

Golden Casino Operator API

Public operator-facing contract for catalog lookup, game launch, artwork delivery, and callback wallet settlement.

Version	2.0.0
Base URL	https://goldencasino.cloud
Docs URL	https://docs.goldencasino.cloud
Auth Header	x-provider-api-key

Quick Start

- Create an active operator in Provider Admin.
- Assign the operator's live game catalog.
- Create an operator API key.
- Sign each catalog and launch request with the operator API key, timestamp, and a fresh nonce.
- Create launch URLs server-side and open them in the player browser.
- If using callback wallet mode, expose signed balance, debit, credit, cancel, and rollback endpoints.

Operator integration flow

Operators integrate through provider API endpoints only. Runtime routes under `/gs2c` and `/api/runtime` are used by the launched game in the player browser and should not be called server-to-server.

- Use catalog APIs to read the operator-scoped game list.
- Use launch-url to create or resume sessions; expiry rules differ by provider.
- Open launch_url in the player browser on desktop or mobile.
- Use callback wallet endpoints for production balance and settlement.

Provider support and release limits

Lobby visibility is not proof of operator wallet support or game certification. Read the operator-scoped catalog using your own API key; never use the public demo catalog as an operator allowlist.

Field	Description
pragmatic	PP callback-wallet transport is implemented. Only enabled, assigned games are returned. Each game's base game and advertised features still require acceptance testing.

Field	Description
pg	Native PG callback-wallet settlement and durable round/history recovery are implemented behind PG_OPERATOR_SETTLEMENT_ENABLED. The switch is disabled by default. With it off, PG is hidden from operator-scoped catalogs and operator launches return 409. Enable it only after migration and deployed browser acceptance; pushed balance refresh remains PP-only.
playngo	Native callback-wallet transport is opt-in per title through PLAYNGO_<TITLE>_OPERATOR_ENABLED. Enabled, assigned titles use HTTPS operator wallets and signed launches under https://goldencasino.cloud/playngo/<title>/. Flags default to disabled; availability also follows the release registry and operator catalog. Development sandboxes remain separate. Transport support does not replace per-title browser acceptance or authorize live funds.
spribe	Crash is the lobby label, not an API provider ID. Aviator and Chicken have opt-in operator-wallet transport with signed launches at https://goldencasino.cloud/games/aviator/ and https://goldencasino.cloud/games/chicken/. The subdomain demos are temporary client demonstrations, not production operator endpoints. Production enablement and live funds remain subject to release approval and operator wallet acceptance.
operator_integration	Reports mode (operator-wallet, demo-only or sandbox-only), wallet_callback and balance_refresh. Transport support is not RTP certification or production approval.
operator_availability	Reports enabled, assigned, configuration_ready, launch_available and blocker codes. Operator catalogs contain launch-available entries only; admin assignment views retain blocked entries. Without operator scope, assigned and configuration_ready are null. These are transport checks, not live-funds approval.
release_readiness	Reports accepted or blocked for the exact installed version from the M5B registry, with evidence and blocker reasons. Acceptance permits integrated staging and operator assignment; production still requires the remaining platform gates. Operator packages distinguish launchable_games from ready_games.
rtp and features	RTP is stored metadata, not a verified achieved payout rate. Feature flags declare support; they do not prove the necessary replay chains exist.

Use the operator API key in request headers

Use x-provider-api-key with a per-request HMAC-SHA256 signature in production. The signature binds the Unix timestamp, one-time nonce, HTTP method, exact path and query, and SHA-256 of the raw body. DB operator keys automatically resolve operator scope, so operators must not send another operator_id. Provider-admin service keys are separate internal credentials.

- The API-key, Bearer, and x-api-key forms all require the same signature headers in production. Keep keys on your backend.
- Separate canonical fields with actual newline characters, not literal backslash-n text. Hash an empty body for GET requests. Sign the decoded URL path followed by the exact encoded query string in its original order.
- Sign the exact UTF-8 body you send; changing whitespace, field order, path, or query after signing invalidates the signature. Generate a new nonce and signature for every retry; transaction IDs still remain stable.
- Reuse is rejected with 409. A missing, invalid, or stale signature returns 401. Keep your server clock synchronized; the default accepted clock skew is 300 seconds and can be configured by the provider.
- Import the Postman collection, set operatorApiKey as a private local/environment value, and select your symbol and player ID. Its Provider API folder generates timestamp, nonce, and signature automatically using Postman's Web Crypto API. Wallet examples are separate manual callback fixtures.

