



SECURE DIGITAL PAYMENTS

Partner API Documentation

Complete Integration Guide for Developers

API Version 1.0

Generated: February 16, 2026

Developer Support: developers@hoopaywallet.com

Website: <https://hoopaywallet.com>

Table of Contents

1. Introduction
2. Authentication & Security
3. Environments
4. Pay User API
5. Collect From User API
6. Wallet Verification API
7. Refund API
8. Fee API
9. Merchant Wallet APIs
10. Transaction History APIs
11. Webhooks
12. Error Codes
13. Best Practices

1. Introduction

Welcome to the HooPay Partner API documentation. This guide provides comprehensive information for integrating your platform with HooPay Wallet services.

What You Can Do

- **Pay User** – Partner pays money to user wallets (credit funds)
- **Collect From User** – Partner collects money from user wallets (debit funds)
- **Wallet Verification** – Verify wallet ownership before transactions
- **Refunds** – Process refunds for previous transactions
- **Fee Calculation** – Get real-time fee estimates
- **Merchant Wallet Monitoring** – Check balances, view ledgers, track activity
- **Transaction History** – Access unified transaction reports and analytics

Getting Started

Contact our partner team at

developers@hoopaywallet.com

to obtain your API credentials.

2. Authentication & Security

API Credentials

Each partner receives two credentials:

Credential	Description	Usage
API Key	Public identifier	Sent in <code>X-API-Key</code> header
API Secret	Private signing key	Used to generate HMAC signature (never sent)

Required Headers

Header	Description
<code>X-API-Key</code>	Your partner API key
<code>X-Signature</code>	HMAC-SHA256 signature of the request body
<code>X-Timestamp</code>	Unix timestamp (must be within 5 minutes)
<code>X-Idempotency-Key</code>	Unique request ID (for POST requests)
<code>Content-Type</code>	<code>application/json</code>
<code>Accept</code>	<code>application/json</code>

HMAC Signature Generation

Generate the signature by creating an HMAC-SHA256 hash of the JSON request body using your API Secret:

PHP EXAMPLE

```
$body = json_encode($requestData);
$signature = hash_hmac('sha256', $body, $apiSecret);
```

JAVASCRIPT EXAMPLE

```
const crypto = require('crypto');
const body = JSON.stringify(requestData);
const signature = crypto.createHmac('sha256', apiSecret)
  .update(body)
  .digest('hex');
```

PYTHON EXAMPLE

```
import hmac
import hashlib
import json

body = json.dumps(request_data, separators=(',', ':'))
signature = hmac.new(
    api_secret.encode(),
    body.encode(),
    hashlib.sha256
).hexdigest()
```

3. Environments

Environment	Base URL	Purpose
Sandbox	https://hoopaywallet.com/api/v1/partner	Testing & development (use sandbox API keys)
Production	https://hoopaywallet.com/api/v1/partner	Live transactions (use production API keys)

Sandbox Testing

Use sandbox credentials to test your integration without affecting real user wallets. Test wallet IDs: [123456](#) , [234567](#) , [345678](#)

4. Pay User API

Partner pays money to a HooPay user's wallet. Use this for payouts, rewards, refunds, and fund transfers from your platform to user wallets.

POST

/partner/pay-user

REQUEST PARAMETERS

Parameter	Type	Description
<code>user_wallet_id</code>	string	6-digit HooPay wallet ID
<code>amount</code>	string	Amount to pay user (as string, e.g., "100.00")
<code>currency</code>	string	Currency code (default: USD)
<code>reference_id</code>	string	Reference unique transaction reference
<code>description</code>	string	Transaction description
<code>metadata</code>	object	Custom key-value pairs

EXAMPLE REQUEST

```
POST /api/v1/partner/pay-user HTTP/1.1
Host: hoopaywallet.com
X-API-Key: hpk_prod_xxxxxxxxxxxxxx
X-Signature: a1b2c3d4e5f6...
Content-Type: application/json

{
  "reference_id": "PAY-2024-001234",
  "user_wallet_id": "123456",
  "amount": "100.00",
  "currency": "USD",
  "description": "Trading payout from FXPrimus"
}
```

SUCCESS RESPONSE (201 CREATED)

```
{
  "success": true,
  "data": {
    "deposit_id": "dep_abcl23xyz",
    "reference_id": "PAY-2024-001234",
    "status": "pending_settlement",
    "amount": "100.00",
    "fee_amount": "0.00",
    "net_amount": "100.00",
  }
}
```

```
        "currency": "USD",
        "user_wallet_id": "123456",
        "created_at": "2024-12-01T15:30:00Z"
    }
}
```

