

Wise Gaming Platform Technical Documentation

Executive Summary

This document provides a high-level technical overview of the multi-tenant gaming platform. The platform is built on AWS cloud infrastructure and provides a comprehensive solution for managing gaming operations, including user management, authentication, payments, bonuses, game integrations, and administrative operations.

Table of Contents

1. [Platform Architecture](#)
 2. [Technologies Used](#)
 3. [Deployment Architecture](#)
 4. [Infrastructure Components](#)
 5. [Core Services](#)
 6. [Multi-Tenancy Architecture](#)
 7. [Marsbet Tenant Information](#)
 8. [Access Links](#)
 9. [Key Features](#)
-

Platform Architecture

Overall Design

The platform follows a **serverless, microservices architecture** built on AWS. It uses:

- **AWS Lambda** for compute
- **AWS API Gateway** for API management
- **AWS CloudFront** as CDN and proxy
- **DynamoDB** for NoSQL data storage
- **Aurora PostgreSQL** for relational data
- **AWS Cognito** for authentication

Architecture Pattern

The platform implements a **multi-tenant architecture** where each tenant (brand) operates independently with:

- Separate user pools
 - Isolated data storage
 - Dedicated CloudFront distributions
 - Independent configuration
-

Technologies Used

Backend Technologies

1. **Runtime Environment:** Node.js (versions 12.x, 14.x, 22.x)
2. **Serverless Framework:** AWS SAM (Serverless Application Model)
3. **API Gateway:** AWS API Gateway (REST and HTTP APIs)
4. **Compute:** AWS Lambda Functions

Frontend Technologies

1. **UI Framework:** React.js
2. **Build Tool:** Webpack with custom config-overrides
3. **Styling:** Less (CSS preprocessor)
4. **Package Manager:** Yarn/npm
5. **Deployment:** S3 static website hosting

Database Technologies

1. **NoSQL:** AWS DynamoDB (Pay-per-request model)
2. **Relational:** AWS Aurora PostgreSQL (Serverless mode)
3. **Search:** AWS OpenSearch (optional)

Storage Technologies

1. **Static Assets:** AWS S3
2. **CDN:** AWS CloudFront
3. **Content Management:** S3 with CloudFront OAC (Origin Access Control)

AWS Services

- **VPC:** Virtual Private Cloud for network isolation
- **CloudFront:** Content delivery and request routing
- **Lambda:** Serverless functions
- **API Gateway:** API management
- **Cognito:** User authentication and management
- **DynamoDB:** NoSQL database
- **RDS Aurora:** Managed PostgreSQL database
- **S3:** Object storage
- **SES:** Email service
- **SSM Parameter Store:** Configuration management
- **CloudFormation:** Infrastructure as Code

Deployment Architecture

Environments

The platform supports three deployment environments:

1. **Development (dev):** Development and testing
2. **Integration (int):** Pre-production integration environment
3. **Production (prod):** Live production environment

Deployment Locations

The platform is deployed on **AWS** in the **EU-Central-1 (Frankfurt)** region for compliance and latency optimization.

Infrastructure Deployment

Infrastructure components are deployed using **AWS CloudFormation** with the following hierarchy:

```
Infrastructure Layer (VPC, S3, DynamoDB, Aurora, CloudFront)
  ↓
Service Layer (Lambda APIs for each microservice)
  ↓
Tenant Layer (Per-tenant CloudFormation stack with Cognito, S3,
CloudFront)
```

Infrastructure Components

Core Infrastructure

1. **VPC:** Custom VPC with public and private subnets across multiple availability zones
2. **Database Tier:**
 - DynamoDB tables for session management, user data, gaming data
 - Aurora PostgreSQL for complex queries and analytics
3. **Storage:**
 - S3 buckets for static content and UI deployments
 - CloudFront distributions for global CDN
4. **Networking:** NAT Gateway, Internet Gateway, Route Tables

Key CloudFormation Stacks

- **S3:** Storage buckets and policies
- **VPC:** Network infrastructure
- **DynamoDB:** All NoSQL tables
- **OpenSearch:** Search capabilities
- **DatabaseManagement:** Aurora PostgreSQL setup
- **CloudFront:** CDN configurations
- **Individual Service Stacks:** Each microservice

