Redemption API

Business Partner Integration Guide

Version	1.0
Base URL	https://giftcardshop.co
Last Updated	June 2026
Classification	Confidential

1. Overview

When a customer tops up their balance on your platform, they complete payment through our checkout flow. The Payment Provider emails them an activation key directly. Your backend then calls our **Redeem API** to register that redemption against your Business Partner account, recording the transaction and unlocking any wallet or balance logic on your side.

The integration follows a simple two-step handshake:

```
Customer completes payment via checkout
   → Payment Provider emails activation key to customer
   → Customer enters key in your app
   → Your backend calls POST /api/redeem
   → We validate & record the redemption
   → You credit the customer's account
```

2. Get Your API Key

Log in to the Business Partner portal and navigate to:

Profile → API Keys

Click **Generate Key**. You will be shown your key exactly once. Copy it immediately and store it securely (in an environment variable, secret manager, or similar). The key format is:

```
apk_Xk9mR3vTqZpL2wN8cYdA...  (64 characters total)
```

> ⚠ **Important** This key is shown only once. If you lose it, you must regenerate a new one. Treat it like a password — never commit it to source code or expose it in frontend JavaScript.

3. Make the Redeem Request

Endpoint

```
POST  https://giftcardshop.co/api/redeem
```

Required Headers

Header	Required	Value
Authorization	Yes	Bearer apk_YOUR_KEY_HERE
Content-Type	Yes	application/json
Accept	Yes	application/json
Idempotency-Key	Recommended	A unique string per request (see Section 5)

Request Body

```
{
  "key":              "ABCD-1234-EFGH-5678",
  "partner_user_id":  "user_abc123",
  "idempotency_key":  "order_9981_attempt_1"
}
```

Field	Type	Required	Description
key	string	Yes	The activation key the customer received from the Payment Provider. Must be 4–200 characters, alphanumeric, dashes and underscores only.

Field	Type	Required	Description
partner_user_id	string	Yes	Your internal user identifier for the customer who is redeeming. Stored for transaction reconciliation. Max 255 characters.
idempotency_key	string	Recommended	A unique string for this redemption attempt. On retries, use the same key to prevent double-processing. Max 255 characters.

Note The idempotency_key can also be sent as the Idempotency-Key HTTP header instead of in the request body. Both are accepted.

4. Handle the Response

Success — 200 OK

The key was valid, had not been previously redeemed, and has now been marked as redeemed.

```
{
  "status":           "success",
  "transaction_id":   42,
  "partner_user_id":  "user_abc123",
  "amount_cents":     5302,
  "currency":         "EUR"
}
```

Field	Description
status	Always "success" on a 200 response.
transaction_id	Internal transaction ID. Save this for your records and support queries.
partner_user_id	The user ID you sent, echoed back for confirmation.
amount_cents	Face value in cents (e.g. 5302 = €53.02).
currency	ISO 4217 currency code, typically EUR.

Action Required Upon receiving a 200 response, credit the customer's account immediately. The key is now permanently marked as redeemed in our system.

Error Responses

All errors return a consistent JSON structure:

```
{
  "status":           "<error_code>",
  "transaction_id":   43,
  "partner_user_id":  "user_abc123"
}
```

HTTP Status	Status Field	Meaning	Action
400	`invalid_format`	Key contains illegal characters or is too short/long.	Validate the key format: letters, numbers, dashes, underscores, 4–200 characters.
404	`invalid_key`	Key not found in our system for your partner account.	Key may be wrong, belong to another partner, or not yet issued. Ask the customer to double-check.
409	`already_redeemed`	This key was already successfully redeemed.	Do not credit again. Check transaction_id to confirm original redemption.
403	`blocked`	The key has been administratively blocked.	Contact platform support. Do not retry.
500	`credit_failed`	Internal error on our end.	Safe to retry after a short delay. Use the same idempotency key.
401	(no body)	Missing or invalid API key.	Check your Authorization header and key validity.
429	(no body)	Rate limit exceeded (120 req/min per key).	Back off and retry after 60 seconds.

5. Idempotency

Network failures happen. If you send a request and never receive a response (timeout, dropped connection), you cannot know whether it was processed. Idempotency keys solve this problem.

The safe pattern:

1. Generate a unique `idempotency_key` before making the request (e.g. a UUID, or `order_{id}_redeem`).
2. If the request times out or returns a 500, retry with the exact same `idempotency_key`.
3. Our system looks up the original attempt and returns the same response — no double-processing occurs.

```
// JavaScript example
const idempotencyKey = `order_${orderId}_redeem`;

// First attempt (times out)
```

```
await fetch('/api/redeem', {
  body: JSON.stringify({ key, partner_user_id, idempotency_key:
idempotencyKey })
});

// Retry — safe, same idempotency_key
await fetch('/api/redeem', {
  body: JSON.stringify({ key, partner_user_id, idempotency_key:
idempotencyKey })
});
```

⚠ **Warning** Do NOT generate a new idempotency key per retry. That would create a second, separate attempt and risk double-crediting the customer.