Preferred auth header

```
x-provider-api-key: ppop_operator-a_REPLACE_ME
x-pp-timestamp: <unix-seconds>
x-pp-nonce: <unique-16-to-128-character-value>
x-pp-signature: <hex-hmac-sha256>
```

Bearer alternative

```
Authorization: Bearer ppop_operator-a_REPLACE_ME
x-pp-timestamp: <unix-seconds>
x-pp-nonce: <unique-16-to-128-character-value>
x-pp-signature: <hex-hmac-sha256>
```

Operator request signature payload

```
HMAC_SHA256(operator_api_key,
  pp-operator-v1\n{timestamp}\n{nonce}\n{METHOD}\n{path-and-query}\n{sha256(raw_body)})
```

Signed requests with Node.js (server-side)

```
import { createHash, createHmac, randomBytes } from "node:crypto";

const baseUrl = "https://goldencasino.cloud";
const apiKey = process.env.OPERATOR_API_KEY;
if (!apiKey) throw new Error("Set OPERATOR_API_KEY on your backend");

async function providerRequest(method, pathAndQuery, payload) {
  const url = new URL(pathAndQuery, baseUrl);
  if (url.origin !== new URL(baseUrl).origin) throw new Error("Invalid API origin");
  const body = payload === undefined ? "" : JSON.stringify(payload);
  const timestamp = String(Math.floor(Date.now() / 1000));
  const nonce = randomBytes(16).toString("hex");
  const target = decodeURIComponent(url.pathname) + url.search;
  const bodyHash = createHash("sha256").update(body, "utf8").digest("hex");
  const canonical = ["pp-operator-v1", timestamp, nonce,
    method.toUpperCase(), target, bodyHash].join("\n");
  const signature = createHmac("sha256", apiKey).update(canonical, "utf8").digest("hex");
  const response = await fetch(url, {
    method, redirect: "error", signal: AbortSignal.timeout(30000),
    headers: { "content-type": "application/json", "x-provider-api-key": apiKey,
      "x-pp-timestamp": timestamp, "x-pp-nonce": nonce, "x-pp-signature": signature },
    ...(payload === undefined ? {} : { body }),
  });
  if (!response.ok) throw new Error("HTTP " + response.status + ": " + await response.text());
  return response.json();
}

const catalog = await providerRequest("GET", "/api/provider/v1/games");
// Select a symbol from catalog.data.items before launching.
const launch = await providerRequest("POST", "/api/provider/v1/launch-url", {
  symbol: "vs20doghouse", external_player_id: "operator-a:player-129001",
  currency: "USD", return_url: "https://operator.example.com/lobby"
});
// Send launch.data.launch_url to your player browser; never expose apiKey.
```

Read the operator-scoped live catalog

Catalog and launch use the same allowlist. A game outside the operator catalog is hidden from catalog responses and denied on launch.

Method	Path	Purpose
GET	/api/provider/v1/games	Returns the operator-scoped live catalog with feature flags, launch paths, artwork URLs, and optional provider_id filtering.

Method	Path	Purpose
GET	/api/provider/v1/games/{symbol}	Returns one catalog title by symbol or slug after the same operator access checks.

List games

```
# Calculate fresh signing headers using the server-side example above.
curl -X GET "https://goldencasino.cloud/api/provider/v1/games" \
-H "x-provider-api-key: <operator-api-key>" \
-H "x-pp-timestamp: <unix-seconds>" \
-H "x-pp-nonce: <fresh-nonce>" \
-H "x-pp-signature: <signature-for-this-exact-request>"
```

List games by provider

```
# Calculate fresh signing headers using the server-side example above.
curl -X GET "https://goldencasino.cloud/api/provider/v1/games?provider_id=pragmatic" \
-H "x-provider-api-key: <operator-api-key>" \
-H "x-pp-timestamp: <unix-seconds>" \
-H "x-pp-nonce: <fresh-nonce>" \
-H "x-pp-signature: <signature-for-this-exact-request>"

# Calculate fresh signing headers using the server-side example above.
curl -X GET "https://goldencasino.cloud/api/provider/v1/games?provider_id=pg" \
-H "x-provider-api-key: <operator-api-key>" \
-H "x-pp-timestamp: <unix-seconds>" \
-H "x-pp-nonce: <fresh-nonce>" \
-H "x-pp-signature: <signature-for-this-exact-request>"
```

