EGW API (Quick Guide)

Table of Contents

Base URL: `https://sandbox.playegamenow.com

Auth: Processor Bearer token (Authorization header).
All responses are JSON.

Definitions:
Processor – Payment processor used by Operators and Players to transfer money for Game Platform purchases and redeems.
Operator – Operators sell Game Platform points directly to customers/players
Customer/Player – Customers/Players purchase and redeem points on Game Platforms offered by Operators.

Step 1 Get Available Games for Authenticated Processor

Notes
- This function shows all Game Platforms the Processor has selected to offer to their Operators. These are not the Operator Game Platform players play on, just the list of Game Platforms offered to Operators.

***GET** `/operator/games/`**

Returns games assigned to the authenticated processor.
Auth: Processor Bearer token (Authorization header).
`status` values:
- `active` if game is visible
- `inactive` if game is not visible

Response (200)
```json
[
 {
  "game_id": "6d882f16-477f-42d5-86a9-1780b851b589",
  "name": "Golden Dragon Mobi",
  "game_pos_url_box": "https://pos.goldendragoncity.com/pos/kiosk#",
   "status": "active"
 }
]
```

Step 2 Operator CRUD (Create, Read, Update, Disable)

Notes
All operator CRUD endpoints are scoped to the authenticated processor from the Bearer token.

A processor can only list, retrieve, update, disable, or enable operators linked to that processor.

- `operator_status` in responses is the status of the operator link for the authenticated processor:
  - `active`
  - `inactive`
- 'username': Assigned by Processor- The Operator username is used by the operator to log into the EGW operator portal where the Operator can edit their personal information,
- Password is generated automatically by EGW API and is sent to Operator email
- 'email': Operators email
- 'business_name': Operators' Business Name
- 'contact_first_name': Operators primary contact first name
- 'contact_last_name': Operators primary contact last name
- 'address': Operator Address
- 'contact_phone': Operators' primary contact phone
- 'operator_site_url': Operators URL (

Create Operator

Notes
- `email` and `contact_email` are separate fields:
- `email` belongs to the operator account (ex. sales@example.com) and receives the password reset email.
- `contact_email` belongs to the operator's primary contact and is not automatically copied from `email`.
- Password is generated automatically.
- System sends password reset link to `email`.
- Returned `operator_id` is operator Wallet user id (UUID). Use it in `/operator/g/*` endpoints.
- `username` cannot be changed after creation.

Required Fields
- `username` (string)
- `email` (email string, used for operator login/password reset)
- `contact_first_name` (string)
- `contact_last_name` (string)

Optional Fields
- `contact_email` (email string, operator business contact email)
- `business_name` (string)
- `address` (string)
- `contact_phone` (string, unique if provided)
- `operator_site_url` (URL string)

****POST** `/operator/operators/`**

Create a new operator account.
The created operator is automatically linked to the authenticated processor with `operator_status=active`.

Auth: Processor Bearer token (Authorization header).

Request Body
```json
{
  "username": "operator_demo_2",
  "email": " operator_login @example.com",
  "contact_email": "operator_contact@example.com",
  "business_name": "Example Operator",
  "contact_first_name": "John",
  "contact_last_name": "Doe",
  "address": "221B Baker Street, London",
  "contact_phone": "9195551212",
  "operator_site_url": "https://operator.example.com"
}
```

Response (201)
- Returned `operator_id` is generated by EGW API and is the operator user ID (UUID). Use it in `/operator/g/*` endpoints.

```json
{
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "username": "operator_demo_02"
 "email": "operator_login@example.com",
 "operator_status": "active",
 "business_name": " Example Operator ",
 "contact_first_name": "John",
 "contact_last_name": "Doe",
 "contact_email": "operator_contact@example.com",
 "address": "221B Baker Street, London",
 "contact_phone": 9195551212",
```

```
  "operator_site_url": "https://operator.example.com"
}
```

Common Errors
- `400 Bad Request` — validation errors.
- `400 Bad Request` — duplicate `username`.
  ```json
  {
    "username": ["Username already exists."]
  }
  ```
- `400 Bad Request` — duplicate `email`.
  ```json
  {
    "email": ["Email already exists."]
  }
  ```
- `400 Bad Request` — duplicate `contact_phone`.
  ```json
  {
    "contact_phone": ["Contact phone already exists."]
  }
  ```
- `401/403` — missing or invalid processor token.