Core Services

Authentication & User Management

- **AuthManagement:** User authentication via AWS Cognito
- **UserManagement:** User profile management
- **AccountManagement:** User account operations

Gaming Operations

- **ThirdPartyIntegrationManagement:** Integration with game providers (Pragmatic Play, Evolution, etc.)
- **GameManagement:** Game configuration and management
- **ContentManagement:** Game content and asset management

Financial Operations

- **BalanceManagement:** Wallet and balance operations
- **TransactionManagement:** Transaction tracking and reporting
- **PaymentsIntegrationManagement:** Payment provider integrations
- **BonusManagement:** Bonus and promotional campaigns

Operational Services

- **WageringManagement:** Wagering requirement tracking
- **DocumentManagement:** KYC and document management
- **NotificationManagement:** User notifications
- **RiskManagement:** Risk assessment and control
- **RiskGroupManagement:** Risk group configurations

Administrative Tools

- **TenantCockpit:** Multi-tenant administration
- **AppWebSocketManagement:** Real-time app connections
- **CRMWebSocketManagement:** Real-time CRM connections
- **RecommendationManagement:** Game and content recommendations

Supporting Services

- **Tracker:** Analytics and tracking
- **ContentManagement:** Content delivery
- **AffiliateManagement:** Affiliate program management

Multi-Tenancy Architecture

Tenant Isolation

Each tenant operates as an independent unit with:

1. **Dedicated AWS Cognito User Pool:** Separate authentication

2. **Dedicated S3 Buckets:** For app and backoffice UIs
3. **Dedicated CloudFront Distribution:** For app and backoffice domains
4. **Shared Database:** Tenant data segmented by **tenant** attribute
5. **Shared API Services:** Services handle multi-tenancy via headers (**X-Atom-Tenant**)

Tenant Configuration

Tenant configuration is managed via:

- **AWS SSM Parameter Store:** Tenant-specific parameters
 - **CloudFormation Stack:** Per-tenant infrastructure
 - **Custom Headers:** Tenant identification in requests
-

Marsbet Tenant Information

Overview

Marsbet is one of the platform's production tenants, operating as a gaming platform with specialized features including geo-location capabilities.

Special Features

Marsbet has **Geo-Location enabled**, which utilizes CloudFront request interceptors to manage geographic restrictions and location-based routing.

Architecture Components

- **User Pool:** Dedicated Cognito user pool
- **Identity Provider:** Custom authentication flows
- **User Groups:** TenantAdmin, StandardUser, SuperAdmin, CRMAdmin, BonusAdmin, CustomerService, FinanceAdmin

Regional Deployments

The platform supports Marsbet in multiple regions:

- Marsbet (main)
 - Marsbet-PT (Portugal)
 - Marsbet-MX (Mexico)
-

Access Links

Backoffice URLs

For Marsbet backoffice access:

Integration Environment:

- Backoffice URL: <https://backoffice.int.wgplatform.com>

- CloudFront Distribution ID: Configurable via AWS console

Production Environment:

- Backoffice URL: <https://backoffice.wgplatform.com>
- CloudFront Distribution ID: Configurable via AWS console

App URLs

Integration Environment:

- App URL: <https://int.marsbet.com>

Production Environment:

- App URL: <https://marsbet.com>

Backoffice Features

The backoffice provides access to:

- User management and customer service
- Bonus and campaign management
- Transaction and wallet management
- Risk and limits management
- Content and game configuration
- Payment method configuration
- Document verification
- Analytics and reporting

Affiliate Backoffice

The platform also includes an affiliate backoffice system for partner management at:

- Affiliate Backoffice UI (React application)
- Manages affiliate campaigns, landing pages, and performance

Key Features

Security Features

1. **AWS Cognito Integration:** Enterprise-grade authentication
2. **Multi-Factor Authentication (MFA):** Optional MFA support
3. **Advanced Security Mode:** Audit mode enabled
4. **TLS Encryption:** All communications over HTTPS (TLS 1.2+)
5. **Session Management:** Secure session handling with DynamoDB
6. **Access Control:** Role-based access control (RBAC)

Scalability Features

1. **Serverless Architecture:** Automatic scaling
2. **Aurora Serverless:** Auto-scaling database
3. **DynamoDB On-Demand:** Pay-per-request pricing
4. **CloudFront Global CDN:** Low latency worldwide
5. **Lambda Concurrency:** Automatic request handling

