SayFin API Reference

Version: v1.0.0 **Base URL**: `https://api.sayfin.ai` **Protocol**: HTTP/HTTPS **Content-Type**: `application/json`

Table of Contents

Authentication

We use **HMAC-SHA256** signature-based authentication to secure your API requests.

Base URL: `https://api.sayfin.ai`

✅ Recommended: Use Our SDKs (We Handle Authentication for You)

Our official SDKs **automatically handle all authentication**.

Python SDK:

```python
from sayfin import SayFin

client = SayFin(
    api_key="your_api_key_id",
    api_secret="your_api_secret"
)

recommendations = client.get_recommendations({...})
```

TypeScript SDK:

```typescript
import { SayFin } from '@sayfin/sdk';

const client = new SayFin({
  apiKey: 'your_api_key_id',      // We provide this to you
  apiSecret: 'your_api_secret'    // We provide this to you
});

const recommendations = await client.getRecommendations({...});
```

Installation:

```bash
# Python
pip install sayfin

# TypeScript/Node.js
npm install @sayfin/sdk
```

Alternative: Manual HMAC-SHA256 Implementation

If you're using a language other than Python or TypeScript, you'll need to implement HMAC-SHA256 authentication yourself.

What You Need

1. **Your API Credentials** (we provide these to you):

- `api_key_id` - Your unique API key identifier
- `api_secret` - Your secret key for signing requests

2. **For Each Request**, you need to:
    - Generate a current Unix timestamp
    - Create a signature message from your request details
    - Calculate an HMAC-SHA256 signature using your secret
    - Include the signature and timestamp in your request headers

Step-by-Step Implementation

Step 1: Create the Signature Message

Concatenate your request details in this exact format:

```
{METHOD}\n{PATH}\n{BODY}\n{TIMESTAMP}
```

Example:

```
POST
/v1/recommendations
{"basic_info":{"transaction_id":"txn_001",...}}
1706356800
```

Step 2: Calculate HMAC-SHA256 Signature

Use your `api_secret` to sign the message:

```python
import hmac
import hashlib

signature = hmac.new(
    api_secret.encode('utf-8'),      # Your secret
    signature_message.encode('utf-8'), # Message from Step 1
    hashlib.sha256
).hexdigest()
```

Step 3: Add Headers to Your Request

Include these headers with every authenticated request:

```
POST /v1/recommendations HTTP/1.1
Host: api.sayfin.ai
Content-Type: application/json
Authorization: HMAC-SHA256 Signature={signature}, KeyId={api_key_id}
X-Request-Timestamp: {timestamp}
```

Complete Example (Python without SDK)

```python
import hmac
import hashlib
import time
import json
import requests

# Your credentials
API_KEY_ID = "your_api_key_id"
API_SECRET = "your_api_secret"

# Request details
method = "POST"
path = "/v1/recommendations"
body = json.dumps({
    "basic_info": {
        "transaction_id": "txn_001",
        "amount": 2.20,
        "currency_code": "GBP"
    },
    "actor_info": {
        "merchant_id": "merchant_123",
        "mcc": 5499
    }
})

# Step 1: Generate timestamp
timestamp = str(int(time.time()))

# Step 2: Create signature message
signature_message = f"{method}\n{path}\n{body}\n{timestamp}"

# Step 3: Calculate HMAC-SHA256 signature
signature = hmac.new(
    API_SECRET.encode('utf-8'),
    signature_message.encode('utf-8'),
    hashlib.sha256
).hexdigest()

# Step 4: Make authenticated request
response = requests.post(
    f"https://api.sayfin.ai{path}",
```

```
    headers={
        "Content-Type": "application/json",
        "Authorization": f"HMAC-SHA256 Signature={signature}, KeyId={API_K
        "X-Request-Timestamp": timestamp
    },
    data=body
)

print(response.json())
```