List Operators

Query Parameters
- `status` (optional): `active`, `inactive`, or `all`
  - default: `active`
- `username` (optional): case-insensitive partial match
- `email` (optional): case-insensitive partial match against login email
- `phone` (optional): partial match against `contact_phone`
- `page` (optional): page number
- `page_size` (optional): results per page
  - default: `20`
  - maximum: `100`

Examples
```http
GET /operator/operators/
GET /operator/operators/?status=inactive
GET /operator/operators/?status=all
GET /operator/operators/?username=demo&email=example.com&phone=501
```

GET /operator/operators/?page=2&page_size=20
```

**GET** `/operator/operators/`

Returns a paginated list of operators linked to the authenticated processor.

By default, only operators with `operator_status=active` are returned.

### Response (200)
```json
{
 "count": 1,
 "next": null,
 "previous": null,
 "results": [
  {
   "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
   "username": "operator_demo_02",
   "email": "operator_login@example.com",
   "operator_status": "active",
   "business_name": "Example Operator",
   "contact_first_name": "John",
   "contact_last_name": "Doe",
   "contact_email": "operator_contact@example.com",
   "address": "221B Baker Street, London",
   "contact_phone": 9195551212",
   "operator_site_url": "https://operator.example.com"
  }
 ]
}
```

Common Errors
- `400 Bad Request` — invalid `status`.
 ```json
 {
  "status": ["Select a valid choice. invalid is not one of the available choices."]
 }
 ```

Get Individual Operator

GET `/operator/operators/{operator_id}/`

Returns one operator linked to the authenticated processor.
 ```

This endpoint can return both active and inactive operator links.

URL Param
- `operator_id` (UUID, eWallet user id)

Response (200)
```json
{
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "username": "operator_demo_02",
 "email": "operator_login@example.com",
 "operator_status": "inactive",
 "business_name": " Example Operator ",
 "contact_first_name": "John",
 "contact_last_name": "Doe",
 "contact_email": "operator_contact@example.com",
 "address": "221B Baker Street, London",
 "contact_phone": 9195551212",
 "operator_site_url": "https://operator.example.com"
}
```

Common Errors
- `404 Not Found` — operator is not linked to authenticated processor.

Update Operator

****PATCH** `/operator/operators/{operator_id}/`**

Partially updates an active operator linked to the authenticated processor.

URL Param
- `operator_id` (UUID, eWallet user id)

Allowed Request Fields
- `email`
- `contact_email`
- `business_name`
- `contact_first_name`
- `contact_last_name`
- `address`
- `contact_phone`
- `operator_site_url`

Not Allowed

- `username`
- `operator_id`
- `operator_status`

Request Body
```json
{
 "email": "new_login@example.com",
 "contact_email": "new_contact@example.com",
 "business_name": " Example Operator Updated",
 "contact_phone": "+380501112234"
}
```

Response (200)
```json
{
 "email": "new_login@example.com",
 "contact_email": "new_contact@example.com",
 "business_name": " Example Operator Updated",
 "contact_first_name": "John",
 "contact_last_name": "Doe",
 "address": "221B Baker Street, London",
 "contact_phone": "+380501112234",
 "operator_site_url": "https://operator.example.com"
}
```

Common Errors
- `400 Bad Request` — empty body.
  ```json
  {
   "detail": ["At least one field is required."]
  }
  ```
- `400 Bad Request` — field cannot be updated.
  ```json
  {
   "username": ["This field cannot be updated."]
  }
  ```
- `400 Bad Request` — duplicate `email`.
  ```json
  {
   "email": ["Email already exists."]
  ```

