management, transfer wallet operations, game catalog queries, game launch, bonus-call controls, and transaction history retrieval.

1) Authenticate using the issued API token and approved source IP.

2) Create or sync user accounts in your platform.

3) Manage wallet balances with deposit and withdraw APIs.

4) Retrieve provider and game lists for your lobby.

5) Generate launch URLs and handle session flows.

6) Integrate callback endpoints for real-time wallet processing.

# 3. Base URL / Swagger Reference

Base URL: https://api-prd-v2.meganodes.net

Swagger UI: https://api-prd-v2.meganodes.net/swagger/index.html

# 4. Authentication / Access Requirements

Every request must include authorization and JSON headers. Using a secure token management process is strongly recommended for production operations.

| Header | Value | Example |
|---|---|---|
| Authorization | Bearer {API_TOKEN} | Authorization: Bearer ********-****-****-****-************ |
| Accept | application/json | Accept: application/json |
| Content-Type | application/json | Content-Type: application/json |

# 5. Main API Endpoints Detail Guide

This section provides a detailed integration reference for every main API endpoint, including request format, response format, and practical JSON examples.

## 5.1 Agent Information

**Endpoint: POST /agent/info**
Purpose: Retrieve the information of the agent (name, currency, point, RTP).
Use this API at login or monitoring intervals to validate agent status and available balance.

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code (0 indicates success) |
| message | string | Response message such as OK |
| timestamp | integer | Response timestamp |
| data.name | string | Agent name |
| data.currency | string | Agent currency |
| data.point | number | Current agent point balance |
| data.win_ratio | number | Current RTP setting |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "name": "test",
    "currency": "USD",
    "point": 50000.0,
    "win_ratio": 95.0
  }
}
```

## 5.2 Change Agent RTP

**Endpoint: POST /agent/change_rtp**
Purpose: Change RTP (Return to Player) for the agent.
RTP can only be set within the allowed range. Apply changes through controlled operational workflows.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| win_ratio | number | Yes | Target RTP value (allowed range: 75 to 95) |

Request Example

```
{
  "win_ratio": 85
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083
}
```

## 5.3 Create User

**Endpoint: POST /user/create**
Purpose: Create a new user profile in MegaNodes.
A consistent user provisioning flow helps avoid duplicate users and reconciliation issues.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| name | string | Yes | User name (2 to 50 chars, alphanumeric with _ or .) |

Request Example

```
{
  "name": "test"
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.user_code | integer | Generated user code |
| data.name | string | User name |
| data.point | number | Current user point |

Response Example

```
{
```

```
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "user_code": 30001,
    "name": "test",
    "point": 26500.0
  }
}
```

## 5.4 Get User Information

**Endpoint: POST /user/info**
Purpose: Retrieve profile and balance information for a specific user.
Use this endpoint as a trusted source when reconciling local wallet state with platform data.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | Unique user code |

Request Example

```
{
  "user_code": 30001
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.user_code | integer | User code |
| data.name | string | User name |
| data.point | number | User point balance |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "user_code": 30001,
    "name": "test",
    "point": 26500.0
  }
}
```

## 5.5 Deposit User Wallet

**Endpoint: POST /wallet/deposit**
(Transfer Wallet Only)
Purpose: Add points to user wallet (amount is deducted from agent wallet).
Use idempotency and transaction logs to ensure financial integrity for all wallet operations.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | Target user code |
| amount | number | Yes | Deposit amount |

Request Example

```
{
  "user_code": 30001,
  "amount": 50000.0
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.user_code | integer | User code |
| data.point | number | Updated user point |
| data.pre_point | number | User point before processing |
| data.trans_idx | integer | Transaction ID |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "user_code": 30001,
    "point": 76500.0,
    "pre_point": 26500.0,
    "trans_idx": 17605946189
  }
}
```

## 5.6 Withdraw User Wallet

**Endpoint: POST /wallet/withdraw**

(Transfer Wallet Only)
Purpose: Deduct points from user wallet (amount is returned to agent wallet).
Apply strict validation to prevent negative balances and to keep accounting records consistent.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | Target user code |
| amount | number | Yes | Withdraw amount |

Request Example

```
{
  "user_code": 30001,
  "amount": 10000.0
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.user_code | integer | User code |
| data.point | number | Updated user point |
| data.pre_point | number | User point before processing |
| data.trans_idx | integer | Transaction ID |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "user_code": 30001,
    "point": 16500.0,
    "pre_point": 26500.0,
    "trans_idx": 17605946290
  }
}
```

## 5.7 Withdraw All User Wallet

**Endpoint: POST /wallet/withdraw-all**
(Transfer Wallet Only)
Purpose: Withdraw all points from the user wallet.
This endpoint is useful for session closure and settlement flows that require a full transfer.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | Target user code |

Request Example

```
{
  "user_code": 30001
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.user_code | integer | User code |
| data.point | number | Updated user point after full withdrawal |
| data.pre_point | number | User point before processing |
| data.trans_idx | integer | Transaction ID |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "user_code": 30001,
    "point": 0.0,
    "pre_point": 26500.0,
    "trans_idx": 17605946399
  }
}
```