Complete Example (JavaScript/Node.js without SDK)

```
const crypto = require("crypto");
const axios = require("axios");

// Your credentials
const API_KEY_ID = "your_api_key_id";
const API_SECRET = "your_api_secret";

// Request details
const method = "POST";
const path = "/v1/recommendations";
const body = JSON.stringify({
  basicInfo: {
    transactionId: "txn_001",
    amount: 2.2,
    currencyCode: "GBP",
  },
  actorInfo: {
    merchantId: "merchant_123",
    mcc: 5499,
  },
});

// Step 1: Generate timestamp
const timestamp = Math.floor(Date.now() / 1000).toString();

// Step 2: Create signature message
const signatureMessage = `${method}\n${path}\n${body}\n${timestamp}`;

// Step 3: Calculate HMAC-SHA256 signature
const signature = crypto
  .createHmac("sha256", API_SECRET)
  .update(signatureMessage)
  .digest("hex");

// Step 4: Make authenticated request
axios
  .post(`https://api.sayfin.ai${path}`, body, {
    headers: {
      "Content-Type": "application/json",
      Authorization: `HMAC-SHA256 Signature=${signature}, KeyId=${API_KEY_
      "X-Request-Timestamp": timestamp,
```

```
  },
})
.then((response) => console.log(response.data))
.catch((error) => console.error(error));
```

Authentication Headers

Header	Required	Description	Example
`Authorization`	Yes	HMAC signature + Key ID	`HMAC-SHA256 Signature=abc123..., KeyId=key_456`
`X-Request-Timestamp`	Yes	Unix timestamp (seconds)	`1706356800`
`Content-Type`	Yes	Must be `application/json`	`application/json`

Common Authentication Errors

If you encounter authentication errors, here's what they mean and how to fix them:

Status Code	Error	What It Means	How to Fix
401 Unauthorized	Invalid signature	Your signature calculation is incorrect	Double-check your signature logic
401 Unauthorized	Unknown key	Your `api_key_id` is invalid	Verify you're using the correct key
401 Unauthorized	Timestamp expired	Your request timestamp is more than 5 min old	Use a current timestamp
403 Forbidden	Key expired	Your API key has expired	Contact us for a new key
403 Forbidden	Key revoked	Your API key was revoked	Contact support@sayfin.com

Getting Your API Credentials

How to Get Started:

1. **Contact us**: Email sales@sayfin.com or support@sayfin.com
2. **We provide your credentials**:
    - `api_key_id` (e.g., `key_abc123xyz`)
    - `api_secret` (e.g., `secret_def456uvw`)
3. **You receive credentials** via secure channel (encrypted email, password manager share)
4. **Store them securely** in your environment:
    - Environment variables
    - AWS Secrets Manager
    - HashiCorp Vault
    - Azure Key Vault
    - Other secrets management solution
5. **Configure your SDK** or code with these credentials

Example: Using Environment Variables

```
export SAYFIN_API_KEY_ID="key_abc123xyz"
export SAYFIN_API_SECRET="secret_def456uvw"
```

Then in your code:

```
import os
from sayfin import SayFin

client = SayFin(
    api_key=os.getenv('SAYFIN_API_KEY_ID'),
    api_secret=os.getenv('SAYFIN_API_SECRET')
)
```

Testing Authentication

Verify Your Credentials:

```python
from sayfin import SayFin

# Test credentials
client = SayFin(
    api_key="test_key_id",
    api_secret="test_secret",
    base_url="https://api.sayfin.ai"
)

# Should return 200 OK
response = client.check_health()
print(response.status)  # "healthy"
```

Endpoints

Health Check

Check API and database connectivity.

Endpoint: `GET /health` **Authentication**: None required **Rate Limit**: Unlimited

Request

```bash
curl https://api.sayfin.ai/health
```

Response

Success (200 OK):

```
OK
```

Failure (503 Service Unavailable):

```
Service Unavailable
```

Use Cases

- Kubernetes liveness/readiness probes
- Load balancer health checks
- Monitoring systems

Get Recommendations

Get top 3 payment acquirer recommendations based on comprehensive transaction characteristics.

Endpoint: `POST /v1/recommendations` **Authentication**: HMAC-SHA256 (production) **Rate Limit**: 100000 requests/minute

Architecture Position: SayFin sits **between the PSP (Payment Service Provider) and the acquirer**, not after the merchant. The PSP calls SayFin to get routing recommendations before selecting an acquirer.

```
Merchant → PSP → [SayFin] → Acquirer (Stripe/Adyen/etc.)
```

Request

Recommended Approach: Use our official SDKs for production integrations:

Python SDK:

```python
from sayfin import SayFin
from sayfin.types import RecommendationRequest

client = SayFin(api_key="your_api_key")

recommendation = client.get_recommendations(
    RecommendationRequest(
        transaction_id="txn_12345",
        merchant_id="merchant_abc",
        amount=2500,
        currency="EUR",
        card_scheme="visa",
        # ... additional fields
    )
)
```

TypeScript SDK:

```typescript
import { SayFin } from "@sayfin/sdk";

const client = new SayFin({ apiKey: "your_api_key" });

const recommendation = await client.getRecommendations({
  transactionId: "txn_12345",
  merchantId: "merchant_abc",
  amount: 2500,
  currency: "EUR",
  cardScheme: "visa",
  // ... additional fields
});
```

Alternative: For non-Python/TypeScript environments, use the REST API directly with cURL or your HTTP client.

