Shankomee Game Provider API Documentation

Table of Contents

Overview

The Shankomee Game Provider API is a comprehensive gaming platform that provides seamless integration for client sites to offer Shan Komee games to their players. The API handles player management, balance operations, game launching, transaction processing, and real-time callbacks.

Key Features

- **Player Management**: Automatic player creation and management

- **Balance Operations**: Real-time balance updates and transfers

- **Game Launching**: Direct game URL generation

- **Transaction Processing**: Complete transaction lifecycle management

- **Real-time Callbacks**: Webhook notifications for game events

- **Report Generation**: Comprehensive transaction reporting

- **Multi-Agent Support**: Support for multiple agents and sub-agents

---

Authentication

API Key Authentication

Most endpoints require authentication using Laravel Sanctum tokens.

```http
Authorization: Bearer {your_token}
```

Transaction Middleware

Some endpoints use a custom transaction middleware with a specific header key:

```http
X-Transaction-Key: yYpfrVcWmkwxWx7um0TErYHj4YcHOOWr
```

Base URL

```
Production: https://your-provider-domain.com/api
Staging: https://staging-provider-domain.com/api
```

API Endpoints

1. Authentication Endpoints

Login

```http
POST /login
```

Request Body:

```json
{
  "user_name": "agent001",
  "password": "password123"
}
```

Response:

```json
{
  "status": "Request was successful.",
  "message": "Login successful",
  "data": {
    "user": {
      "id": 1,
      "user_name": "agent001",
      "name": "Agent Name",
      "type": 20,
      "shan_agent_code": "SCT931"
    },
    "token": "1|abc123def456..."
  }
}
```

Register

```http
POST /register
```

Request Body:

```json
{
  "user_name": "newagent",
  "name": "New Agent",
  "phone": "1234567890",
  "password": "password123",
  "password_confirmation": "password123",
  "shan_agent_code": "NEW001",
  "shan_agent_name": "New Agent Name",
  "shan_secret_key": "secret123",
  "shan_callback_url": "https://client-site.com"
}
```

2. Game Launch Endpoints

Client Launch Game (No Authentication)

```http
POST /client/launch-game
```

Request Body:

```json
{
  "agent_code": "SCT931",
  "product_code": 15,
  "game_type": "shan_komee",
  "member_account": "PLAYER001",
  "balance": 1000.00,
  "nickname": "Player Nickname",
  "callback_url": "https://client-site.com/callback"
}
```

Response:

```json
{
  "code": 200,
  "message": "Game launched successfully",
  "url": "https://shan-ko-mee-mm.vercel.app/?user_name=PLAYER001&balance=1000.00"
}
```

Authenticated Launch Game

```http
POST /seamless/launch-game
Authorization: Bearer {token}
```

Request Body:

```json
{
  "product_code": 15,
  "game_type": "shan_komee",
  "member_account": "PLAYER001",
  "balance": 1000.00,
  "nickname": "Player Nickname"
}
```

Response:

```json
{
  "code": 200,
  "message": "Game launched successfully",
  "url": "https://shan-ko-mee-mm.vercel.app/?user_name=PLAYER001&balance=1000.00"
}
```

3. Balance Management Endpoints

Get Balance

```http
POST /shan/getbalance
```

Request Body:

```json
{
  "batch_requests": [
    {
      "member_account": "PLAYER001",
      "product_code": 15
    },
    {
      "member_account": "PLAYER002",
      "product_code": 15
    }
  ],
  "operator_code": "OP001",
  "currency": "MMK"
}
```

Response:

```json
{
  "status": "Request was successful.",
  "message": "Balance retrieved successfully",
  "data": [
    {
      "member_account": "PLAYER001",
      "product_code": 15,
      "balance": 1000.00,
      "code": 0,
      "message": "Success"
    },
    {
      "member_account": "PLAYER002",
```

      "product_code": 15,
      "balance": 500.00,
      "code": 0,
      "message": "Success"
    }
  ]
}

4. Transaction Processing

Create Transaction
```http
POST /transactions
X-Transaction-Key: yYpfrVcWmkwxWx7um0TErYHj4YcHOOWr
```

