slotengine®

Product Overview

SlotEngine® RGS is a complete remote gaming server solution, for development and operation of online slot games. It consists of an API using super-light and lightning-fast framework for game server apps; Core Gaming and Casino Integration services, and an Administrative API and portal for operator or regulator access.

Cloud or Private Infrastructure

SlotEngine® was designed for infrastructure implementation that can scale vertically and horizontally, to accommodate operations of any size, under any legal model. It may be deployed privately, or, for example, under AWS or Google cloud infrastructure.

Open Source

SlotEngine® uses open-source components, allowing for cost efficient operations, and easily manageable, proven source code solutions.

Social or Regulated Games

SlotEngine® may be deployed for social or simulated games anywhere in the world, immediately. For regulated casino gaming, separate approval and certification by the gaming jurisdictions' authorized testing labs will be required. SlotEngine® has been certified under GLI's international GLI-19 standard. Its RNG has also been certified for UK and other European gaming jurisdictions. SlotEngine® is regulator-ready.

Details

Online game software is constructed under a multi-tier client/server architecture. Games consist of client-side software driving the UI and player experience, and **server-side software** driving the game logic, gaming math, outcome calculations, bet handling, financial accounting, player-game-and-casino validations, and transaction tracking. Moreover, the server-side integrates with a casino-platform, connecting the player to his account at the online casino, and to his casino wallet. **SlotEngine® is the server-side solution for online slot games.**

SlotEngine® does not place any limitations on the design of the client-side software. It is client-technology agnostic. The **SlotEngine®API** endpoints are TLS-secured web services, using standard **JSON** payloads in both incoming **HTTPS** requests and outgoing responses. The game-client code may be written in anything. Many online casino operators choose games in **HTML5/JS**, for ease of integration into the casino's own websites and apps.

Each game request/response pair is a highly efficient transaction, running under a choice of Web Servers: SlotEngine® has been tested with both **Apache** and **NGINX** web servers for high transaction throughput, and a ms-measured latency in a round-the-globe test, against Amazon EC2 and RDS services. The web server using FastCGI runs each PHP-FPM transaction in its own process space, in a single thread. The thread terminates with a resulting response, and memory is freed. Each transaction starts from a fresh, blank-slate state. Clean state and a single-thread / single-purpose design, makes for a highly efficient, uncluttered operational model, easy to debug, maintain, and scale.

Under SlotEngine®, game server software must be written using **PHP7**, using an object-oriented programming style, relying on the **SlotEngine® PHP framework**. Using a template solution, a Game server class will inherit from the relevant class of the SlotEngine Game Framework, and implement own custom logic by overriding default methods. Additional PHP7 classes are defined to describe the game's custom parameters such as PAR details, and any custom bonus, game or jackpot details. Most slot game concepts are already handled; the developer's role is that of an integrator, bringing components together, and only specifying their uniqueness, where applicable.

SlotEngine already handles core concepts such as **line games, ways games, scatters, wilds, freespins, standalone jackpots**, and similar. Sample games include a *lines* game with a second-screen bonus, as well as a *ways* game with free spins. However, the framework is completely open, and as game designers invent new features, anyone may implement new classes or functions to extend the framework to suit their needs.

SlotEngine® CIM (casino integration modules) include Player and Wallet concept integration. These are highly efficient PHP7 classes that are customized (once) per operator installation, to link the player's online casino account and his wallet to the games under SlotEngine®. Player information allows the game access to account status, player details, limits, and similar information. Wallet information allows the game access to player's online casino wallet, providing the ability for seamless updates per bet and win (**"seamless wallet"**).

Support for integration with legacy casino platforms also includes an API for **"transfer wallet"** concept, where game funding is transferred into the rgs transfer-wallet prior to game play, then game play occurs against this transfer-wallet, and finally remaining funds are transferred out to the calling platform once player completes their activities.