Headers:

```
Content-Type: application/json
Authorization: HMAC-SHA256 Signature=<signature>, KeyId=<key_id> (producti
```

Body (Comprehensive Payload):

```
{
  "basic_info": {
    "transaction_id": "txn_12345",
    "amount": 2.2,
    "currency_code": "GBP",
    "currency_numeric": "826",
    "site_id": 6,
    "machine_au_time": "170126124639",
    "machine_au_time_exact": "2026-01-17T12:46:39.630",
    "timeout_ms": 15000,
    "payment_method_id": 1,
    "billing_provider_id": 59,
    "rrn": "117124639679"
  },
  "device_info": {
    "hw_serial": "4434334222185871",
    "fw_version": "4.0.25.2-RC05",
    "type": "VPOS Touch"
  },
  "machine_info": {
    "id": 627567943,
    "name": "4635 - PEM002 - Pembury Hosp - A&E",
    "external_id": "4434334222185871",
    "type": "Cold Drinks",
    "decimal_place": 2,
    "offset": "0.00",
    "city": "Tunbridge Wells",
    "zip_code": "TN2 4QJ",
    "country": {
      "name": "UNITED KINGDOM",
      "numeric_code": "826",
      "alpha2_code": "GB",
      "alpha3_code": "GBR"
    },
    "geo_location": {
      "state": "England",
      "city": "Tunbridge Wells",
      "country_code": "GB",
      "zip_code": "TN2 4QJ",
      "address": "",
      "latitude": 51.1484526,
      "longitude": 0.3074703
    }
```

```
  },
  "actor_info": {
    "merchant_id": "merchant_abc",
    "merchant_name": "SuperCups Vending Ltd",
    "merchant_country": {
      "name": "UNITED KINGDOM",
      "numeric_code": "826",
      "alpha2_code": "GB",
      "alpha3_code": "GBR"
    },
    "geo_location": {
      "state": "England",
      "city": "Ashford",
      "country_code": "GB",
      "zip_code": "TN24 0TT",
      "address": "Moat Way"
    },
    "mcc": 5499,
    "phone_number": "1233500321"
  },
  "custom_data": {
    "merchant_account": "Nayax_UK_H2H",
    "additional_params": {}
  },
  "card_data": {
    "masked_card_number": "483204xxxxxx2068",
    "entry_mode": "NFC",
    "exp_year": "30",
    "exp_month": "09",
    "brand_info": {
      "primary_id": "6",
      "primary_description": "VISA",
      "secondary_id": "0"
    },
    "is_debit_card": true
  },
  "products": [
    {
      "code": 0,
      "name": "Product 1",
      "price": 2.2,
      "unit_of_measurement": "Unit",
      "measurement": 1.0,
      "tax_id": "",
```

```
      "tax_value": 0.0
    }
  ],
  "tax_info": {
    "tax_percent": 0.0,
    "amount": 2.2,
    "tax_amount": 0.0,
    "total_amount": 2.2
  }
}
```


Field Descriptions:

Basic Info (Required)

Field	Type	Required	Description	Example Values
`transaction_id`	string	Yes	Unique transaction identifier	`"txn_12345"`
`amount`	float	Yes	Transaction amount	`2.20`, `150.50`
`currency_code`	string	Yes	ISO 4217 currency code (uppercase)	`"USD"`, `"EUR"`, `"GBP"`
`currency_numeric`	string	No	ISO 4217 numeric currency code	`"826"` (GBP), `"840"` (USD)
`site_id`	integer	No	Site/location identifier	`6`, `123`
`machine_au_time`	string	No	Machine authorization time	`"170126124639"`
`machine_au_time_exact`	string	No	Exact ISO 8601 timestamp	`"2026-01-17T12:46:39.630"`
`timeout_ms`	integer	No	Request timeout in milliseconds	`15000`
`payment_method_id`	integer	No	Payment method type identifier	`1`
`billing_provider_id`	integer	No	Billing provider identifier	`59`
`rrn`	string	No	Retrieval Reference Number	`"117124639679"`