Request Body:
```json
{
  "banker": {
    "player_id": "SKP0101"
  },
  "players": [
    {
      "player_id": "PLAYER001",
      "bet_amount": 100.00,
      "win_lose_status": 1,
      "amount_changed": 150.00
    },
    {
      "player_id": "PLAYER002",
      "bet_amount": 50.00,
      "win_lose_status": 0,
      "amount_changed": 50.00
    }
  ]
}
```

Response:

```json
{
  "status": "Request was successful.",
  "message": "Transaction Successful",
  "data": {
    "status": "success",
    "wager_code": "abc123def456",
    "players": [
      {
        "player_id": "PLAYER001",
        "bet_amount": 100.00,
        "win_lose_status": 1,
        "amount_changed": 150.00,
        "current_balance": 1150.00
      },
      {
        "player_id": "PLAYER002",
        "bet_amount": 50.00,
        "win_lose_status": 0,
```

        "amount_changed": 50.00,
        "current_balance": 450.00
      }
    ],
    "banker": {
      "player_id": "SKP0101",
      "balance": 9850.00
    },
    "agent": {
      "player_id": "AGENT001",
      "balance": 10000.00
    }
  }
}

5. Report and Analytics Endpoints

Get Report Transactions (Grouped)
```http
POST /provider/shan/report-transactions
```

Request Body:
```json
{
  "agent_code": "SCT931",
  "date_from": "2025-01-01",
  "date_to": "2025-01-31",
  "member_account": "PLAYER001",
  "group_by": "both"
}
```

Response:

```json
{
  "status": "Request was successful.",
  "message": "Report transactions retrieved successfully",
  "data": {
    "agent_info": {
      "agent_id": 1,
      "agent_code": "SCT931",
      "agent_name": "Agent Name"
    },
    "filters": {
      "date_from": "2025-01-01",
      "date_to": "2025-01-31",
      "member_account": "PLAYER001",
      "group_by": "both"
    },
    "report_data": [
      {
        "agent_id": 1,
        "member_account": "PLAYER001",
        "total_transactions": 25,
        "total_transaction_amount": "1500.00",
        "total_bet_amount": "1000.00",
        "total_valid_amount": "1000.00",
        "avg_before_balance": "500.00",
        "avg_after_balance": "600.00",
        "first_transaction": "2025-01-01 08:00:00",
        "last_transaction": "2025-01-31 23:59:59",
        "agent": {
          "id": 1,
          "user_name": "AGENT001",
          "name": "Agent Name"
        }
      }
    ],
    "summary": {
      "total_groups": 5,
      "total_transactions": 125,
      "total_transaction_amount": "7500.00",
      "total_bet_amount": "5000.00",
      "total_valid_amount": "5000.00",
      "unique_agents": 2,
      "unique_members": 5
    }
  }
}
```

Get Member Transactions

```http
POST /provider/shan/member-transactions
```

Request Body:

```json
```

~~~
{
  "agent_code": "SCT931",
  "member_account": "PLAYER001",
  "date_from": "2025-01-01",
  "date_to": "2025-01-31",
  "limit": 50
}
```

**Response:**
```json
{
  "status": "Request was successful.",
  "message": "Member transactions retrieved successfully",
  "data": {
    "agent_info": {
      "agent_id": 1,
      "agent_code": "SCT931",
      "agent_name": "Agent Name"
    },
    "member_account": "PLAYER001",
    "filters": {
      "date_from": "2025-01-01",
      "date_to": "2025-01-31",
      "limit": 50
    },
    "transactions": [
      {
        "id": 1,
        "user_id": 1,
~~~

```
        "agent_id": 1,
        "member_account": "PLAYER001",
        "transaction_amount": "150.00",
        "bet_amount": "100.00",
        "valid_amount": "100.00",
        "before_balance": "1000.00",
        "after_balance": "1150.00",
        "wager_code": "abc123def456",
        "settled_status": "settled_win",
        "created_at": "2025-01-15 10:30:00",
        "agent": {
          "id": 1,
          "user_name": "AGENT001",
          "name": "Agent Name"
        }
      }
    ],
    "total_found": 25
  }
}
```