Catalog response

```
{
  "success": true,
  "data": {
    "total": 1,
    "operator_id": "operator-a",
    "items": [
      {
        "symbol": "vs20doghouse",
        "slug": "doghouse",
        "title": "The Dog House",
        "provider_id": "pragmatic",
        "provider": "Pragmatic Play",
        "family": "line-free-spins",
        "rtp": 96.51,
        "lines": 20,
        "asset_version": "v2",
        "asset_symbol": "vs20doghouse",
        "game_service_version": "v3",
        "loader_mode": "generic",
        "launch_status": "stable",
        "live_catalog_enabled": true,
        "certified_browsers": [
          "chrome"
        ],
        "features": {
          "bonus": true,
          "collect": true,
          "free_spins": true,
          "buy_feature": true,
          "special_bet": false
        },
        "special_bet_profile": null,
        "thumbnail": {
          "symbol": "vs20doghouse",
          "slug": "doghouse",
          "title": "The Dog House",
          "url": "https://goldencasino.cloud/api/provider/v1/game-art/vs20doghouse?title=The+Dog+House&slug=doghouse"
        },
        "thumbnail_url": "https://goldencasino.cloud/api/provider/v1/game-art/vs20doghouse?title=The+Dog+House&slug=doghouse",
        "launch_path": "/games/doghouse"
      }
    ]
  }
}
```

Create a browser launch URL

Create launch_url server-side and open it in the player's browser. Treat it as a bearer secret and do not reconstruct its query parameters. For PP and PG, expires_at is the current inactivity deadline, extended by runtime activity; it can be null. For native Play'n GO, expires_at is the 30-minute play-session deadline; the signed browser launch grant expires after five minutes. Obtain a fresh launch URL to resume an expired grant; confirmed unfinished rounds keep their original wallet route, wager and currency, and may retain an expired session deadline. Crash expires_at is its five-minute signed login grant, while PostgreSQL authorization for an authenticated session lasts one hour unless revoked. An expired launch grant is not permission to refund or debit again.

Method	Path	Purpose
POST	/api/provider/v1/launch-url	Creates or refreshes a player game session and returns the browser launch URL.

Field	Description
symbol or slug	One game identifier is required. If both are supplied, they must identify the same game.

Field	Description
external_player_id	Use a stable globally unique, operator-prefixed ID of 1 to 64 characters without surrounding whitespace, such as operator-a:player-129001. PP, PG and native Play'n GO reject cross-operator reuse with 409. Crash isolates the ID by operator and currency and permits the same player ID in different operators; it preserves the supplied ID in callbacks.
coin	Optional finite positive coin value for PP/PG. Native Play'n GO and Crash select actual wagers in the game; this field does not force their stake.
currency	Defaults to operator currency then USD. PP/PG and native Play'n GO accept three-letter currency codes subject to wallet/client acceptance. An unfinished slot round must resume in its original currency; changing it returns 409. Its original coin is retained for PP/PG until settlement. Aviator supports USD only. Chicken supports USD, EUR and GBP. Crash requires the operator's configured default currency; other currencies return 422.
lang	Optional game language code; use tr for Turkish.
return_url	Optional lobby URL. Use HTTPS without credentials or control characters. PP/PG also accept HTTP; native production requires HTTPS; Crash allows loopback HTTP for isolated testing. Verify each game's close/return flow during acceptance.
device	Optional desktop, mobile, or tablet hint; slots normalize tablet to mobile. Crash clients use responsive browser layout.

Launch request

```
# Calculate fresh signing headers using the server-side example above.
curl -X POST "https://goldencasino.cloud/api/provider/v1/launch-url" \
  -H "x-provider-api-key: <operator-api-key>" \
  -H "x-pp-timestamp: <unix-seconds>" \
  -H "x-pp-nonce: <fresh-nonce>" \
  -H "x-pp-signature: <signature-for-this-exact-request>" \
  -H "content-type: application/json" \
  --data-raw '{"symbol": "vs20doghouse", "external_player_id": "operator-a:player-129001", "coin": "0.10", "currency": "USD", "lang": "en", "return_url": "https://operator.example.com/lobby", "device": "desktop"}'
```