Device Info (Optional)

Field	Type	Required	Description	Example Values
`hw_serial`	string	No	Hardware serial number	`"4434334222185871"`
`fw_version`	string	No	Firmware version	`"4.0.25.2-RC05"`
`type`	string	No	Device type	`"VPOS Touch"`, `"Terminal"`

Machine Info (Optional)

Field	Type	Required	Description	Example Values
`id`	integer	No	Machine/terminal ID	`627567943`
`name`	string	No	Machine/terminal name	`"Terminal - Location A"`
`external_id`	string	No	External system ID	`"4434334222185871"`
`type`	string	No	Machine type	`"Cold Drinks"`, `"POS"`
`decimal_place`	integer	No	Number of decimal places	`2`
`city`	string	No	Machine location city	`"Tunbridge Wells"`
`zip_code`	string	No	Machine location ZIP code	`"TN2 4QJ"`
`country`	object	No	Country information	See below
`geo_location`	object	No	Geographic coordinates	See below

Actor Info (Merchant Info) - Recommended for Accuracy

Field	Type	Required	Description	Example Values
`merchant_id`	string	Yes	Merchant identifier	`"merchant_abc"`
`merchant_name`	string	No	Merchant business name	`"SuperCups Vending Ltd"`
`merchant_country`	object	No	Merchant country details	See country object
`geo_location`	object	No	Merchant geographic location	See geo_location object
`mcc`	integer	Recommended	Merchant Category Code	`5499`, `5411`, `5812`
`phone_number`	string	No	Merchant phone number	`"1233500321"`

Card Data - Recommended for Accuracy

Field	Type	Required	Description	Example Values
`masked_card_number`	string	No	Masked PAN (never send full PAN)	`"483204xxxxxx2068"`
`entry_mode`	string	Recommended	Card entry method	`"NFC"`, `"Chip"`, `"Swipe"`, `"Manual"`
`exp_year`	string	No	Card expiration year (YY)	`"30"`, `"26"`
`exp_month`	string	No	Card expiration month (MM)	`"09"`, `"12"`
`brand_info`	object	Recommended	Card brand information	See below
`is_debit_card`	boolean	No	Whether card is debit (vs credit)	`true`, `false`

Brand Info Object:


```
{
  "primary_id": "6",
  "primary_description": "VISA",
  "secondary_id": "0"
}
```


Country Object:

```
{
  "name": "UNITED KINGDOM",
  "numeric_code": "826",
  "alpha2_code": "GB",
  "alpha3_code": "GBR"
}
```


Geo Location Object:


```
{
  "state": "England",
  "city": "Tunbridge Wells",
  "country_code": "GB",
  "zip_code": "TN2 4QJ",
  "address": "Moat Way",
  "latitude": 51.1484526,
  "longitude": 0.3074703
}
```


Products (Optional)

Array of products being purchased:


```
{
  "code": 0,
  "name": "Product 1",
  "price": 2.2,
  "unit_of_measurement": "Unit",
  "measurement": 1.0,
  "tax_id": "",
  "tax_value": 0.0
}
```

Tax Info (Optional)

```
{
  "tax_percent": 0.0,
  "amount": 2.2,
  "tax_amount": 0.0,
  "total_amount": 2.2
}
```

Simplified Payload (Minimum Required Fields):

For quick integration, you can start with just the essential fields:

```
{
  "basic_info": {
    "transaction_id": "txn_12345",
    "amount": 2.2,
    "currency_code": "GBP"
  },
  "actor_info": {
    "merchant_id": "merchant_abc",
    "mcc": 5499
  },
  "card_data": {
    "entry_mode": "NFC",
    "brand_info": {
      "primary_description": "VISA"
    }
  }
}
```

Note: More fields = better ML accuracy. We recommend including as many fields as available from your payment flow.