6. Agent Management Endpoints

Get Users by Agent

```http
POST /provider/shan/users-by-agent
```

Request Body:

```json
{
  "agent_id": 1
}
```

Response:

```json
{
  "status": "Request was successful.",
  "message": "Users retrieved successfully",
  "data": {
    "agent": {
      "id": 1,
      "user_name": "AGENT001",
      "name": "Agent Name",
      "type": 20
    },
    "users": [
      {
        "id": 2,
        "user_name": "PLAYER001",
        "name": "Player One",
        "phone": "1234567890",
        "email": "player1@example.com",
        "type": 40,
        "status": 1,
        "agent_id": 1,
        "shan_agent_code": "SCT931",
        "created_at": "2025-01-01T00:00:00.000000Z",
        "wallet": {
          "balance": 1000.00
        }
      }
    ],
    "total_users": 1
  }
}
```

Get All Agents

```http
GET /provider/shan/agents
```

Response:

```json
{
  "status": "Request was successful.",
  "message": "Agents retrieved successfully",
  "data": {
    "agents": [
      {
        "id": 1,
        "user_name": "AGENT001",
        "name": "Agent Name",
        "type": 20,
        "status": 1,
        "shan_agent_code": "SCT931",
        "user_count": 5,
        "created_at": "2025-01-01T00:00:00.000000Z"
      }
    ],
    "total_agents": 1
  }
}
```

Error Codes

Standard HTTP Status Codes

- `200` - Success
- `400` - Bad Request
- `401` - Unauthorized
- `404` - Not Found
- `422` - Validation Error
- `500` - Internal Server Error

Custom Error Codes (ShankomeeCode Enum)

- `0` - Success
- `999` - Internal Server Error
- `1000` - Member does not exist
- `1001` - Insufficient balance
- `1002` - Proxy key error
- `1003` - Duplicate transaction
- `1004` - Invalid signature
- `1005` - Game list not found
- `1006` - Bet does not exist

- `2000` - Product under maintenance
- `2001` - Invalid currency
- `2002` - Invalid member account

Error Response Format

```json
{
  "status": "Error has occured...",
  "message": "Error description",
  "data": {
    "field_name": [
      "Validation error message"
    ]
  }
}
```

Postman Collection

Environment Variables

Create a Postman environment with these variables:

```json
{
  "base_url": "https://your-provider-domain.com/api",
  "token": "your_auth_token_here",
  "agent_code": "SCT931",
  "member_account": "PLAYER001"
```

}

```

### Collection Structure

#### 1. Authentication

- **Login**: `POST {{base_url}}/login`

- **Register**: `POST {{base_url}}/register`

#### 2. Game Launch

- **Client Launch Game**: `POST {{base_url}}/client/launch-game`

- **Authenticated Launch Game**: `POST {{base_url}}/seamless/launch-game`

#### 3. Balance Management

- **Get Balance**: `POST {{base_url}}/shan/getbalance`

#### 4. Transactions

- **Create Transaction**: `POST {{base_url}}/transactions`

#### 5. Reports

- **Report Transactions**: `POST {{base_url}}/provider/shan/report-transactions`

- **Member Transactions**: `POST {{base_url}}/provider/shan/member-transactions`

#### 6. Agent Management

- **Users by Agent**: `POST {{base_url}}/provider/shan/users-by-agent`

- **All Agents**: `GET {{base_url}}/provider/shan/agents`

### Sample Postman Request

#### Launch Game Request
```

```http
POST {{base_url}}/client/launch-game
Content-Type: application/json

{
  "agent_code": "{{agent_code}}",
  "product_code": 15,
  "game_type": "shan_komee",
  "member_account": "{{member_account}}",
  "balance": 1000.00,
  "nickname": "Test Player",
  "callback_url": "https://client-site.com/callback"
}
```

Transaction Request

```http
POST {{base_url}}/transactions
Content-Type: application/json
X-Transaction-Key: yYpfrVcWmkwxWx7um0TErYHj4YcHOOWr

{
  "banker": {
    "player_id": "SKP0101"
  },
  "players": [
    {
      "player_id": "{{member_account}}",
      "bet_amount": 100.00,
      "win_lose_status": 1,
      "amount_changed": 150.00
    }
  ]
}
```