Launch response

```
{
  "success": true,
  "symbol": "vs20doghouse",
  "slug": "doghouse",
  "operator_id": "operator-a",
  "currency": "USD",
  "lang": "en",
  "session_id": "session-token",
  "contract_version": "2026-09-01",
  "amount_unit": "minor",
  "launch_url":
    "https://goldencasino.cloud/gs2c/html5Game.do?symbol=vs20doghouse&lang=en&mgckey=...",
  "expires_at": "2026-06-16T16:30:00+00:00",
  "data": {
    "symbol": "vs20doghouse",
    "slug": "doghouse",
    "operator_id": "operator-a",
    "currency": "USD",
    "lang": "en",
    "session_id": "session-token",
    "contract_version": "2026-09-01",
    "amount_unit": "minor",
    "launch_url":
      "https://goldencasino.cloud/gs2c/html5Game.do?symbol=vs20doghouse&lang=en&mgckey=...",
    "expires_at": "2026-06-16T16:30:00+00:00"
  }
}
```

Refresh balance inside an open game session

PP only: after a deposit or adjustment, signal active PP game pages. This endpoint does not accept a balance. Unsupported game filters return 409; unfiltered requests skip unsupported providers. sessions_notified counts signals, not acknowledged browser refreshes; inspect subscribers_notified as well.

Method	Path	Purpose
POST	/api/provider/v1/player-balance-refresh	Signals active game pages for an operator/player to refresh balance through the wallet balance callback.

Field	Description
external_player_id	Required stable player identifier from the operator platform.
currency	Optional three-letter currency filter.
game_symbol	Optional game symbol or slug filter.
session_id	Optional provider session filter.
reason	Optional reason such as deposit or manual_adjustment.
request_id	Optional operator correlation id.

Balance refresh request

```
# Calculate fresh signing headers using the server-side example above.
curl -X POST "https://goldencasino.cloud/api/provider/v1/player-balance-refresh" \
  -H "x-provider-api-key: <operator-api-key>" \
  -H "x-pp-timestamp: <unix-seconds>" \
  -H "x-pp-nonce: <fresh-nonce>" \
  -H "x-pp-signature: <signature-for-this-exact-request>" \
  -H "content-type: application/json" \
  --data-raw '{"external_player_id":"operator-a:player-129001","currency":"USD","reason":"deposit","request_id":"operator-request-123"}'
```

Balance refresh response

```
{
  "success": true,
  "operator_id": "operator-a",
  "external_player_id": "operator-a:player-129001",
  "request_id": "operator-request-123",
  "reason": "deposit",
  "sessions_found": 1,
  "sessions_notified": 1,
  "subscribers_notified": 1,
  "sessions": [
    {
      "session_id": "session-token",
      "game_symbol": "vs20doghouse",
      "currency": "USD",
      "active_subscribers": 1,
      "notified": true
    }
  ],
  "data": {
    "operator_id": "operator-a",
    "external_player_id": "operator-a:player-129001",
    "request_id": "operator-request-123",
    "reason": "deposit",
    "sessions_found": 1,
    "sessions_notified": 1,
    "subscribers_notified": 1,
    "sessions": [
      {
        "session_id": "session-token",
        "game_symbol": "vs20doghouse",
        "currency": "USD",
        "active_subscribers": 1,
        "notified": true
      }
    ]
  }
}
```

Use catalog artwork URLs for game thumbnails

Catalog responses include `thumbnail` and `thumbnail_url` fields. The artwork endpoint is public and returns uploaded WebP, PNG, JPEG, or an SVG fallback.

Method	Path	Purpose
GET	/api/provider/v1/game-art/{symbol}	Returns public game artwork used by catalog thumbnail URLs.

Fetch artwork

```
curl "https://goldencasino.cloud/api/provider/v1/game-art/vs20doghouse?title=The+Dog+House&slug=doghouse"
```

Expose signed balance, debit, credit, cancel, and rollback endpoints

Callback wallet mode calls endpoints under the operator `wallet_base_url`. Golden Casino signs the exact raw JSON body with HMAC-SHA256 using the wallet secret, which is separate from the operator API key. Callback endpoints must not require `x-provider-api-key`.

Method	Path	Purpose
POST	{wallet_base_url}/balance	Operator-hosted callback endpoint signed with HMAC-SHA256.