SlotEngine®, through its Core Gaming services tier, implements a **Comp** concept, allowing operators to comp (award free play) individual players for a specific game. An operator's scheme for awarding the comp is arbitrary -- SlotEngine® provides **an operator API** to allow comps to be granted, and managed. Any SlotEngine® game, when it detects a comp balance, will deplete the comp before using any real account balance. Comp reports are available from the Administrator's portal.

SlotEngine® deployment architecture spans multiple tiers, allowing it to scale horizontally and vertically:

At the top level, is the **Web Server Tier** which contains the primary API web server (such as Apache or NGINX, configured with FastCGI and an optimized PHP-FPM processor/interpreter), the game-servers (code) library and the **SlotEngine® Game Framework**. This tier will make requests to Cache tier and to the Core Gaming Tier. The default installation uses **AWS EC2** virtual servers. This tier may be scaled and load-balanced as required.

The Cache tier is a **Redis** installation which may be configured as a cluster, across separate virtual or physical server instances, but by default is installed as a single instance on the Web Server Tier (AWS EC2). Redis layer is used for caching all game session information. After the cache expires, information is re-read from persistent storage, for next cache session. This layer exists in order to make transactions more efficient. This tier may be scaled as required, including clustering across multiple instances.

The next tier includes the **Core Gaming and CIM**. While CIM is custom per installation, dealing with the player and wallet information, the Core gaming classes implement core services such as random number generation and game history tracking.

The final tier is the **Database tier**. The Core Tier makes SQL requests to MYSQL or MYSQL-compatible engine. Default installation uses **MySQL 8.0** installed under **AWS RDS** managed server instance. This tier may also be scaled as required, and the database technology may be replaced by a compatible engine of choice, even configured in a cluster configuration across multiple physical or virtual servers. The database is used to archive all session history, for operator and regulator purposes. **The primary RGS Database is replicated** to an external server instance, for back-office queries. The Administration Portal and API uses the Replica database structure, in a read-only mode, to provide ad-hoc query and reporting services.

