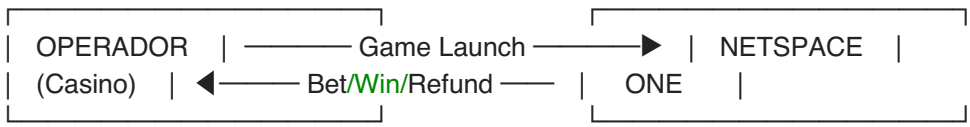
Game Aggregator API - Referencia de Integración

> Documento sintetizado basado en el estándar de la industria (Slotegrator API v1.2.1)
> Aplicable a la integración de Netspace ONE

Resumen Ejecutivo

Un Game Aggregator actúa como intermediario entre los **proveedores de juegos** y los **operadores de casino**. La integración tiene **dos direcciones de comunicación**:



---

1. Llamadas del Operador a Netspace ONE

1.1 Obtener Catálogo de Juegos

```
GET /api/games/catalog
```

Retorna lista de juegos disponibles con: uuid, nombre, imagen, tipo, provider, tecnología.

1.2 Iniciar Sesión de Juego (Dinero Real)

```
POST /api/game/launch
```

Request:

```
{
  "player_id": "player123",
  "game_id": "lotterix",
  "currency": "USD",
  "language": "en",
  "return_url": "https://casino.com/lobby"
}
```

Response:

```
{
  "session_id": "sess_abc123",
  "game_url": "https://game.netspace.one?session=sess_abc123"
}
```

1.3 Iniciar Sesión Demo

```
POST /api/game/launch-demo
```

No requiere autenticación. Solo game_id y opcionalmente language.

---

2. Callbacks de Netspace ONE al Operador

El operador debe implementar un **endpoint único** que recibe todas las acciones.

2.1 Balance (Consulta de Saldo)

POST {webhook_url}
action=balance

Request:

Campo	Tipo	Descripción
action	string	"balance"
player_id	string	ID del jugador
currency	string	Moneda
session_id	string	ID de sesión

Response esperada:

```
{
  "balance": 1500.50
}
```

2.2 Bet (Apuesta)

POST {webhook_url}
action=bet

Request:

Campo	Tipo	Descripción
action	string	"bet"
amount	double	Monto de apuesta
currency	string	Moneda
player_id	string	ID del jugador
transaction_id	string	ID único de transacción
session_id	string	ID de sesión
game_id	string	ID del juego
round_id	string	ID de la ronda
type	string	"bet", "tip", "freespin"
is_free_bet	boolean	(Opcional) true si es Free Bet
operator_free_bet_id	string	(Opcional) ID del Free Bet del operador
netspace_free_bet_id	string	(Opcional) ID interno de Netspace

Free Bet: Cuando is_free_bet: true, el operador NO debe descontar del balance real.

Response esperada:

```
{
  "balance": 1490.50,
  "transaction_id": "op_txn_123"
}
```

2.3 Win (Premio)

POST {webhook_url}
action=win

Request:

Campo	Tipo	Descripción
action	string	"win"
amount	double	Monto del premio
currency	string	Moneda
player_id	string	ID del jugador
transaction_id	string	ID único de transacción
session_id	string	ID de sesión
game_id	string	ID del juego
round_id	string	ID de la ronda
type	string	"win", "jackpot", "freespin"
from_free_bet	boolean	(Opcional) true si viene de Free Bet
operator_free_bet_id	string	(Opcional) ID del Free Bet del operador
netspace_free_bet_id	string	(Opcional) ID interno de Netspace

> **Free Bet:** Cuando from_free_bet: true, el operador SÍ debe acreditar al balance real.

Response esperada:

```
{
  "balance": 1540.50,
  "transaction_id": "op_txn_124"
}
```

2.4 Refund (Reembolso)

POST {webhook_url}
action=refund

Cancela una apuesta específica. Incluye bet_transaction_id de la apuesta a reembolsar.