Response

Success (200 OK):

```
{
  "transaction_id": "txn_12345",
  "recommendations": [
    {
      "acquirer": "Adyen",
      "probability": 0.92,
      "rank": 1
    },
    {
      "acquirer": "Stripe",
      "probability": 0.05,
      "rank": 2
    },
    {
      "acquirer": "Checkout.com",
      "probability": 0.02,
      "rank": 3
    }
  ],
  "model_version": "v1.0.0",
  "timestamp": "2025-12-27T21:00:00Z"
}
```


Field Descriptions:

Field	Type	Description
`transaction_id`	string	Echo of request transaction ID
`recommendations`	array	Top 3 acquirer recommendations (sorted by probability)
`recommendations[].acquirer`	string	Acquirer name
`recommendations[].probability`	float	Approval probability (0.0-1.0)
`recommendations[].rank`	integer	Ranking position (1-3)
`model_version`	string	ML model version used
`timestamp`	string	ISO 8601 timestamp of prediction

Error (400 Bad Request):

```
{
  "error": "Invalid request",
  "details": "Missing required field: transaction_id"
}
```

Error (500 Internal Server Error):

```
{
  "error": "Model prediction failed",
  "details": "Failed to run ONNX inference"
}
```

Examples

Recommended: Python SDK

```python
from sayfin import SayFin

# Initialize client
client = SayFin(api_key="your_api_key")

# Get recommendations with comprehensive data
recommendations = client.get_recommendations({
    "basic_info": {
        "transaction_id": "txn_001",
        "amount": 2.20,
        "currency_code": "GBP",
        "timeout_ms": 15000
    },
    "actor_info": {
        "merchant_id": "merchant_123",
        "merchant_name": "My Store Ltd",
        "mcc": 5499,
        "merchant_country": {
            "alpha2_code": "GB"
        }
    },
    "card_data": {
        "entry_mode": "NFC",
        "brand_info": {
            "primary_description": "VISA"
        },
        "is_debit_card": True
    },
    "device_info": {
        "type": "VPOS Touch",
        "fw_version": "4.0.25"
    }
})

# Access recommendations
top_acquirer = recommendations.recommendations[0]
print(f"Route to: {top_acquirer.acquirer}")
print(f"Probability: {top_acquirer.probability:.2%}")
print(f"Rank: {top_acquirer.rank}")
```

Recommended: TypeScript SDK

```typescript
import { SayFin } from "@sayfin/sdk";

// Initialize client
const client = new SayFin({ apiKey: "your_api_key" });

// Get recommendations with comprehensive data
const recommendations = await client.getRecommendations({
  basicInfo: {
    transactionId: "txn_001",
    amount: 2.2,
    currencyCode: "GBP",
    timeoutMs: 15000,
  },
  actorInfo: {
    merchantId: "merchant_123",
    merchantName: "My Store Ltd",
    mcc: 5499,
    merchantCountry: {
      alpha2Code: "GB",
    },
  },
  cardData: {
    entryMode: "NFC",
    brandInfo: {
      primaryDescription: "VISA",
    },
    isDebitCard: true,
  },
  deviceInfo: {
    type: "VPOS Touch",
    fwVersion: "4.0.25",
  },
});

// Access recommendations
const topAcquirer = recommendations.recommendations[0];
console.log(`Route to: ${topAcquirer.acquirer}`);
console.log(`Probability: ${(topAcquirer.probability * 100).toFixed(2)}%`)
```

Alternative: REST API with cURL (for non-Python/TypeScript environments)

Note: For production use, you'll need to add authentication headers. See the Authentication section.

```
curl -X POST https://api.sayfin.ai/v1/recommendations \
  -H "Content-Type: application/json" \
  -H "Authorization: HMAC-SHA256 Signature={your_signature}, KeyId={your_a
  -H "X-Request-Timestamp: {timestamp}" \
  -d '{
    "basic_info": {
      "transaction_id": "txn_001",
      "amount": 2.20,
      "currency_code": "GBP"
    },
    "actor_info": {
      "merchant_id": "merchant_123",
      "mcc": 5499
    },
    "card_data": {
      "entry_mode": "NFC",
      "brand_info": {
        "primary_description": "VISA"
      }
    }
  }'
```

Installation:

```
# Python SDK
pip install sayfin

# TypeScript SDK
npm install @sayfin/sdk
```

JavaScript (Node.js) without SDK:

Note: This example shows raw REST API usage. For production, we recommend using our TypeScript SDK which handles authentication automatically.

```javascript
const axios = require("axios");
const crypto = require("crypto");

async function getRecommendations() {
  // Generate HMAC signature (see Authentication section for details)
  const apiKeyId = process.env.SAYFIN_API_KEY_ID;
  const apiSecret = process.env.SAYFIN_API_SECRET;
  const timestamp = Math.floor(Date.now() / 1000).toString();

  const body = JSON.stringify({
    transaction_id: "txn_001",
    merchant_id: "merchant_123",
    amount: 1500,
    currency: "USD",
    card_scheme: "visa",
  });

  const signatureMessage = `POST\n/v1/recommendations\n${body}\n${timestam
  const signature = crypto
    .createHmac("sha256", apiSecret)
    .update(signatureMessage)
    .digest("hex");

  const response = await axios.post(
    "https://api.sayfin.ai/v1/recommendations",
    body,
    {
      headers: {
        "Content-Type": "application/json",
        Authorization: `HMAC-SHA256 Signature=${signature}, KeyId=${apiKey
        "X-Request-Timestamp": timestamp,
      },
    },
  );

  console.log(`Top: ${response.data.recommendations[0].acquirer}`);
  console.log(`Probability: ${response.data.recommendations[0].probability
}

getRecommendations();
```

Performance

- **Average Latency**: 7-8ms (embedded model)
- **Cold Start**: ~58ms (first request after container start)
- **p95**: <15ms
- **p99**: <25ms

Submit Feedback

Submit transaction outcome feedback for model retraining and accuracy tracking.

Endpoint: `POST /v1/feedback` **Authentication**: HMAC-SHA256 (production) **Rate Limit**: 100000 requests/minute per merchant

Request

Headers:

```
Content-Type: application/json
Authorization: HMAC-SHA256 Signature=<signature>, KeyId=<key_id> (producti
```

Body:

```
{
  "transaction_id": "txn_12345",
  "acquirer_used": "Adyen",
  "outcome": "approved",
  "processing_time_ms": 250,
  "recommended_rank_used": 1
}
```

Field Descriptions:

Field	Type	Required	Description	Example Values
`transaction_id`	string	Yes	Must match a previous recommendation request	`"txn_12345"`
`acquirer_used`	string	Yes	Which acquirer was actually used	`"Adyen"`, `"Stripe"`
`outcome`	string	Yes	Transaction result	`"approved"`, `"declined"`, `"error"`
`processing_time_ms`	integer	No	Processing time in milliseconds	`250`
`recommended_rank_used`	integer	No	Which recommendation rank was used (1-3)	`1`, `2`, `3`, `null`

Response

Success (200 OK):

```
{
  "feedback_id": 42,
  "transaction_id": "txn_12345",
  "status": "accepted"
}
```

Field Descriptions:

Field	Type	Description
`feedback_id`	integer	Unique feedback record ID
`transaction_id`	string	Echo of request transaction ID
`status`	string	Processing status (`"accepted"`)

Error (400 Bad Request):

```
{
  "error": "Invalid request",
  "details": "transaction_id not found in recommendations table"
}
```

Error (409 Conflict):

```
{
  "error": "Duplicate feedback",
  "details": "Feedback already submitted for transaction_id: txn_12345"
}
```

Examples

cURL:

```
curl -X POST https://api.sayfin.ai/v1/feedback \
  -H "Content-Type: application/json" \
  -H "Authorization: HMAC-SHA256 Signature={your_signature}, KeyId={your_a
  -H "X-Request-Timestamp: {timestamp}" \
  -d '{
    "transaction_id": "txn_12345",
    "acquirer_used": "Adyen",
    "outcome": "approved",
    "processing_time_ms": 250,
    "recommended_rank_used": 1
  }'
```

Python (without SDK - for reference only, use SDK in production):

```python
import requests
# Note: This example omits authentication for brevity
# In production, use the SayFin SDK which handles authentication automatic

response = requests.post(
    "https://api.sayfin.ai/v1/feedback",
    json={
        "transaction_id": "txn_12345",
        "acquirer_used": "Adyen",
        "outcome": "approved",
        "processing_time_ms": 250,
        "recommended_rank_used": 1
    },
    headers={
        # Add authentication headers here (see Authentication section)
    }
)

data = response.json()
print(f"Feedback recorded: {data['success']}")
print(f"Status: {data['status']}")
```

Use Cases

- Track recommendation accuracy
- Measure which rank is typically used (1st, 2nd, 3rd)
- Collect data for model retraining
- Monitor approval rates per acquirer
- Analyze processing time performance

Compare Prediction

Compare PSP's acquirer choice against the model's top recommendation.