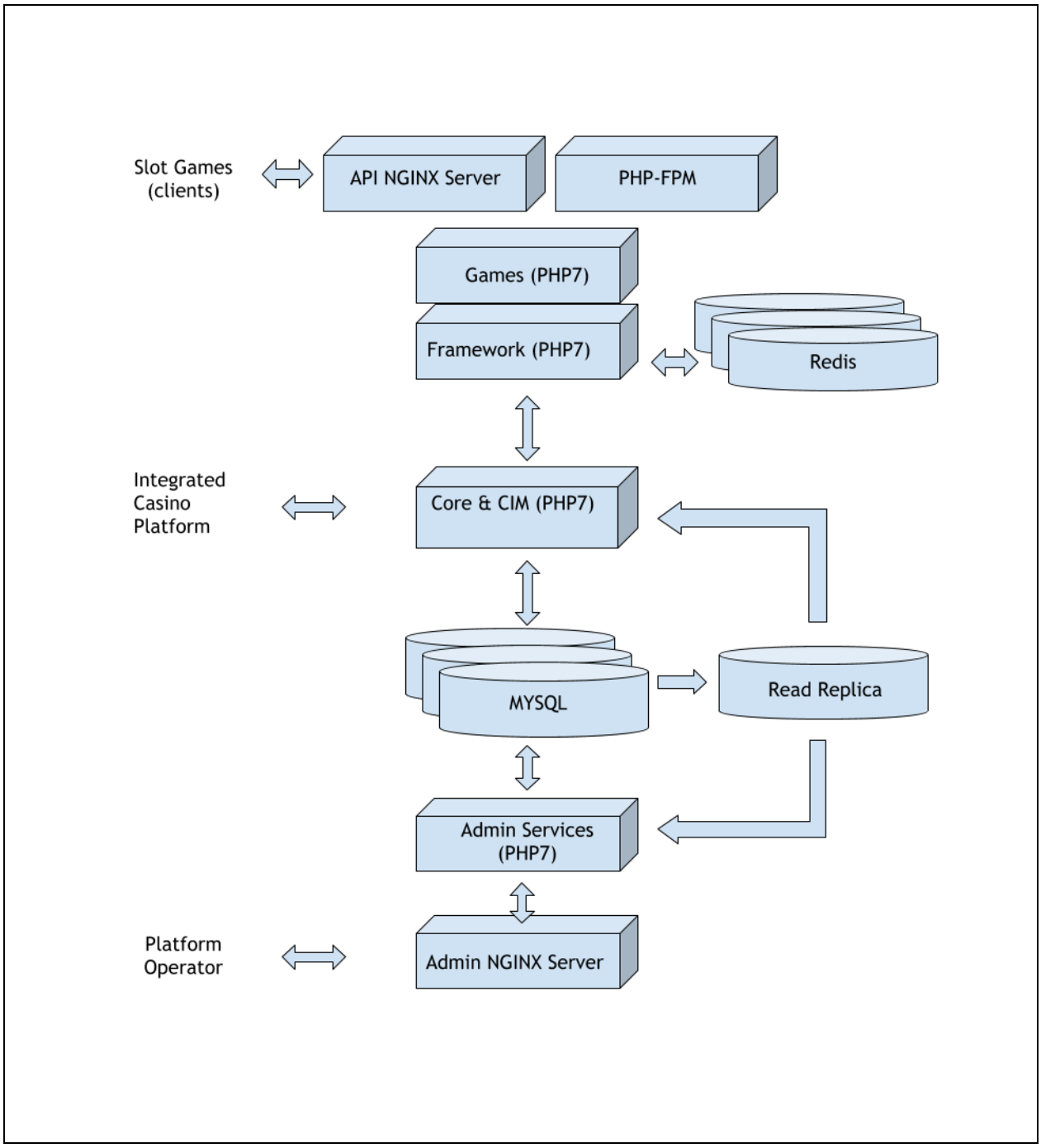
The SlotEngine® Administration Portal provides authorized staff of operators and regulators with management and reporting functionality. Games enrolment, configuration, enable/disable, player/game reset, player/game comps, performance reports, exceptions report and similar functionality is available. These services can run on a separate server instance. The reporting is ran on the replica, while credentials and user administration exists as a separate database.

On installation, SlotEngine® tiers are configured and linked, and the CIM (Casino Integration Modules) are customized to integrate with the online casino. Administrative portal is preconfigured for admin access only. Moreover, an **external CDN (Content Delivery Network) is configured**, for game client code installation, specific to the operator. Default SlotEngine® configuration uses **AWS CloudFront with S3** (Simple Storage Services) for global static content delivery. Finally, the overall integrated solution is tuned and configured redundantly for high availability, backups and the most optimal performance.