2.5 Rollback (Cancelación de Ronda)

POST {webhook_url}
action=rollback

Cancela todas las transacciones de una ronda. Incluye array rollback_transactions con todas las transacciones a revertir.

---

3. Autenticación HMAC

Headers Requeridos

Header	Descripción
X-API-Key	API Key del operador
X-Timestamp	Timestamp Unix (validez: 30-300 segundos)
X-Signature	Firma HMAC-SHA256

Cálculo de Firma (Netspace ONE)

```
const timestamp = Date.now().toString();
const stringToSign = timestamp + method + path + body;
const signature = HMAC_SHA256(secretKey, stringToSign);
```

Validación por el Operador

El operador DEBE validar:

1. X-API-Key coincide con su API Key
2. X-Timestamp no es mayor a 5 minutos
3. X-Signature es válida recalculando el HMAC

---

4. Códigos de Error

Código	Descripción	Uso
INSUFFICIENT_FUNDS	Saldo insuficiente	En bet cuando no hay saldo
INTERNAL_ERROR	Error interno	Cualquier otro error
INVALID_SESSION	Sesión inválida	Sesión expirada o no existe
DUPLICATE_TRANSACTION	Transacción duplicada	Ya procesada (responder con éxito)
PLAYER_NOT_FOUND	Jugador no existe	Player ID no válido
INVALID_SIGNATURE	Firma inválida	HMAC no coincide

Formato de error:

```
{
  "error_code": "INSUFFICIENT_FUNDS",
  "error_description": "Not enough balance"
}
```

---

5. Idempotencia

CRÍTICO: Cada transaction_id debe procesarse UNA SOLA VEZ.

Si el operador recibe la misma transacción dos veces:

1. Verificar si ya fue procesada
2. Si ya existe, retornar el mismo resultado exitoso
3. NO procesar de nuevo (evita cobros duplicados)

---

6. Self-Validation (Tests de Integración)

Endpoint para que el operador valide su implementación:

```
POST /api/self-validate
```

Netspace ONE envía una serie de requests de prueba (balance, bet, win, refund, rollback) y retorna si la implementación es correcta.

Response:

```
{
  "success": true,
  "log": [
    "✓ Balance check passed",
    "✓ Bet transaction passed",
    "✓ Win transaction passed",
    "✓ Insufficient balance handled correctly",
    "✓ Rollback passed",
    "✓ Invalid signature rejected"
  ]
}
```

---

7. Flujo Completo de Integración

```
1. Operador obtiene credenciales (API Key + Secret Key)
2. Operador implementa webhook endpoint
3. Operador llama POST /api/game/launch
4. Jugador es redirigido a game_url
5. Netspace consulta balance → Operador responde
6. Jugador apuesta → Netspace envía bet → Operador descuenta y responde
7. Resultado del juego → Netspace envía win → Operador acredita y responde
8. Jugador cierra juego
9. Operador ejecuta self-validate para verificar integración
10. Operador obtiene certificación
```

---

8. Diferencias Netspace ONE vs Estándar

Aspecto	Estándar (Slotegrator)	Netspace ONE
Algoritmo firma	HMAC-SHA1	HMAC-SHA256
Formato request	form-urlencoded	JSON
Header API Key	X-Merchant-Id	X-API-Key
Header Firma	X-Sign	X-Signature
Nonce	Requerido	No requerido

9. Checklist de Implementación para Operadores

- [ ] Implementar endpoint de webhook
- [ ] Manejar action=balance
- [ ] Manejar action=bet (incluir INSUFFICIENT_FUNDS)
- [ ] Manejar action=win
- [ ] Manejar action=refund
- [ ] Manejar action=rollback
- [ ] Validar headers HMAC en cada request
- [ ] Implementar idempotencia por transaction_id
- [ ] Rechazar requests con timestamp expirado
- [ ] Rechazar requests con firma inválida
- [ ] Pasar todos los tests de self-validate
- [ ] Obtener certificación