Deposit Statuses

Status	Description
<code>pending_settlement</code>	Deposit created, awaiting merchant wallet debit and user wallet credit
<code>pending_funding</code>	Merchant wallet has insufficient funds. Deposit queued and will auto-process when merchant tops up
<code>processing</code>	Settlement in progress
<code>completed</code>	Successfully credited to user wallet
<code>failed</code>	Settlement failed (see <code>failure_reason</code>)
<code>cancelled</code>	Deposit cancelled before settlement

Auto-Reconciliation

Deposits with status `pending_funding` are automatically processed when the merchant wallet is topped up. No manual intervention required.

5. Collect From User API

Partner collects money from a HooPay user's wallet. Requires user authorization via redirect flow. Use this when users need to fund their account on your platform.

⚠ Authorization Required

Collecting from users requires explicit user authorization. The user will be redirected to HooPay to confirm the transaction.

Step 1: Initiate Collection

POST

/partner/collect-from-user

REQUEST PARAMETERS

Parameter	Type	Description
<code>user_wallet_id</code>	string	6-digit HooPay wallet ID
<code>amount</code>	string	<code>Amount</code> to collect (as string, e.g., "500.00")
<code>reference_id</code>	string	Your unique transaction reference
<code>callback_url</code>	string	<code>Redirect</code> URL to redirect after authorization
<code>description</code>	string	Transaction description

SUCCESS RESPONSE

```
{  
  "success": true,  
  "data": {  
    "withdrawal_id": "wth_xyz789abc",  
    "reference_id": "COLLECT-2024-005678",  
    "status": "pending_authorization",  
    "amount": "500.00",  
    "authorization_url": "https://hoopaywallet.com/authorize/wth_xyz789abc",  
    "expires_at": "2024-12-01T16:00:00Z"  
  }  
}
```

Step 2: User Authorization

Redirect the user to the `authorization_url`. The user will:

1. Log in to HooPay (if not already logged in)
2. Review the collection details
3. Enter their PIN to confirm
4. Be redirected to your `callback_url`

Step 3: Handle Callback

Your callback URL will receive query parameters:

```
https://yourplatform.com/callback?  
withdrawal_id=wth_xyz789abc&  
status=completed&  
reference_id=COLLECT-2024-005678
```

Check Collection Status

GET

/partner/collect-from-user/{reference}

6. Wallet Verification API

Verify a wallet ID and retrieve masked user information before initiating transactions.

GET

/partner/wallets/{wallet_id}/verify

SUCCESS RESPONSE

```
{  
  "success": true,  
  "data": {  
    "wallet_id": "123456",  
    "verified": true,  
    "user": {  
      "name_masked": "J*** D**",  
      "email_masked": "j***@***.com",  
      "phone_masked": "+1***456",  
      "kyc_verified": true,  
      "account_status": "active"  
    },  
    "limits": {  
      "can_receive_deposits": true,  
      "can_withdraw": true,  
      "daily_limit_remaining": 9500.00,  
      "monthly_limit_remaining": 95000.00  
    }  
  }  
}
```

□ Best Practice

Always verify the wallet before paying a user to ensure the user exists and can receive funds.

7. Refund API

Reverse a previous deposit or withdrawal transaction.

POST

/partner/refunds

REQUEST PARAMETERS

Parameter	Type	Description
<code>type</code>	string	<code>pay-user</code> or <code>collect-from-user</code>
<code>original_transaction_id</code>	string	ID of original transaction
<code>amount</code>	number	Partial refund amount (full if omitted)
<code>reason</code>	string	Reason for refund

EXAMPLE REQUEST

```
{  
  "type": "pay-user",  
  "original_transaction_id": "pay_abc123xyz",  
  "reason": "Customer requested cancellation"  
}
```

8. Fee API

Calculate transaction fees before processing.