}
```
- `400 Bad Request` — duplicate `contact_phone`.
```json
{
  "contact_phone": ["Contact phone already exists."]
}
```
- `400 Bad Request` — operator link is disabled for authenticated processor.
```json
{
  "detail": ["Operator must be enabled for this processor before update."]
}
```
- `404 Not Found` — operator is not linked to authenticated processor.

Disable Operator

POST `/operator/operators/{operator_id}/disable/`

Disables the operator link for the authenticated processor.
This does not delete the operator globally and does not affect other processors linked to the same operator.

Response (200)
```json
{
  "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
  "username": "operator_demo_02",
  "email": "operator_login@example.com",
  "operator_status": "inactive",
  "business_name": " Example Operator ",
  "contact_first_name": "John",
  "contact_last_name": "Doe",
  "contact_email": "operator_contact@example.com",
  "address": "221B Baker Street, London",
  "contact_phone": "9195551212",
  "operator_site_url": "https://operator.example.com"
}
```

Enable Operator

POST `/operator/operators/{operator_id}/enable/`

Enables the operator link for the authenticated processor.
```

Response (200)

```json
{
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "username": "operator_demo_02",
 "email": "operator_login@example.com",
 "operator_status": "active",
 "business_name": " Example Operator ",
 "contact_first_name": "John",
 "contact_last_name": "Doe",
 "contact_email": "operator_contact@example.com",
 "address": "221B Baker Street, London",
 "contact_phone": "9195551212",
 "operator_site_url": "https://operator.example.com"
}
```

Common Errors

- `400 Bad Request` — missing/invalid fields, amount < 0.01, invalid types.
- `401/403` — missing/invalid processor token, or service disabled.
- `404 Not Found` — operator or operator_game not found / not eligible.
- `500` — error (message: "Failed to process transaction").

Step 3 Operator Games

Notes
- ‘operator_game_id’ - generated by EGW when Operator Game is created

Validation Notes
- `game_id` must be assigned to authenticated processor.
- if `OperatorGame` with same `game_id` already exists for this `operator_id` in scope of authenticated processor, API returns:
  - `400 Bad Request`
  - `{"game_id": ["Game already exists for this operator and processor."]}

Create Operator Games

Notes
- ‘game_id’: This is the Game Platform ID from Get Available Games for Authenticated Processor function.
- ‘display_name’: Name Processor or Operator chooses to call the platform (ex. Lucky’s Game Room – Golden Dragon)
- ‘pos_username’: Username Operator assigned for EGW to access Game Platform POS system.
- ‘pos_password’: Password Operator assigned for EGW to access Game Platform POS system.
- ‘publish_site’: Enter “true” or “false”. True=Game Platform available to players, False=Game Platform not available to players.
- ‘drawer’: Enter 1-5. Drawer Operator assigned for EGW to access Game Platform POS system. Used for Golden Dragon, Dragon City or Magic City only, all other Game Platforms leave blank.
- ‘kiosk_number’: Kiosk number of Golden Dragon, Dragon City or Magic City tenant only, all other Game Platforms leave blank. You will need to create multiple Operator Games to accommodate multiple Operator kiosks.
- ‘station_number’: Enter 1-5. Station number Operator assigned for EGW to access Fortune 2 Go POS system. Used for Fortune 2 Go only, all other Game Platforms leave blank.
- ‘order’: Enter 0 as default
- ‘pos_category’: Enter Game Platform POS category from list
  - Golden Dragon = “dragon”
  - Dragon City= “dragon-city”
  - Magic City= “magic”
  - River Sweeps= “river”
  - Fortune 3 go = “f2go”
  - Ultra Panda = “ultrapanda”

****POST** `/operator/{operator_id}/games/`**

Creates `OperatorGame` for operator from URL param.

Processor is taken from Bearer token.
Auth: Processor Bearer token (Authorization header).