Endpoint: `POST /v1/compare` **Authentication**: HMAC-SHA256 (production) **Rate Limit**: 1000 requests/minute per merchant

Request

Headers:

```
Content-Type: application/json
Authorization: HMAC-SHA256 Signature=<signature>, KeyId=<key_id> (producti
```

Body:

```
{
  "transaction_id": "txn_12345",
  "acquirer_used": "Adyen"
}
```

Field Descriptions:

Field	Type	Required	Description
`transaction_id`	string	Yes	Must match a previous recommendation request
`acquirer_used`	string	Yes	Which acquirer was actually used

Response

Success (200 OK) - Match Found:

```
{
  "transaction_id": "txn_12345",
  "match_found": true,
  "predicted_rank": 1,
  "predicted_probability": 0.92,
  "all_recommendations": [
    {
      "acquirer": "Adyen",
      "probability": 0.92,
      "rank": 1
    },
    {
      "acquirer": "Stripe",
      "probability": 0.05,
      "rank": 2
    },
    {
      "acquirer": "Checkout.com",
      "probability": 0.02,
      "rank": 3
    }
  ]
}
```


Success (200 OK) - No Match:

```
{
  "transaction_id": "txn_12345",
  "match_found": false,
  "predicted_rank": null,
  "predicted_probability": null,
  "all_recommendations": [
    {
      "acquirer": "Stripe",
      "probability": 0.45,
      "rank": 1
    },
    {
      "acquirer": "Adyen",
      "probability": 0.32,
      "rank": 2
    },
    {
      "acquirer": "Checkout.com",
      "probability": 0.18,
      "rank": 3
    }
  ]
}
```


Field Descriptions:

Field	Type	Description
`transaction_id`	string	Echo of request transaction ID
`match_found`	boolean	Whether acquirer_used matches any recommendation
`predicted_rank`	integer/null	Rank of acquirer_used (1-3) or null if not in top 3
`predicted_probability`	float/null	Probability of acquirer_used or null
`all_recommendations`	array	Complete top 3 recommendations

Error (404 Not Found):

```
{
  "error": "Not found",
  "details": "No recommendations found for transaction_id: txn_12345"
}
```

Examples

cURL:

```
curl -X POST https://api.sayfin.ai/v1/compare \
  -H "Content-Type: application/json" \
  -H "Authorization: HMAC-SHA256 Signature={your_signature}, KeyId={your_a
  -H "X-Request-Timestamp: {timestamp}" \
  -d '{
    "transaction_id": "txn_12345",
    "acquirer_used": "Adyen"
  }'
```

Python (without SDK - for reference only):

```python
import requests
# Note: This example omits authentication for brevity
# In production, use the SayFin SDK which handles authentication automatic

response = requests.post(
    "https://api.sayfin.ai/v1/compare",
    json={
        "transaction_id": "txn_12345",
        "acquirer_used": "Adyen"
    },
    headers={
        # Add authentication headers here (see Authentication section)
    }
)

data = response.json()
if data['match_found']:
    print(f"Match! Predicted rank: {data['predicted_rank']}")
    print(f"Probability: {data['predicted_probability']:.2%}")
else:
    print("No match - acquirer not in top 3 recommendations")
```

Use Cases

- Validate if PSP's manual routing aligns with model
- Measure model adoption rate
- Track how often non-top-3 acquirers are used
- A/B testing between manual and automated routing

Data Models

RecommendationRequest

```
{
  "transaction_id": "string",
  "merchant_id": "string",
  "amount": "integer (cents)",
  "currency": "string (ISO 4217)",
  "card_scheme": "string",
  "merchant_category": "string",
  "customer_country": "string (ISO 3166-1 alpha-2)"
}
```

RecommendationResponse

```
{
  "transaction_id": "string",
  "recommendations": [
    {
      "acquirer": "string",
      "probability": "float (0.0-1.0)",
      "rank": "integer (1-3)"
    }
  ],
  "model_version": "string",
  "timestamp": "string (ISO 8601)"
}
```

FeedbackRequest

```
{
  "transaction_id": "string",
  "acquirer_used": "string",
  "outcome": "string (approved|declined|error)",
  "processing_time_ms": "integer (optional)",
  "recommended_rank_used": "integer (1-3, optional)"
}
```

FeedbackResponse

```
{
  "feedback_id": "integer",
  "transaction_id": "string",
  "status": "string (accepted)"
}
```

CompareRequest

```
{
  "transaction_id": "string",
  "acquirer_used": "string"
}
```

