UnifiPay Direct API Reference — AI Integration Guide

How to Use This Document

- This document contains the complete specifications for the UnifiPay Direct API. The original source is `api-reference-direct/index.html` (an HTML page intended for human readers), and this document is a direct transcription of the API specifications contained therein.
- The AI coding assistant relies solely on this document when writing payment integration code. It does not make assumptions about content not included in this document, or fill in gaps by drawing from other UnifiPay documents (such as the Classic/PG·Merchant-oriented `api-reference/index.html`).
- Direct is the "payment link issuance" method. In this workflow, the merchant's server first issues a payment link and shares it with the customer, who then proceeds with the payment directly via that link; this differs from the API configuration used in Classic.
- In the text, terms such as `apiKey`, `apiSecret`, `X-API-Key`, signature, and `transactionId` are all example values provided for illustrative purposes. When actually integrating the system, these must be replaced with the actual values issued by UnifiPay.
- Sections not found on the original page (e.g., a payment refund API, automatic recurring payments, etc.) are also omitted from this document. Direct does not support refunds via API — they can only be processed through the console (see §2.5).

Table of Contents

- 3. Test Guide

Overview

This is an API reference for developers integrating the UnifiPay API. It provides a detailed explanation of how to integrate the entire process — from creating stablecoin payments to settlement — via the REST API. For information on the integration process and business details, refer to the Guides page.

Three Key Features (text from the original card):

Feature	Description
Quick Integration	Complete payment integration using only the REST API — no SDK required
HMAC Security	HMAC signature authentication applied to API requests
Webhook Support	Receive real-time payment status updates via asynchronous webhooks

---

1. API Integration Diagram

This diagram illustrates the data flow between the merchant's server, the customer, and Unifi Pay using the payment link issuance method.

Legend:

- **Solid arrow** = Synchronous call (Sync)
- **Gray dashed arrow (open arrowhead)** = Response
- **Orange dashed arrow** = Asynchronous Webhook (Async)

Actors: `Customer (Browser)`, `Your Server (Backend)`, `Unifi Pay (API Server)`

1.1 Payment Diagram

This shows the entire flow from issuing a payment link to receiving a webhook. (SVG sequence diagram paraphrased as a textual procedure.)

1. **[Synchronous]** Your Server → Unifi Pay: `POST /api/seller/v1/payment/link`
   Headers: `X-API-Key` · `X-Timestamp` · `X-Authorization-Hmac`
2. **[Response]** Unifi Pay → Your Server: `200 OK` — `linkUrl`, `linkId`
3. **[Synchronous]** Your Server → Customer: Share the payment link (`linkUrl`) with the customer

4. **[Synchronous]** Customer → Unifi Pay: Accesses the payment link
   (From this point onward, the process takes place on the "Unifi Pay Payment Link Page.")
5. **[Synchronous]** Customer → Unifi Pay: A payment request is generated when the customer clicks "Proceed to Payment"
   `transactionId (UUID)` · `orderId = LINK-{linkId}-{randomstring}`
6. **[Synchronous]** Customer → Unifi Pay: Connect wallet and approve payment (on-chain transfer)
7. **[Asynchronous Webhook]** Unifi Pay → Your Server: `POST {callbackUrl}`
   Sent only if `callbackUrl` is registered. Body: `status: CONFIRMED / FAILED / CANCELED`

Additional Explanation (Original Text):

- **Your Server**: Requests a payment link from Unifi Pay using the `apiKey` and `HMAC Signature`.
- **Payment Link**: Share the issued `linkUrl` with the customer. There are no restrictions on the sharing channel.
- **Customer**: When the customer accesses the link and proceeds with the payment, a payment request is generated, and the payment is completed via wallet connection and signature.
- **Webhook**: Sent only if a `callbackUrl` was registered when the payment link was issued. If not registered, you must check the payment status via the Payment Status Check API.

---

2. API Detail

Base URL

Environment	Base URL
Preview	`https://app-api-pay.unifi.me`
Production	`https://api-pay.unifi.me`

2.1 Authentication

HMAC authentication is required by default for API requests between servers. The signature value must be included in the `X-Authorization-Hmac` header.

> **Default Authentication Rule**: Unless otherwise specified, HMAC authentication is the default for all server-to-server APIs. APIs that use a different authentication method (e.g., unauthenticated public APIs) are noted separately for each API.

HMAC Generation Formula

```
BASE64(HMACSHA256(apiSecret, {HTTP_METHOD}{URI}{X-API-Key}{X-Timestamp}
{REQUEST_BODY}))
```

Example 1 — With a REQUEST_BODY

Endpoint: `POST /api/seller/v1/payment/link`

Request Example:

```
request header:
  X-API-Key:            "b6ef2f7c-3e74-438c-b456-689e821992be"
  X-Timestamp:          1773017787000
  X-Authorization-Hmac: "some-hmac-signature"
request body:
{
  "requestId": "REQ-20260304-0001",
  "storeId": "123",
  "serviceName": "MyShop",
  "itemName": "ITEM-GAME-001",
  "itemPrice": 12.34,
  "orderCurrencyCode": "USD",
  "returnUrl": "https://myshop.com/payment/returnUrl",
  "callbackUrl": "https://myshop.com/payment/callbackUrl",
  "expiresAt": "2026-08-31T00:00:00Z"
}
```