URL Param
- `operator_id` (UUID, user id)

Request Body Golden Dragon
```json
{
 "game_id": "92d9a074-3a27-463a-9255-b8ba671df802",
 "display_name": "Golden Dragon",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "1",
 "kiosk_number": " 9628449",
 "station_number":  ,
 "order": 0,
 "pos_category": "dragon"
}
```
Response (201) Golden Dragon
```json
{
 "operator_game_id": 3,
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "game_name": "Golden Dragon",
 "game_pos_url": "https://example.com/game-pos",
 "display_name": " Golden Dragon Mobi ",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "1",
 "kiosk_number": "9628449",
 "station_number": ,
 "order": 0,

 "pos_category": "dragon"
}
```

**### Request Body Dragon City**
```

```json
{
 "game_id": " c054e606-8778-49d7-b519-edfccb899b25",
 "display_name": "Dragon City",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "4",
 "kiosk_number": "2638591",
 "station_number": ,
 "order": 0,
 "pos_category": "dragon-city"
}
```

Response (201) Dragon City

```json
{
 "operator_game_id": 1,
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "game_name": "Dragon City",
 "game_pos_url": "https://example.com/game-pos",
 "display_name": "Dragon City",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "4",
 "kiosk_number": "2638591",
 "station_number": ,
 "order": 0,
 "pos_category": "dragon-city"
}
```

Request Body Magic City

```json
{
 "game_id": " 822b6efa-1e8c-466f-a8a8-a850512784cb",
 "display_name": "Magic City",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "2",
 "kiosk_number": "15676879",
 "station_number": ,
 "order": 0,
 "pos_category": "magic"
```

}
```

### Response (201) Magic City

```json
{
 "operator_game_id": 4,
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "game_name": "Magic City",
 "game_pos_url": "https://example.com/game-pos",
 "display_name": "Magic City",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "2",
 "kiosk_number": "15676879",
 "station_number": ,
 "order": 0,
 "pos_category": "magic"
}
```

Request Body River Sweeps

```json
{
 "game_id": " e3d253f9-903d-4f1b-aeb4-5063697eff51",
 "display_name": "River Sweeps",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "",
 "kiosk_number": "",
 "station_number": ,
 "order": 0,
 "pos_category": "river "
}
```

Response (201) River Sweeps

```json
{
 "operator_game_id": 5,
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "game_name": "River Sweeps",
 "game_pos_url": "https://example.com/game-pos",
 "display_name": "River Sweeps",
 "pos_username": "cashier",
 "pos_password": "password",
```
```

```
 "publish_site": true,
 "drawer": "",
 "kiosk_number": "",
 "station_number": ,
 "order": 0,
 "pos_category": "river "
}
```

Request Body Fortune 2 go
```json
{
 "game_id": " 2f4776d5-a091-4058-a13f-42e715f00ca8",
 "display_name": "Fortune 2 go",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "1",
 "kiosk_number": "",
 "station_number": 1,
 "order": 0,
 "pos_category": "f2go"
}
```
Response (201) Fortune 2 Go
```json
{
 "operator_game_id": 2,
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "game_name": "Fortune 2 go",
 "game_pos_url": "https://example.com/game-pos",
 "display_name": "Fortune 2 go",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "1",
 "kiosk_number": "",
 "station_number": 1,
 "order": 0,
 "pos_category": "f2go"
}
```

Request Body Ultra Panda
```json
{
```

```
 "game_id": " 184aee6c-deca-4de1-bcee-fdc8af792cc1",
 "display_name": "Ultra Panda",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "",
 "kiosk_number": "",
 "station_number": ,
 "order": 0,
 "pos_category": "ultrapanda"
}
```

Response (201) Ultra Panda

```json
{
 "operator_game_id": 6,
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "game_name": " Ultra Panda ",
 "game_pos_url": "https://example.com/game-pos",
 "display_name": " Ultra Panda ",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "",
 "kiosk_number": "",
 "station_number": ,
 "order": 0,
 "pos_category": "ultrapanda"
}
```

Get Active Operator Games

****GET** `/operator/{operator_id}/games/`**

Returns active operator games (`publish_site=true`) for the given operator id.
Only games that belong to the authenticated processor are returned.

Auth: Processor Bearer token (Authorization header).

URL Param
- `operator_id` (UUID, user id)

Response (200)

