SuperPay Bank Partner API

Integration Guide

Effective November 14, 2026

1. Overview

The SuperPay Bank Partner API lets financial institutions embed card-optimization intelligence directly into their digital banking surfaces. Two endpoints are available:

• POST /v1/bank/recommend — Real-time, single-transaction card recommendation.

• POST /v1/bank/portfolio — Cross-portfolio gap analysis for a set of cards and a spending profile.

Base URL: https://superpayrewards.com

2. Authentication

All requests must include your sp_bank_… key in the Authorization header:

`Authorization: Bearer sp_bank_<your-key>`

Keys are issued after application approval. Sandbox keys (sp_bank_...) are available at superpayrewards.com/for-banks for testing. Production keys require approval from hello@superpayrewards.com.

3. POST /v1/bank/recommend

Returns the best card(s) from the SuperPay catalog (or your scoped portfolio) for a given transaction.

• amount (number, USD) — required. Must be positive and "d 1,000,000.

• mcc (string, preferred) — ISO 18245 4-digit MCC. Takes precedence over category.

• category (string, fallback) — used when MCC is absent or unmapped.

• merchant_name (string, recommended) — display name for name-based resolution.

• card_product_ids (string[], optional) — scope results to specific card IDs.

```
curl -X POST https://superpayrewards.com/v1/bank/recommend \
  -H "Authorization: Bearer sp_bank_xxxxxxxx_..." \
  -H "Content-Type: application/json" \
  -d '{"amount":87.45,"mcc":"5411","merchant_name":"Whole Foods Market"}'
```

```
const res = await fetch('https://superpayrewards.com/v1/bank/recommend', {
  method: 'POST',
  headers: { 'Authorization': 'Bearer sp_bank_...', 'Content-Type': 'application/
json' },
  body: JSON.stringify({ amount: 87.45, mcc: '5411', merchant_name: 'Whole Foods
Market' }),
});
const { top_cards, category } = await res.json();
console.log(`Best card: ${top_cards[0].name} — ${top_cards[0].reason}`);
```

```
import requests
response = requests.post(
    'https://superpayrewards.com/v1/bank/recommend',
    headers={'Authorization': 'Bearer sp_bank_...', 'Content-Type': 'application/
json'},
    json={'amount': 87.45, 'mcc': '5411', 'merchant_name': 'Whole Foods Market'},
)
best = response.json()['top_cards'][0]
print(f"Best card: {best['name']} — {best['reason']}")
```

• top_cards[] — ranked list of cards (rank, id, name, reward_rate, reason, estimated_reward_cents)

• category — resolved spend category (e.g. 'groceries')

• portfolio_scoped — true when card_product_ids filtered the results

4. POST /v1/bank/portfolio

Analyzes a set of card products against a spending profile and returns projected annual rewards, optimal allocation by category, and market gap analysis.

Access level required: partners with accessLevel 'recommend' receive HTTP 403 ACCESS_DENIED.

• card_product_ids (string[], required) — SuperPay card IDs to analyze. Max 100.

• spend_profile (object, required) — { [category]: monthly_usd_spend }. Omitting returns 400 MISSING_SPEND_PROFILE.

```
curl -X POST https://superpayrewards.com/v1/bank/portfolio \
  -H "Authorization: Bearer sp_bank_xxxxxxxx_..." \
  -H "Content-Type: application/json" \
  -d '{"card_product_ids":
["amex_gold","chase_sapphire_preferred"],"spend_profile":
```

```
{"groceries":800,"dining":400}}'
```

```
const res = await fetch('https://superpayrewards.com/v1/bank/portfolio', {
  method: 'POST',
  headers: { 'Authorization': 'Bearer sp_bank_...', 'Content-Type': 'application/
json' },
  body: JSON.stringify({
    card_product_ids: ['amex_gold', 'chase_sapphire_preferred'],
    spend_profile: { groceries: 800, dining: 400, travel: 600 },
  }),
});
const data = await res.json();
for (const card of data.cards) {
  // card.rewardsByCategory = { groceries: { reward_rate: 4, annual_reward_cents:
38400 }, … }
  console.log(`${card.name}: $
${(card.projected_annual_reward_cents/100).toFixed(2)}/yr`);
}
```

```
import requests
response = requests.post(
    'https://superpayrewards.com/v1/bank/portfolio',
    headers={'Authorization': 'Bearer sp_bank_...', 'Content-Type': 'application/
json'},
    json={'card_product_ids': ['amex_gold', 'chase_sapphire_preferred'],
          'spend_profile': {'groceries': 800, 'dining': 400}},
)
data = response.json()
for cat, winner in data['optimal_allocation'].items():
    print(f"{cat}: {winner['card_name']} @ {winner['reward_rate']}%")
print(f"Annual rewards: ${data['summary']
['total_projected_annual_reward_cents']/100:.2f}")
```