Multi-Tenancy Features

1. **Tenant Isolation:** Complete data separation
2. **Configurable Features:** Per-tenant customization
3. **Geo-Location Support:** Location-based features
4. **Independent Scaling:** Per-tenant resource allocation

Integration Features

1. **Payment Providers:** Multiple payment integrations
2. **Game Providers:** Integration with major gaming content providers
3. **Webhook Support:** Callback management
4. **API-First Design:** RESTful API architecture
5. **OpenAPI Specifications:** API documentation

Operational Features

1. **Real-Time Monitoring:** CloudWatch integration
2. **Logging:** Comprehensive logging via CloudWatch Logs
3. **Error Tracking:** DynamoDB Stream error handling
4. **Backup & Recovery:** Automated database backups
5. **Infrastructure as Code:** CloudFormation templates

Deployment Process

Service Deployment

1. **Source Control:** GitLab with CI/CD pipelines
2. **Build Process:** Automated builds via GitLab CI
3. **Deployment:** AWS SAM deploy for Lambda functions
4. **Infrastructure:** CloudFormation for all resources

CI/CD Pipeline

- **Build Stage:** Compile and package applications
- **Deploy Stage:** Deploy to integration or production
- **Environment Separation:** Dedicated environments per tenant
- **Automated Testing:** Unit and integration tests

Database Architecture

DynamoDB Tables

The platform uses numerous DynamoDB tables including:

- Users, Tenants, Sessions
- Wallets, Transactions
- Bonuses, BonusClaims, BonusWheels
- Games, GameOrders
- Notifications
- Documents
- Payment methods
- Risk groups and limits
- Affiliate data
- And many more...

Aurora PostgreSQL

Used for:

- User profiles (comprehensive user data)
 - Casino games catalog
 - Sportsbook reports
 - Complex analytics queries
 - Search functionality with full-text search
-

Network Architecture

VPC Configuration

- **CIDR:** 10.0.0.0/16
- **Public Subnets:** 10.0.0.0/24, 10.0.2.0/24
- **Private Subnets:** 10.0.1.0/24, 10.0.3.0/24
- **Multi-AZ:** High availability across availability zones

Security Groups

- Default Lambda Security Group for function isolation
 - VPC-based networking for sensitive operations
 - NAT Gateway for outbound internet access
-

Monitoring & Observability

CloudWatch Integration

- **Log Groups:** Per-service logging
- **Metrics:** Performance and usage metrics
- **Alarms:** Automated alerting

- **Dashboards:** Custom monitoring dashboards

Error Handling

- **DynamoDB Streams:** Real-time data change capture
 - **Stream Errors:** Dedicated table for error tracking
 - **Retry Mechanisms:** Built-in retry logic
-

Documentation References

For detailed documentation on specific services:

- **Bonus Management:** See [bonus-management-summary.md](#)
 - **Service Specifications:** See individual service directories
 - **API Documentation:** OpenAPI specs in service directories
 - **Database Schemas:** See [infrastructure/initDatabase.sh](#)
-

Maintenance & Updates

Regular Maintenance

- Database migrations via [initDatabase.sh](#)
- CloudFormation stack updates
- Dependency updates via package managers
- SSL certificate renewals

Deployment Best Practices

1. Always test in integration environment first
 2. Use GitLab merge requests for code review
 3. Follow semantic versioning
 4. Document breaking changes
 5. Monitor CloudWatch after deployments
-

Important Notes

This documentation provides a high-level overview. For detailed implementation information, refer to:

- Individual service directories for specific functionality
 - CloudFormation templates for infrastructure details
 - API documentation for service endpoints
 - Database schema files for data structures
-

Platform Capabilities Summary

- ✔ Multi-tenant architecture with complete isolation
- ✔ Serverless, auto-scaling infrastructure
- ✔ Support for casino, sportsbook, and live casino games
- ✔ Comprehensive bonus and promotional system
- ✔ Advanced risk management and limits
- ✔ Multiple payment provider integrations
- ✔ Real-time user management and notifications
- ✔ Document verification and KYC support
- ✔ Affiliate program management
- ✔ Geo-location based features
- ✔ WebSocket support for real-time updates
- ✔ Mobile and desktop optimized UIs

Contact

info@wisegaming.com

www.wisegaming.com