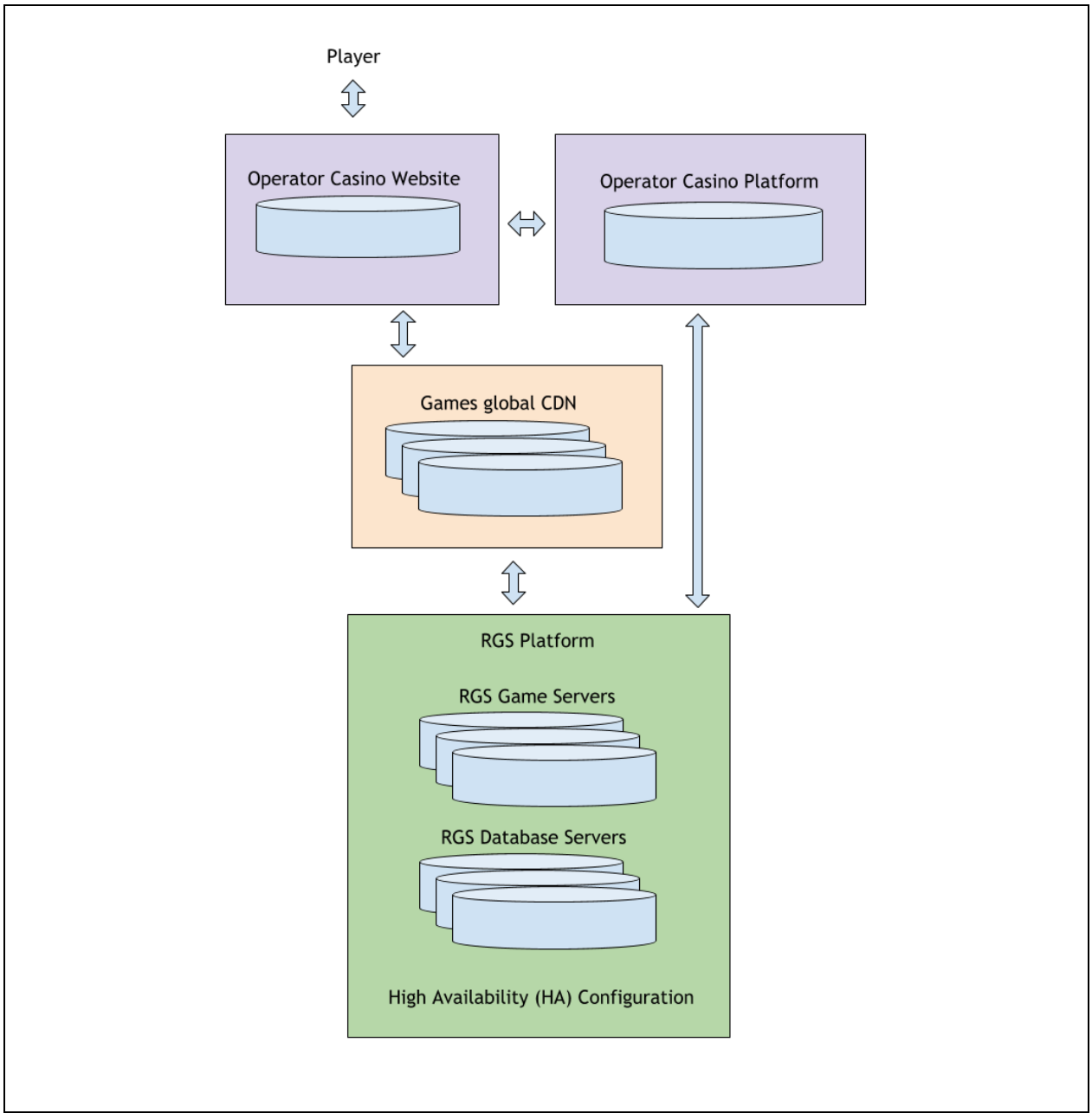
SlotEngine® access is secure. API endpoints are only accessible across HTTPS, from game-client code with issued access code and API key: this information is issued to game studios responsible for publishing the game. After establishing a secure session, all communication with the SlotEngine®game-servers is authenticated using a session token, which remains valid only for a limited time. Moreover, since the game client-side functionality is limited to interpreting game results only, **the external code can not determine the outcome of any game**, therefore, security is also inherent in the design of the total solution. All API communications are encrypted, under the HTTPS protocol and exchange concept.

The Core Gaming and CIM tier is not accessible by public, it is not reachable. It can only be accessed by the internal transactions of the RGS. **The Database tier is not accessible by public,** it is not reachable. It can only be accessed by the Core Gaming Tier, internal processes of the RGS. **The Administrative Portal is accessible only by secure logon, from within preconfigured address range**. **The Administrative API** is only reachable by authorized sessions from the Admin portal, **a vendor specific apikey, and is unreachable by the public.**

SlotEngine® Software Stack



SlotEngine® Deployment



Digital Game
Digital Game Systems Corporation

213 N. Stephanie St. Suite G-297
Henderson, NV, 89074, USA
+1.702.408.8916

support@digitalgamesystems.com