Method	Path	Purpose
POST	{wallet_base_url}/debit	Operator-hosted callback endpoint signed with HMAC-SHA256.
POST	{wallet_base_url}/credit	Operator-hosted callback endpoint signed with HMAC-SHA256.
POST	{wallet_base_url}/cancel	Operator-hosted callback endpoint signed with HMAC-SHA256.
POST	{wallet_base_url}/rollback	Operator-hosted callback endpoint signed with HMAC-SHA256.

- Treat `transaction_id` as the idempotency key.
- Return the original result for duplicate successful transaction IDs.
- Fail closed on invalid signature, stale timestamp, insufficient funds, or unknown player.
- Rollback callbacks include `original_transaction_id`. Credit and cancel callbacks can also reference the original debit. Cancel closes an outcome with no additional balance movement; rollback reverses the referenced debit amount. These are different operations.
- Use integer `amount_minor` as the canonical amount (100 minor units per whole unit in this contract). The decimal amount field is supplied for compatibility. Return balance as a finite, non-negative decimal string with at most two decimal places.
- Callback retries can carry a fresh `request_id`, timestamp, nonce, and signature while preserving `transaction_id` and financial identity. Check the operator, player, session, round, game, currency, kind, amount, and original debit for duplicate transactions; reject conflicting duplicates.
- An uncertain callback outcome must be retried with its original `transaction_id`. Do not create a replacement debit, issue a speculative refund, or treat a timeout as rejection.
- Recovery is provider-specific: PP may compensate a failed step with rollback; PG, native Play'n GO and Crash retain durable settlement intents. An expired browser launch does not cancel an unfinished payment.

Callback headers

```
content-type: application/json
x-pp-request-id: <request-id>
x-pp-timestamp: <unix-seconds>
x-pp-nonce: <unique-32-hex-value>
x-pp-signature: <hex-hmac>
```

Signature payload

```
HMAC_SHA256(wallet_secret, pp-wallet-v1\n{timestamp}\n{nonce}\n{raw_json_body})
```

Callback request

```
{
  "contract_version": "2026-09-01",
  "operator_id": "operator-a",
  "player_id": "operator-a:player-129001",
  "session_id": "session-token",
  "game_symbol": "vs20doghouse",
  "currency": "USD",
  "request_id": "7d8a3996269f4cf3bb9f5aa6f65de3c1",
  "transaction_id": "session-token:doSpin:0:1:debit",
  "round_id": "round-01K5JQ7E4F1F5Y6Y2N1R4W9C0A",
  "transaction_type": "debit",
  "action_type": "doSpin",
  "amount": 2,
  "amount_minor": 200
}
```

Success response

```
{
  "success": true,
  "balance": "997.90",
  "currency": "USD",
  "transaction_id": "session-token:doSpin:0:1:debit"
}
```

Reject response

```
{
  "success": false,
  "error": "insufficient_funds",
  "message": "Insufficient funds"
}
```

History, reconciliation and recovery

The provider-neutral operator API exposes tenant-scoped sessions, rounds, wallet transactions, JSON reconciliation, and CSV reconciliation. Native Play'n GO also exposes its detailed stage history. Recovery controls remain provider-admin operations. Reconcile by original transaction ID, stable round ID, currency, and integer minor units.

Method	Path	Purpose
GET	/api/operator/v1/contract	Returns the frozen contract version, currencies, bet limits, amount unit, and callback set.
GET	/api/operator/v1/sessions/{session_id}	Returns one owned session, its status and activity timestamps.
GET	/api/operator/v1/transactions	Searches tenant-scoped wallet transactions by round, session, player, currency, or status.
GET	/api/operator/v1/transactions/{transaction_id}	Returns one owned wallet transaction; unknown or foreign records return 404.
GET	/api/operator/v1/rounds	Searches tenant-scoped round history using stable round IDs and integer minor-unit totals.
GET	/api/operator/v1/rounds/{round_id}	Returns one owned round and its recorded wallet transactions.
GET	/api/operator/v1/reconciliation	Compares wallet and game-history totals exactly for each currency.
GET	/api/operator/v1/reconciliation.csv	Downloads currency-separated reconciliation totals as CSV.
GET	/api/provider/v1/playngo/history	Returns native session history; session_id is a required query parameter.
GET	/api/provider/v1/playngo/history/{round_id}	Returns one native round's detailed stage history; session_id is required.