• cards[] — per card: found_in_catalog, projected_annual_reward_cents (sum across all spend categories), rewardsByCategory: { [cat]: { reward_rate, annual_reward_cents } }

• optimal_allocation — category-keyed object: { groceries: { card_id, card_name, reward_rate, estimated_monthly_reward_cents } }

• summary — total_projected_annual_reward_cents, total_monthly_gap_cents, portfolio_card_count

• by_category[] — per-category analysis with best_portfolio_card and best_market_card

5. Go-Live Checklist

Complete every step before switching from sandbox to your production key:

1. Smoke test sandbox
POST /v1/bank/recommend with sandbox key + real MCC. Expect HTTP 200, top_cards[0].reward_rate > 0, X-RateLimit-Limit: 300 header.

2. Portfolio smoke (if using)
POST /v1/bank/portfolio with 2-3 card IDs and spend_profile object. Expect HTTP 200, optimal_allocation keyed by category, cards[].rewardsByCategory present.

3. Register production origins
Confirm all prod domains are listed in your approved application. Browser request from prod origin must return 200, not 403 ORIGIN_NOT_ALLOWED.

4. Secure key storage
Load sp_bank_… from secrets manager (AWS Secrets Manager, Vault, etc.). Key must not appear in source control, CI logs, or browser network requests.

5. Handle rate limits
Read X-RateLimit-Remaining on every response. Implement exponential back-off on 429 RATE_LIMITED. Retry-After header indicates wait time.

6. Handle errors gracefully
Map 400/401/403/500 codes to UX states (fallback card, hide widget). Never surface raw error JSON to end users.

7. Switch to production key
Replace sandbox key in secrets manager. Verify first live call returns source: bank_partner_api.

8. Monitor usage
Alert on 429 rate > 1% and p95 latency > 500ms. Contact hello@superpayrewards.com for quota increases.

6. Error Codes

`401 MISSING_API_KEY` — Authorization header absent or malformed.
`401 INVALID_API_KEY` — Key not recognized or revoked.
`400 INVALID_AMOUNT` — amount missing, zero, or >$10,000.
`400 NO_CARDS_IN_PORTFOLIO` — Requested card IDs not in partner's authorized portfolio.
`400 MISSING_CARD_IDS` — /portfolio requires card_product_ids.
`400 PORTFOLIO_NOT_CONFIGURED` — /recommend: non-sandbox key with no portfolio configured — set allowedCardProductIds or pass card_product_ids.
`403 ORIGIN_NOT_ALLOWED` — Request origin not registered for this key.
`429 RATE_LIMITED` — Exceeded 300 req/min per merchant key.

6. Pricing Tiers

Starter — Free sandbox access, up to 10k calls/month in production.
Growth — Up to 100k calls/month. Volume pricing. Contact for details.
Enterprise — Unlimited calls, SLA, dedicated support. Contact hello@superpayrewards.com.

Apply at superpayrewards.com/for-banks. Questions: hello@superpayrewards.com

© 2026 SuperPay Ai, Inc. · https://superpayrewards.com/for-banks · hello@superpayrewards.com