## 5.8 Get Providers

**Endpoint: POST /game/providers**
Purpose: Retrieve the provider list available to the agent.
Call this endpoint regularly to keep your lobby provider mapping current.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| lang | string | Yes | Language code (example: en) |

Request Example

```
{
  "lang": "en"
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data[].provider_code | integer | Provider code |
| data[].provider_name | string | Provider display name |
| data[].localized_title | string | Localized provider title |
| data[].status | string | Provider status (Normal or Maintenance) |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": [
    {
      "provider_code": 1,
      "provider_name": "Pragmatic Play",
      "localized_title": "Pragmatic Play",
      "status": "Normal"
    }
  ]
}
```

## 5.9 Get Games

**Endpoint: POST /game/games**
Purpose: Retrieve a paginated game list for a provider and language.
Use this endpoint to build lobby pages and synchronize catalog updates.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| provider_code | integer | Yes | Provider code |
| category | string | No | Game category (Slot, Live, Lobby) |
| lang | string | Yes | Language code |
| offset | integer | No | Start index (0-based) |
| limit | integer | Yes | Page size (10 to 1000) |

Request Example

```
{
  "provider_code": 1,
  "category": "Slot",
  "lang": "en",
  "offset": 0,
  "limit": 100
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.total | integer | Total game count |
| data.list[].provider_code | integer | Provider code |
| data.list[].category | string | Game category |
| data.list[].game_symbol | string | Unique game symbol |
| data.list[].game_name | string | Game name |
| data.list[].localized_title | string | Localized title |
| data.list[].game_image | string | Game image URL |
| data.list[].launch_enable | boolean | Whether game launch is enabled |
| data.list[].under_maintenance | boolean | Whether game is under maintenance |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "total": 1,
    "list": [
      {
        "provider_code": 1,
        "category": "Slot",
        "game_symbol": "vswaysdogs",
        "game_name": "The Dog House Megaways",
        "localized_title": "The Dog House Megaways",
        "game_image": "https://.../game_pic/vswaysdogs.jpg",
        "launch_enable": true,
        "under_maintenance": false
      }
    ]
  }
}
```

## 5.10 Get Game Launch URL

**Endpoint: POST /game/game-url**

Purpose: Generate a launch URL for a user and game.
This endpoint should be called after validating wallet status and user session state.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | User code |
| provider_code | integer | Yes | Provider code |
| game_symbol | string | Yes | Game symbol |
| lang | string | Yes | Language code |
| return_url | string | Yes | Lobby return URL |
| win_ratio | number | Yes | Optional RTP override (75 to 95) |

Request Example

```
{
  "user_code": 30001,
  "provider_code": 1,
  "game_symbol": "vs20doghouse",
  "lang": "en",
  "return_url": "https://lobby.best-casino.com",
  "win_ratio": 95
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.launch_url | string | Game launch URL |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "launch_url": "https://..."
  }
}
```

## 5.11 Start Bonus Call

**Endpoint: POST /game/start-call**
Purpose: Start a bonus-call flow for an in-progress game.
Use this API only for call-enabled games and record call IDs for later traceability.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | User code |
| provider_code | integer | Yes | Provider code |
| game_symbol | string | Yes | Game symbol |
| call_amount | number | Yes | Total call amount |
| meta | object | No | Optional metadata for call context |

Request Example

```
{
  "user_code": 30001,
  "provider_code": 1,
  "game_symbol": "vs20doghouse",
  "call_amount": 10000.0,
  "meta": {}
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.call_id | integer | Created call ID |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "call_id": 30001
  }
}
```

## 5.12 Stop Bonus Call

**Endpoint: POST /game/stop-call**
Purpose: Stop an existing bonus-call by call ID.
Graceful stop handling helps avoid pending states in operational dashboards.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| call_id | string | Yes | Bonus-call ID |

Request Example

```
{
  "call_id": "30001"
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data | object | Response data object |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {}
}
```

## 5.13 Get Round Detail Link

**Endpoint: POST /game/round-details**
Purpose: Retrieve a URL for round history details.
This endpoint is useful for customer support and dispute investigation workflows.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | User code |
| provider_code | integer | Yes | Provider code |
| game_symbol | string | Yes | Game symbol |
| lang | string | Yes | Language code |
| round_id | string | No | Round identifier |

Request Example

```
{
  "user_code": 30001,
  "provider_code": 1,
  "game_symbol": "vs20doghouse",
  "lang": "en",
  "round_id": "12345713400010008"
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |
| data.url | string | Round detail URL |

Response Example

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {
    "url": "https://..."
  }
}
```

## 5.14 Get User Transaction List

**Endpoint: POST /game/transaction**
Purpose: Retrieve transaction history for a user in a time window.
Use paginated pulls and persistent checkpoints for efficient reconciliation jobs.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | User code |
| start_time | string | Yes | Start time |
| end_time | string | Yes | End time |
| offset | integer | Yes | Offset |
| limit | integer | Yes | Limit |