Get Fee Schedule

GET

/partner/fees

RESPONSE

```
{  
  "success": true,  
  "data": {  
    "deposit": {  
      "percentage": 0.00,  
      "min": null,  
      "max": null  
    },  
    "withdrawal": {  
      "percentage": 1.50,  
      "min": 0.50,  
      "max": 50.00  
    }  
  }  
}
```

Calculate Fee

POST

/partner/fees/calculate

REQUEST

```
{  
  "type": "withdrawal",  
  "amount": 500.00,  
  "currency": "USD"  
}
```

RESPONSE

```
{  
  "success": true,  
  "data": {  
    "amount": 500.00,  
    "fee_amount": 7.50,  
    "net_amount": 492.50,  
    "fee_breakdown": {  
      "percentage_rate": 1.50  
    }  
  }  
}
```

9. Merchant Wallet APIs

Monitor your settlement account balances, view transaction ledgers, and track merchant wallet activity.

□ Account Monitoring

Use these APIs to check your balance before initiating payouts, set up automated alerts for low balances, and access transaction ledgers for reconciliation.

9.1 List All Merchant Wallets

GET

/merchant-wallets

Retrieve all your merchant wallets across different currencies with current balances.

RESPONSE EXAMPLE

```
{  
  "success": true,  
  "data": {  
    "wallets": [  
      {  
        "id": 1,  
        "currency": "USD",  
        "balance": 15250.75,  
        "formatted_balance": "15,250.75 USD",  
        "status": "active",  
        "is_low_balance": false,  
        "low_balance_threshold": 1000.00  
      }  
    ],  
    "summary": {  
      "total_wallets": 2,  
      "active_wallets": 2,  
      "currencies": ["USD", "EUR"]  
    }  
  }  
}
```

9.2 Get Wallet Balance

GET

/merchant-wallets/{currency}/balance

Lightweight endpoint to quickly check your current balance. Perfect for balance checks before initiating payouts.

9.3 Get Wallet Details

GET

/merchant-wallets/{currency}

Get detailed information including 24-hour activity statistics (credits, debits, transaction counts).

9.4 Get Transaction Ledger

GET

/merchant-wallets/{currency}/ledger

Access your complete transaction ledger for reconciliation and audit purposes.

QUERY PARAMETERS

Parameter	Type	Description
type	string	Filter: credit, debit, all (default: all)
from_date	date	Start date (YYYY-MM-DD)
to_date	date	End date (YYYY-MM-DD)
per_page	integer	Results per page (1-100, default: 20)

Use Cases:

- **Pre-Payout Check:**

Verify sufficient balance before batch payouts

- **Daily Reconciliation:**

Download ledger entries and match with internal records

- **Low Balance Alerts:**

Monitor is_low_balance field for automated alerts

- **Monthly Reporting:**

Generate settlement reports with ledger data

10. Transaction History APIs

Access unified transaction history across all deposits and withdrawals. Perfect for reporting, analytics, and monitoring.

Unified View

These APIs provide a consolidated view of all your partner transactions, combining both deposits (Pay User) and withdrawals (Collect From User) into a single, sortable feed.

10.1 List Transactions

GET

/transactions

Retrieve a paginated list of all your transactions with powerful filtering options.

QUERY PARAMETERS

Parameter	Type	Description
<code>type</code>	string	Filter: deposit, withdrawal, all (default: all)
<code>status</code>	string	Filter by status (completed, pending, failed)
<code>user_wallet_id</code>	string	Filter by user wallet ID (6-digit)
<code>from_date</code>	date	Start date (YYYY-MM-DD)
<code>to_date</code>	date	End date (YYYY-MM-DD)
<code>per_page</code>	integer	Results per page (1-100, default: 20)

RESPONSE EXAMPLE

```
{  
  "success": true,  
  "data": [  
    {  
      "type": "deposit",  
      "id": "dep_abc123",  
      "reference_id": "ORDER-001",  
      "amount": 50.00,  
      "currency": "USD",  
      "status": "completed",  
      "user_wallet_id": "310146",  
      "created_at": "2025-01-28T10:30:00Z"  
    },  
    {  
      "type": "withdrawal",  
      "id": "pwd_xyz789",  
      "reference_id": "WITHDRAWAL-456",  
      "amount": 100.00,  
      "fee_amount": 2.50,  
      "net_amount": 97.50,  
      "currency": "USD",  
      "status": "completed",  
      "user_wallet_id": "123456",  
      "created_at": "2025-01-28T09:15:00Z"  
    }  
  ],  
  "meta": {  
    "current_page": 1,  
    "total": 95,  
    "per_page": 20  
  }  
}
```

10.2 Get Transaction Summary

GET

/transactions/summary

Get aggregated statistics about your transactions including counts, totals, and fees.