CompareResponse


```
{
  "transaction_id": "string",
  "match_found": "boolean",
  "predicted_rank": "integer|null",
  "predicted_probability": "float|null",
  "all_recommendations": [
    {
      "acquirer": "string",
      "probability": "float",
      "rank": "integer"
    }
  ]
}
```

Error Handling

HTTP Status Codes

Code	Meaning	Example Scenario
200	OK	Successful request
400	Bad Request	Missing required field, invalid JSON
401	Unauthorized	Invalid or expired API key
403	Forbidden	API key lacks permissions
404	Not Found	Transaction ID not found
409	Conflict	Duplicate feedback submission
429	Too Many Requests	Rate limit exceeded
500	Internal Server Error	Model prediction failed, database error
503	Service Unavailable	Database unavailable, health check failed

Error Response Format

All errors return JSON with the following structure:

```
{
  "error": "Brief error description",
  "details": "Detailed error message for debugging"
}
```

Common Errors

Missing Required Field

Request:

```
{
  "transaction_id": "txn_001"
  // Missing other required fields
}
```

Response (400):

```
{
  "error": "Invalid request",
  "details": "Missing required field: merchant_id"
}
```

Invalid Currency Code

Request:

```
{
  "currency": "INVALID"
}
```

Response (400):

```
{
  "error": "Invalid request",
  "details": "Invalid currency code: INVALID. Must be ISO 4217 (USD, EUR,
}
```

Transaction Not Found

Request:

```
{
  "transaction_id": "txn_nonexistent"
}
```

Response (404):

```
{
  "error": "Not found",
  "details": "No recommendations found for transaction_id: txn_nonexistent
}
```

Database Unavailable

Response (503):

```
{
  "error": "Service unavailable",
  "details": "Database connection pool exhausted"
}
```

Rate Limiting

Rate Limits (Production)

Endpoint	Limit	Window
`/v1/recommendations`	1000 requests	1 minute
`/v1/feedback`	5000 requests	1 minute
`/v1/compare`	1000 requests	1 minute
`/health`	Unlimited	-

Rate Limit Headers

Responses include rate limit information:

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 995
X-RateLimit-Reset: 1703721600
```

Rate Limit Exceeded

Response (429):

```
{
  "error": "Too Many Requests",
  "details": "Rate limit exceeded. Retry after 42 seconds.",
  "retry_after_seconds": 42
}
```

Headers:

```
HTTP/1.1 429 Too Many Requests
Retry-After: 42
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 0
X-RateLimit-Reset: 1703721600
```

Best Practices

1. Transaction ID Uniqueness

Always use globally unique transaction IDs:

```python
import uuid

transaction_id = f"txn_{uuid.uuid4()}"
```

2. Error Handling

Implement retry logic with exponential backoff:

```python
import time
import requests

def call_api_with_retry(url, payload, max_retries=3):
    for attempt in range(max_retries):
        try:
            response = requests.post(url, json=payload, timeout=5)
            response.raise_for_status()
            return response.json()
        except requests.exceptions.RequestException as e:
            if attempt == max_retries - 1:
                raise
            time.sleep(2 ** attempt)  # Exponential backoff
```

3. Feedback Loop

Always submit feedback for closed transactions:

```
# 1. Get recommendations
recommendations = get_recommendations(transaction_data)

# 2. Route transaction to acquirer
acquirer = recommendations['recommendations'][0]['acquirer']
outcome = process_transaction(acquirer, transaction_data)

# 3. Submit feedback
submit_feedback({
    "transaction_id": transaction_data['transaction_id'],
    "acquirer_used": acquirer,
    "outcome": "approved" if outcome.success else "declined",
    "processing_time_ms": outcome.processing_time,
    "recommended_rank_used": 1
})
```

4. Timeouts

Set reasonable timeouts to avoid blocking:

```
response = requests.post(
    url,
    json=payload,
    timeout=5  # 5 seconds timeout
)
```

5. Monitoring

Track key metrics:

- Request latency (p50, p95, p99)
- Error rate by status code
- Recommendation accuracy (via feedback)
- Which rank is most commonly used

Versioning

API versioning is handled via URL path:

- **Current**: `/v1/recommendations`
- **Future**: `/v2/recommendations`

Breaking changes will be released as new versions. Non-breaking changes (new optional fields) may be added to existing versions.

Support

For API support:

- **Email**: yinon@sayfin.ai
- **Documentation**: https://docs.sayfin.ai
- **Status Page**: https://status.sayfin.ai

Last Updated: 2025-12-27 **API Version**: v1.0.0