Integration Examples

JavaScript Integration

Launch Game

```javascript
async function launchGame(agentCode, memberAccount, balance) {
  try {
    const response = await fetch('/api/client/launch-game', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        agent_code: agentCode,
        product_code: 15,
        game_type: 'shan_komee',
        member_account: memberAccount,
        balance: balance,
        callback_url: 'https://your-site.com/callback'
      })
    });
    const data = await response.json();

    if (data.code === 200) {
      // Redirect to game URL
      window.location.href = data.url;
    } else {
      console.error('Launch failed:', data.message);
    }
  } catch (error) {
    console.error('Error:', error);
  }
}
```

Get Balance

```javascript
async function getBalance(memberAccount) {
  try {
    const response = await fetch('/api/shan/getbalance', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        batch_requests: [
          {
            member_account: memberAccount,
            product_code: 15
          }
    ],
    currency: 'MMK'
   })
  });

  const data = await response.json();
  return data.data[0].balance;
 } catch (error) {
  console.error('Error getting balance:', error);
  return 0;
 }
}
```

PHP Integration

Launch Game
```php
<?php
function launchGame($agentCode, $memberAccount, $balance) {
  $url = 'https://your-provider-domain.com/api/client/launch-game';

  $data = [
    'agent_code' => $agentCode,
    'product_code' => 15,
    'game_type' => 'shan_komee',
    'member_account' => $memberAccount,
    'balance' => $balance,
    'callback_url' => 'https://your-site.com/callback'
  ];
```

```
    $options = [
        'http' => [
            'header' => "Content-type: application/json\r\n",
            'method' => 'POST',
            'content' => json_encode($data)
        ]
    ];

    $context = stream_context_create($options);
    $result = file_get_contents($url, false, $context);

    if ($result === FALSE) {
        return false;
    }

    $response = json_decode($result, true);

    if ($response['code'] === 200) {
        return $response['url'];
    }

    return false;
}
?>
```

Webhook Callbacks

Balance Update Callback

The provider will send balance update callbacks to your specified callback URL.

Callback URL Format

```
https://your-client-site.com/api/shan/client/balance-update
```

Callback Payload

```json
{
  "wager_code": "abc123def456",
  "game_type_id": 15,
  "players": [
    {
      "player_id": "PLAYER001",
      "balance": 1150.00
    },
    {
      "player_id": "SKP0101",
      "balance": 9850.00
    }
  ],
  "banker_balance": 9850.00,
  "agent_balance": 9850.00,
  "timestamp": "2025-01-15T10:30:00Z",
  "total_player_net": 100.00,
  "banker_amount_change": -100.00,
  "signature": "abc123def456..."
```

}

Callback Response

Your callback endpoint should respond with:

```json
{
  "status": "success",
  "code": "SUCCESS",
  "message": "Balances updated successfully."
}
```

Security

API Key Security

- Store API keys securely
- Use HTTPS for all API communications
- Implement rate limiting on your side
- Validate all incoming data

Signature Verification

For webhook callbacks, verify the signature:

```php
function verifySignature($payload, $signature, $secretKey) {
    ksort($payload);
    $expectedSignature = hash_hmac('md5', json_encode($payload), $secretKey);
    return hash_equals($expectedSignature, $signature);
}
```

Transaction Security

- Always use the `X-Transaction-Key` header for transaction endpoints
- Implement idempotency checks for duplicate transactions
- Validate all transaction data before processing

Support

Technical Support

- **Email**: support@shankomee-provider.com
- **Documentation**: https://docs.shankomee-provider.com
- **Status Page**: https://status.shankomee-provider.com

Integration Support

- **Developer Portal**: https://developers.shankomee-provider.com
- **SDK Downloads**: Available for JavaScript, PHP, Python
- **Test Environment**: https://staging.shankomee-provider.com

SLA

- **Uptime**: 99.9% availability
- **Response Time**: < 200ms for most endpoints
- **Support Response**: < 4 hours during business hours

Changelog

Version 2.0.3 (Current)

- Added comprehensive reporting endpoints

- Enhanced agent management

- Improved error handling

- Added webhook signature verification

Version 2.0.2

- Added authenticated launch game endpoint

- Enhanced balance management

- Improved transaction processing

Version 2.0.1

- Initial release

- Basic game launching

- Balance operations

- Transaction processing

This documentation is updated regularly. Please check for the latest version at our developer portal.