RESPONSE EXAMPLE

```
{  
  "success": true,  
  "data": {  
    "deposits": {  
      "count": 150,  
      "total_amount": 15000.00,  
      "pending_count": 5  
    },  
    "withdrawals": {  
      "count": 85,  
      "total_amount": 8500.00,  
      "total_fees": 212.50,  
      "pending_count": 2  
    },  
    "totals": {  
      "transaction_count": 235,  
      "volume": 23500.00,  
      "fees_collected": 212.50  
    }  
  }  
}
```

Common Use Cases:

- **Internal Dashboards:**

Build real-time transaction activity dashboards

- **Monthly Reports:**

Generate settlement reports for accounting

- **User Activity Tracking:**

Monitor transactions for specific users

- **Failed Transaction Monitoring:**

Identify and investigate failures

11. Webhooks

Receive real-time notifications for transaction status changes.

Webhook Events

Event	Description
deposit.completed	Pay User transaction successfully completed
deposit.pending_funding	Deposit queued due to insufficient merchant wallet balance. Will auto-process when merchant tops up.
deposit.completed_after_funding	Queued deposit completed after merchant wallet was topped up
deposit.failed	Pay User transaction failed
withdrawal.authorized	User authorized the collection
withdrawal.completed	Collect From User transaction successfully completed
withdrawal.failed	Collect From User transaction failed
withdrawal.cancelled	Collection cancelled by user
refund.completed	Refund successfully processed

Webhook Payload

```
POST /your-webhook-endpoint HTTP/1.1
X-Webhook-Signature: sha256=abc123...
Content-Type: application/json

{
  "event": "withdrawal.completed",
  "timestamp": "2024-12-01T15:30:00Z",
  "data": {
    "withdrawal_id": "wth_xyz789abc",
    "reference_id": "COLLECT-2024-005678",
    "status": "completed",
    "amount": "500.00",
    "fee_amount": "7.75",
    "net_amount": "492.25",
    "user_wallet_id": "123456"
  }
}
```

Verifying Webhook Signatures

```
// PHP
	payload = file_get_contents('php://input');
	$signature = $_SERVER['HTTP_X_WEBHOOK_SIGNATURE'];
	$expected = 'sha256=' . hash_hmac('sha256', $payload, $webhookSecret);

	if (!hash_equals($expected, $signature)) {
		http_response_code(401);
		exit('Invalid signature');
}
```

12. Error Codes

HTTP Status Codes

Code	Meaning
200	Success
201	Created
400	Bad Request - Invalid parameters
401	Unauthorized - Invalid credentials
403	Forbidden - Insufficient permissions
404	Not Found - Resource doesn't exist
409	Conflict - Duplicate request
422	Unprocessable - Validation failed
429	Too Many Requests - Rate limited
500	Server Error

Error Response Format

```
{  
  "success": false,  
  "error": {  
    "code": "INVALID_WALLET",  
    "message": "The specified wallet ID does not exist"  
  }  
}
```

Common Error Codes

Code	Description	Solution
UNAUTHORIZED	Invalid API key or signature	Check credentials and signature
INVALID_WALLET	Wallet not found	Verify wallet ID format (6 digits)
INSUFFICIENT_BALANCE	User lacks funds	Request smaller amount
LIMIT_EXCEEDED	Transaction limit reached	Check daily/monthly limits
DUPLICATE_REFERENCE	Reference ID already used	Use unique reference
RATE_LIMITED	Too many requests	Wait and retry with backoff

13. Best Practices

□ Security

- Store API credentials securely (use environment variables)
- Always use HTTPS in production
- Validate all webhook signatures
- Implement IP whitelisting if possible
- Rotate API keys periodically

□ Idempotency

- Always include `X-Idempotency-Key` for POST requests
- Use unique, deterministic keys (e.g., `order_123_deposit`)
- Safe to retry requests with the same idempotency key

⚡ Performance

- Implement exponential backoff for retries
- Cache fee schedules (refresh hourly)
- Use webhooks instead of polling for status updates
- Batch operations where possible

□ Testing

- Start with sandbox environment
- Test all error scenarios
- Verify webhook handling
- Test with various amounts and currencies

□ Monitoring

- Log all API requests and responses
- Monitor for failed transactions
- Set up alerts for high error rates
- Track webhook delivery success

HooPay Wallet - Partner API Documentation

Version 1.0 | February 16, 2026

□ Developer Support: **developers@hoopaywallet.com**

© 2026 HooPay Wallet. All rights reserved.