Field	Description
GET /api/operator/v1/contract	Returns contract version, supported currencies, minimum/maximum bet, callbacks and ownership fields inside success/data.
GET /api/operator/v1/sessions/{session_id}	Returns one owned session with player, game, currency, status and activity timestamps inside success/data. Unknown or foreign records return 404.
GET /api/provider/v1/playngo/history?session_id=...	Returns session_id, currency, stages and mode at the top level (no success/data wrapper). Use the session_id returned by launch. Operator keys can only read their own sessions.

Field	Description
GET /api/provider/v1/playngo/history/{round_id}?session_id=...	Returns success/data with round_id, intent, status, session_id, operator_id, currency, mode, stages, ledger and wallet. Supply a recorded round ID, not a transaction ID. Unknown rounds return 404; another operator's session returns 403.
GET /api/operator/v1/transactions	Searches only the authenticated operator's transaction journal by round, session, player, currency, or status.
GET /api/operator/v1/rounds	Returns stable round IDs with integer bet_minor and win_minor totals.
Pagination	Transactions and rounds accept limit (default 100, clamped to 1-500; zero uses 100) and offset (default 0, negative values use 0). Responses include data.items, total, limit and offset. Filters: session_id, player_id and currency; transactions also accept round_id and status (pending, success or failed).
GET /api/operator/v1/transactions/{transaction_id} and /rounds/{round_id}	Return one owned record inside success/data; round detail also includes transactions. Unknown or foreign records return 404.
GET /api/operator/v1/reconciliation	Compares wallet and game-history deltas per currency. balanced requires zero difference and no pending financial transaction.
GET /api/operator/v1/reconciliation.csv	Downloads the same currency-separated totals for external reconciliation.
Reporting totals	Admin reports are scoped by operator, provider and a single currency; there is no implicit currency conversion. history_entries counts ledger entries. The legacy rounds count in some history summaries is also an entry count, not guaranteed to be paid wagers or completed game rounds.
Recovery access	Provider-admin reconciliation is privileged. Never send a provider master key to an operator or player. Native receipt reconciliation mirrors recorded evidence without sending wallet callbacks; game settlement retries preserve the original intent. Crash recovery requests are audited and consumed by the owning worker. Do not run admin recovery from an operator launch integration.

Handle JSON errors by HTTP status

Provider API errors are returned as JSON. Operators should log request IDs and response detail fields during integration testing.

Field	Description
400	Missing required launch field or invalid request.
401	Missing/invalid API key, signature, nonce, or timestamp.
403	Inactive operator or game not enabled for this operator.
404	Game not found.
409	Reused request nonce, player identity belongs to another scope, an unfinished round has a different currency, or the requested provider operation is unsupported.
422	Invalid request or unsupported currency. detail may be a string or an array of validation errors.
500	Production/staging environment is missing valid HTTPS GAME_BASE_URL or another server configuration is invalid.

Field	Description
502 / 503	Provider runtime unavailable or not configured.

Error response

```
{
  "detail": "Operator is not active"
}
```

Go-live checklist

Complete these checks before sending production traffic through a new operator integration.

- Health endpoint returns 200.
- Public GAME_BASE_URL uses HTTPS.
- Operator status is active.
- Operator catalog contains the expected games.
- Every advertised feature and bet level passes acceptance testing with the deployed replay pool; catalog flags alone do not certify a title.
- Demo-only and sandbox-only providers are excluded from the production offer.
- Operator API key works for catalog and launch.
- Browser can open launch_url on desktop and mobile.
- Callback wallet signature verification passes.
- Balance, debit, credit, cancel, and rollback callbacks pass diagnostics.
- Operator confirms idempotency behavior for duplicate transaction_id.
- Operator confirms game close/return flow with return_url.
- For PG operator rollout, migration 0010 is applied and PG_OPERATOR_SETTLEMENT_ENABLED is enabled only after deployed desktop/mobile acceptance.

Downloads

- OpenAPI JSON: <https://docs.goldencasino.cloud/openapi.json>
- Postman collection: <https://docs.goldencasino.cloud/postman.json>
- Markdown guide: <https://docs.goldencasino.cloud/operator-guide.md>
- PDF guide: <https://docs.goldencasino.cloud/downloads/operator-api-guide.pdf>