```json
[
```

```
 {
  "operator_game_id": 42,
  "display_name": "Fortune 2 go",
  "publish_site": true,
  "game_pos_url": "https://example.com/game-pos",
  "game_name": "Fortune 2 go"
 }
]
```

Update Game for Operator

Notes
- The following attributes can be updated:
  "display_name":
  "pos_username":
  "pos_password":
  "publish_site":
  "drawer":
  "kiosk_number":
  "station_number":
  "order": 0,

****PATCH** `/operator/{operator_id}/games/{operator_game_id}/`**

Updates existing `OperatorGame` by id.
Record must belong to:
- operator from URL param
- authenticated processor from token

Auth: Processor Bearer token (Authorization header).

URL Params
- `operator_id` (UUID, user id)
- `operator_game_id` (integer, operator game id)

Request Body (example)
```json
{
 "display_name": "Fortune 2 go Updated",
 "publish_site": true,
 }
```

Response (200)
```json
```

```
{
 "operator_game_id": 2,
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "game_name": "Fortune 2 go Updated",
 "game_pos_url": "https://example.com/game-pos",
 "display_name": "Fortune 2 go Updated",
 "pos_username": "cashier",
 "pos_password": "password",
 "publish_site": true,
 "drawer": "",
 "kiosk_number": "",
 "station_number": 1,
 "order": 0,
 "pos_category": "f2go"
}
```

Disable all Operator Games

Notes
- Disables visibility of all operator games for this operator in scope of authenticated processor
- Only records that belong to authenticated processor are affected.
- 'updated' = number of rows actually changed.

****POST** `/operator/{operator_id}/games/disable-all/`**
(`processor_publish_site=false`).
Auth: Processor Bearer token (Authorization header).

URL Param
- `operator_id` (UUID, user id)

Response (200)
```json
{
 "detail": "All games for this operator were disabled by processor.",
 "updated": 2
}
```

Enable all Operator Games

Notes
- Enables visibility of all operator games for this operator in scope of authenticated processor
- Only records that belong to authenticated processor are affected.

- ‘updated’ = number of rows actually changed

****POST** `/operator/{operator_id}/games/enable-all/`**
(`processor_publish_site=true`).
Auth: Processor Bearer token (Authorization header).

URL Param
- `operator_id` (UUID, user id)

Response (200)
```json
{
  "detail": "All games for this operator were enabled by processor.",
  "updated": 2
}
```

Step 4 Create Customer/Add Points/Redeem Points

Notes
- Password is generated automatically.
- System sends password reset link to `email`.

Common Errors
- `400 Bad Request` — missing/invalid fields, amount < 0.01, invalid types.
- `401/403` — missing/invalid processor token, or service disabled.
- `404 Not Found` — operator or operator_game not found / not eligible.
- `500` — error (message: "Failed to process transaction").

Fields
- `payment_provider_id` (string, required) Created by EGW when payment Processor is onboarded to EGW platform.
- `operator_id` (UUID string, required) Created by EGW when operator is created on EGW platform.
- `operator_wallet_id` (string, required) Unique id, provided by payment Processor, that identifies the operator in their system.
- `player_wallet_id` (string, required) Unique id, provided by payment Processor, that identifies the player in their system.
- `amount` (decimal string, required, min 0.01)
- `operator_game_id` (integer, required) - each game will have it's unique, you will get it from operator admin – Unique id created by EGW when operator game is created on EGW platform.
- `player_game_user_id` (string, required) – The username a player uses to log into a game platform.

`player_game_user_id` Validation (purchase/redeem)
Validation depends on game platform name:

Game platform	Regex pattern	Example
Golden Dragon	`^M-\d{3}-\d{3}-\d{3}$`	`M-290-204-128`
Golden Dragon City	`^M-\d{3}-\d{3}-\d{3}$`	`M-598-461-474`
Fortune 2 go (`f2go`)	`^\d{4}-\d{6}$`	`1601-230020`
River Sweeps	`^\d{2}(?:-\d{2}){5}$`	`95-72-66-97-03-07`
Magic City	`^M-\d{3}-\d{3}-\d{3}$`	`M-000-071-456`
Ultra Panda	`^[A-Za-z0-9]{1,64}$`	`DemoPlayer1`

Notes on Async Processing
These endpoints only initiate a transaction.
The returned `session_id` is the transaction UUID for later status checks (handled separately).

Webhook (Session Result)
After you call create, redeem or purchase you will get session_id, after this you need to wait webhook in which EGW will send you information about status of this session(success, failed) and additional information, such as created customer id

Webhook details:
- EGW send a `POST` request to processor webhook URL configured in admin panel.
- Request body is JSON.
- Headers:
  - `Content-Type: application/json`
  - `Authorization: Bearer <processor.webhook_secret>`
  - `X-Idempotency-Key: tx:<session_id>`

Webhook payload example:
```json
{
  "event": "operator.transaction",
  "action": "purchase",
  "session_id": "5a1f6e95-4fb1-4fbe-ab03-4588f0f76c0f",
  "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
  "operator_game_id": 2,
  "payment_provider_id": "Leonardo ",
  "status": "success",
  "amount": "10.00",
  "fee": null,
  "player_wallet_id": "123",
  "player_game_user_id": "1601-230020",
  "game_name": "Fortune 2 go",
  "receipt": {
    "account_name": "John Doe",
    "purchase_amount": "10.00",
    "timestamp": "2026-05-23T12:30:45Z",
    "site_url": "www.fortune2go20.com",
    "game": "Fortune 2 go"
  },
  "error_message": null
}
```

Webhook `curl` example (for receiver-side testing):
```bash
curl -X POST "https://processor.example.com/webhook" \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <processor.webhook_secret>" \
  -H "X-Idempotency-Key: tx:5a1f6e95-4fb1-4fbe-ab03-4588f0f76c0f" \
  -d '{
```