Request Example

```
{
  "user_code": 30001,
  "start_time": "2026-01-01 11:00:00",
  "end_time": "2026-01-01 12:00:00",
  "offset": 0,
  "limit": 1000
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| code | integer | Response code |
| message | string | Response message |
| timestamp | integer | Response timestamp |

| | | |
|---|---|---|
| data[].idx | integer | Transaction row ID |
| data[].round_id | string | Round ID |
| data[].provider_name | string | Provider name |
| data[].symbol | string | Game symbol |
| data[].gametype | string | Game type |
| data[].type | integer | Transaction type enum |
| data[].amount | number | Transaction amount |
| data[].prebalance | number | Balance before transaction |
| data[].balance | number | Balance after transaction |
| data[].iscall | boolean | Bonus-call indicator |
| data[].round_finished | boolean | Round completion flag |

Response Example


```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": [
    {
      "idx": 17605946189,
      "user_idx": 30001,
      "round_id": "123456789",
      "provider_idx": 1,
      "provider_name": "Pragmatic Play",
      "symbol": "vswaysdogs",
      "game_name": "The Dog House Megaways",
      "category": 1,
      "gametype": "Slot",
      "type": 4,
      "regdate": "2026-01-01T11:30:00Z",
      "details": "",
      "amount": 1000.0,
      "prebalance": 11000.0,
      "balance": 10000.0,
      "iscall": false,
      "round_finished": true
    }
  ]
}
```

# 6. Request / Response Format

All API requests use JSON body with HTTP POST.

Most responses follow the standard envelope shown below.

```
{
  "code": 0,
  "message": "OK",
  "timestamp": 1744616083,
  "data": {}
}
```

# 7. Error Handling

For stable production behavior, validate HTTP status and API response code together.

| Code | Name | Meaning |
|---|---|---|
| 0 | Ok | Successful response |
| 400 | BadRequest | Invalid request payload or format |
| 401 | Unauthorized | Invalid or expired API token |
| 423 | AccountLocked | Agent is locked or pending approval |
| 429 | TooManyRequests | Rate limited |
| 500 | ServiceError | Internal service failure |
| 1005 | IPNotAllowed | Source IP is not allowlisted |
| 1006 | IPTemporarilyBlocked | IP temporarily blocked by security policy |
| 2001 | PointNotEnough | Insufficient user or agent points |
| 2002 | UserNotFound | Target user does not exist |
| 2003 | ProviderNotFound | Target provider does not exist |

# 8. Callback Integration

The callback URL can be set in the Account Settings. All callback requests are delivered using Bearer Authentication with Account callback token.

| Callback URL | Method | Purpose |
|---|---|---|
| /GetBalance | POST | Get User's Current Balance |
| /UTransaction | POST | Handle user transaction requests |

# 9. Callback Request and Response Examples

## 9.1 GetBalance Callback

**Endpoint: POST /GetBalance**
Description: Get User's Current Balance.
Use this callback whenever the platform needs the latest user balance before processing gameplay events.

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| user_code | integer | Yes | Partner user identifier used for wallet lookup |

Request Example

```
{
  "user_code": 1
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| data.user_balance | number | Current balance for the requested user |
| error | integer | Processing result code (0 indicates success) |
| description | string | Response message |

Response Example

```
{
  "data": {
    "user_balance": 10000.0
  },
  "error": 0,
  "description": "OK"
}
```

## 9.2 UTransaction Callback

**Endpoint: POST /UTransaction**
Description: User Transaction requests.
Use this callback to process Bet/Win/Cancel wallet actions. transaction_type values: Bet (1), Win (2), Cancel (3).

### Request Format

| Field | Type | Required | Description |
|---|---|---|---|
| trans_id | integer | Yes | Transaction ID |
| provider_name | string | Yes | Game Provider's Name |
| provider_code | integer | Yes | Provider identifier |
| game_name | string | Yes | Game Name |
| symbol | string | Yes | Game symbol code |
| category | integer | Yes | Game category code |
| gametype | string | Yes | Game type text (example: Slot) |
| round_id | string | Yes | Round identifier for idempotent processing |
| user_code | integer | Yes | Partner user identifier |
| transaction_type | integer | Yes | 1=Bet, 2=Win, 3=Cancel |
| is_call | boolean | Yes | Indicates bonus-call transaction context |
| amount | number | Yes | Transaction amount |
| round_finished | Boolean | Yes | Flag for determine round is finished |

Request Example

```
{
  "trans_id": 1234567890,
  "provider_name": "Pragmatic Play",
  "game_name": "The Dog House Megaways",
  "symbol": "vswaysdogs",
  "provider_code": 1,
  "category": 1,
  "gametype": "Slot",
  "round_id": "123456789",
  "user_code": 1,
  "transaction_type": 1,
  "is_call": false,
  "amount": 1000.0,
  "round_finished": false
}
```

### Response Format

| Field | Type | Description |
|---|---|---|
| data.user_balance | number | Updated user balance after transaction processing |
| error | integer | Processing result code (0 indicates success) |
| description | string | Response message |

Response Example


```
{
  "data": {
    "user_balance": 10000.0
  },
  "error": 0,
  "description": "OK"
}
```