6. Code Examples

cURL

```
curl -X POST https://giftcardshop.co/api/redeem \
  -H "Authorization: Bearer apk_YOUR_KEY_HERE" \
  -H "Content-Type: application/json" \
  -H "Accept: application/json" \
  -H "Idempotency-Key: order_9981_redeem" \
  -d '{
    "key": "ABCD-1234-EFGH-5678",
    "partner_user_id": "user_abc123"
  }'
```

JavaScript (Node.js / fetch)

```
const response = await fetch('https://giftcardshop.co/api/redeem', {
  method: 'POST',
  headers: {
    'Authorization': 'Bearer apk_YOUR_KEY_HERE',
    'Content-Type': 'application/json',
    'Accept': 'application/json',
    'Idempotency-Key': 'order_9981_redeem',
  },
  body: JSON.stringify({
    key: 'ABCD-1234-EFGH-5678',
    partner_user_id: 'user_abc123',
  }),
});
```

```
const data = await response.json();

if (data.status === 'success') {
  console.log(`Redeemed €${data.amount_cents / 100}`);
  console.log(`Transaction ID: ${data.transaction_id}`);
} else {
  console.error('Redeem failed:', data.status);
}
```

PHP (Guzzle)

```
$client = new \GuzzleHttp\Client();

$response = $client->post('https://giftcardshop.co/api/redeem', [
    'headers' => [
        'Authorization'   => 'Bearer apk_YOUR_KEY_HERE',
        'Content-Type'    => 'application/json',
        'Accept'          => 'application/json',
        'Idempotency-Key' => 'order_9981_redeem',
    ],
    'json' => [
        'key'             => 'ABCD-1234-EFGH-5678',
        'partner_user_id' => 'user_abc123',
    ],
]);

$data = json_decode($response->getBody(), true);

if ($data['status'] === 'success') {
    $amountEur = $data['amount_cents'] / 100;
    $txId = $data['transaction_id'];
}
```

Python (requests)

```
import requests

response = requests.post(
    'https://giftcardshop.co/api/redeem',
    headers={
        'Authorization': 'Bearer apk_YOUR_KEY_HERE',
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        'Idempotency-Key': 'order_9981_redeem',
    },
    json={
```

```
        'key': 'ABCD-1234-EFGH-5678',
        'partner_user_id': 'user_abc123',
    }
)

data = response.json()

if data['status'] == 'success':
    print(f"Redeemed EUR {data['amount_cents'] / 100:.2f}")
    print(f"Transaction: {data['transaction_id']}")
else:
    print(f"Failed: {data['status']}")
```

7. Complete Integration Flow

The table below illustrates the end-to-end redemption flow between the customer, your backend, and our API.

Step	From	To	Action
1	Customer	Your App	Enters activation key received via email from the Payment Provider
2	Your Backend	Our API	POST /api/redeem with Bearer token, key, and partner_user_id
3	Our API	—	Validates format, looks up key, checks status, marks as redeemed
4	Our API	Your Backend	Returns 200 with status, transaction_id, and amount_cents
5	Your Backend	Customer	Credits wallet / balance in your system

8. Rate Limits

Limit	Value
Requests per minute (per API key)	120
HTTP status when exceeded	429 Too Many Requests

If you hit the rate limit, wait 60 seconds before retrying. Always use idempotency keys on retries to prevent duplicate processing.

9. Security Best Practices

1. Never expose your API key in frontend code. All calls to /api/redeem must originate from your backend server — never from a browser or mobile app directly.

2. Store your key in environment variables. Never hardcode it in source files. Example:

```
# .env
PLATFORM_API_KEY=apk_YOUR_KEY_HERE
```

3. Rotate your key periodically. Regenerate it in the portal under Profile → API Keys. Old keys are invalidated immediately upon regeneration.

4. Log transaction_id on every successful redeem. This is what our support team uses to investigate any disputes.

5. Never retry on 409 already_redeemed. This is a terminal state — the key has been used. Do not credit the customer again.

10. Viewing Your Transactions

All redemptions are visible in the Business Partner portal under:

Business Partner Portal → Transactions

Each entry shows the date, partner_user_id, key (last 4 characters only), amount, and status.

11. Support

If you encounter any issues — wrong keys, missing transactions, or integration questions — please contact the platform team and include the following information:

Information	Details
Your partner account ID	Shown in your profile page
`transaction_id`	From the API response (if available)
Last 4 characters of the key	Never share the full key
Approximate time of the request	Include timezone

Quick Reference Card

Item	Value
Base URL	https://giftcardshop.co
Endpoint	POST /api/redeem
Auth	Bearer apk_YOUR_KEY_HERE
Required Headers	Authorization, Content-Type, Accept
Required Body Fields	`key`, `partner_user_id`
Recommended	`idempotency_key` (body or header)
Success Response	200 OK with status, transaction_id, amount_cents, currency
Rate Limit	120 requests/minute per API key
Key Format	4–200 chars, alphanumeric + dashes + underscores
Portal	Profile → API Keys (generate), Transactions (view)