```
  "event": "operator.transaction",
  "action": "purchase",
  "session_id": "5a1f6e95-4fb1-4fbe-ab03-4588f0f76c0f",
  "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
  "operator_game_id": 2,
  "payment_provider_id": "Leonardo ",
  "status": "success",
  "amount": "10.00",
  "fee": null,
  "player_wallet_id": "123",
  "player_game_user_id": "1601-230020",
  "game_name": "Fortune 2 go",
  "receipt": {
    "account_name": "John Doe",
    "purchase_amount": "10.00",
    "timestamp": "2026-05-23T12:30:45Z",
    "site_url": "www.fortune2go20.com",
    "game": "Fortune 2 go"
  },
  "error_message": null
 }'
```

Create Customer + Add Points

****POST** `/operator/g/create/`**

Creates a new customer on game platform and immediately adds points to their balance.

Request Body

```json
{
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "payment_provider_id": "Leonardo ",
 "operator_wallet_id": "1231",
 "player_wallet_id":  "123",
 "amount": "25.00",
 "operator_game_id": 2,
 "customer": {
  "first_name": "John",
  "last_name": "Doe",
  "email": "john@example.com",
  "date_of_birth": "1975-01-18"
 }
}
```

Response (201)
```json
{
 "session_id": "uuid",
 "amount": "25.00",
 "operator_game_id": 2,
 "status": "pending"
}
```

Purchase (add points to existing customers)

****POST** `/operator/g/purchase/`**

Adds points to an existing game platform customer.

Request Body
```json
{
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "payment_provider_id": "Leonardo ",
 "operator_wallet_id": "1231",
 "player_wallet_id": "123",
 "amount": "10.00",
 "operator_game_id": 2,
 "player_game_user_id": "1601-230020"
}
```

Response (201)
```json
{
 "session_id": "uuid",
 "amount": "10.00",
 "operator_game_id": 2,
 "status": "pending"
}
```

Redeem (withdraw points from existing customers)

****POST** `/operator/g/redeem/`**

Withdraws points from an existing game platform customer.

Request Body

```json
{
 "operator_id": "73b83a81-0293-45ae-99ba-bdd7f3555aa4",
 "payment_provider_id": "Leonardo ",
 "operator_wallet_id": "1231",
 "player_wallet_id":  "123",
 "amount": "5.00",
 "operator_game_id": 2,
 "player_game_user_id": "1601-230020"
}
```

Response (201)

```json
{
 "session_id": "uuid",
 "amount": "5.00",
 "operator_game_id": 2,
 "status": "pending"
}
```