Generate signature:

```
X-Authorization-Hmac = Base64(
  HmacSHA256(
    apiSecret,
    "POST"                                    ← HTTP_METHOD
    + "/api/seller/v1/payment/link"           ← URI
    + "b6ef2f7c-3e74-438c-b456-689e821992be" ← X-API-Key
    + "1773017787000"                         ← X-Timestamp
    +
```

```
"{"requestId":"REQ-20260304-0001","storeId":"123","serviceName":"MyShop","itemName":"
ITEM-GAME-001","itemPrice":12.34,"orderCurrencyCode":"USD","returnUrl":"https://
myshop.com/payment/returnUrl","callbackUrl":"https://myshop.com/payment/
callbackUrl","expiresAt":"2026-08-31T00:00:00Z"}"       ← REQUEST_BODY (공백 없이)
  )
)
```

Example 2 — Without a REQUEST_BODY

Endpoint: `GET /api/seller/v1/payment/123`

Request Example:

```
request header:
  X-API-Key:            "b6ef2f7c-3e74-438c-b456-689e821992be"
  X-Timestamp:          1773017787000
  X-Authorization-Hmac: "some-hmac-signature"
```

Signature Generation:

```
X-Authorization-Hmac = Base64(
  HmacSHA256(
    apiSecret,
    "GET"                                       ← HTTP_METHOD
    + "/api/seller/v1/payment/123"              ← URI
    + "b6ef2f7c-3e74-438c-b456-689e821992be" ← X-API-Key
    + "1773017787000"                           ← X-Timestamp
                                                ← REQUEST_BODY 없음 (생략)
  )
)
```

Example 3 — With Query Parameters

Endpoint: `GET /api/seller/v1/payment/settlement/transaction`

Request Example:

```
request url:
  GET /api/seller/v1/payment/settlement/transaction?
from=2026-03-01T00:00:00Z&to=2026-03-31T23:59:59Z&paymentType=PURCHASE&page=0&size=20

request header:
  X-API-Key:            "b6ef2f7c-3e74-438c-b456-689e821992be"
```

```
  X-Timestamp:          1773017787000
  X-Authorization-Hmac: "some-hmac-signature"
```

Signature Generation:

```
X-Authorization-Hmac = Base64(
  HmacSHA256(
    apiSecret,
    "GET"                                                  ← HTTP_METHOD
    + "/api/seller/v1/payment/settlement/transaction"  ← URI (쿼리스트링 제외)
    + "b6ef2f7c-3e74-438c-b456-689e821992be"         ← X-API-Key
    + "1773017787000"                                    ← X-Timestamp
                                                           ← REQUEST_BODY 없음 (생략)
  )
)
```

> ⚠️ **Handling Query Parameters**: The `{URI}` used for the signature includes only the path; the query string (everything after `?`) is not included. Signing with the query string included will cause authentication to fail.

Request Header Parameters

Parameter	Type	Required	Description
`X-Authorization-Hmac`	STRING	Required	The generated HMAC signature value. Required for all server-to-server API requests.
`X-API-Key`	STRING	Required	The apiKey issued by Unifi Pay.
`X-Timestamp`	STRING	Required	Unix timestamp in milliseconds. Requests will be rejected if the timestamp differs from the current time by more than 5 minutes.

> ⚠️ **Timestamp Validity Range**: The `X-Timestamp` must be within 5 minutes of the current server time. If it falls outside this range, the request will be automatically rejected to prevent replay attacks.

Server Timestamp Lookup

Retrieve the timestamp from the Unifi Pay server to verify the server's status, and use the timestamp value from the response in the HMAC signature.

```
GET /api/v1/server/time
```

> **Authentication Method**: This API is an unauthenticated public API. Call it without the `X-API-Key`, `X-Timestamp`, and `X-Authorization-Hmac` headers. Since it is used to obtain the timestamp for the HMAC signature, it is excluded from the default authentication rule (HMAC required) described above.

Response Fields:

Field	Type	Description
`timestamp`	INTEGER(INT64)	The server's timestamp in milliseconds

Example of a successful response (200 OK):

```
{
  "timestamp": 1773017787000
}
```

Example of an error response (500 Internal Server Error):

```
{
  "code": "INTERNAL_SERVER_ERROR",
  "message": ""
}
```

2.2 Payment Link Issuance API

Issues a payment link for a fixed amount and a product. Sharing the link received in the response with the customer directs them to the payment page without requiring any additional integration.

```
POST /api/seller/v1/payment/link
```

Request Body Parameters

Parameter	Type	Length	Required	Description
`requestId`	STRING	64	Required	A unique identifier (id) for the issuance request. Use only letters, numbers, hyphens, and underscores. Resubmitting

Parameter	Type	Length	Required	Description
				the request with the same `requestId` will be rejected as a duplicate issuance.
`storeId`	STRING	128	Required	Merchant ID information provided by the requester.
`serviceName`	STRING	40	Required	Service or merchant information to be displayed on the link screen and payment window.
`itemName`	STRING	100	Required	Name of the product being sold (product name).
`itemPrice`	DOUBLE	—	Required	Price of the product being sold. Expressed in the order currency and must be greater than 0.
`orderCurrencyCode`	STRING	8	Required	Order currency code following ISO 4217. Supports `USD` or `JPY`.
`returnUrl`	STRING	2048	Optional	The URL to which the user will be redirected after payment is complete. If not specified, the payment will proceed without a return button.
`callbackUrl`	STRING	2048	Optional	The Webhook URL to receive the payment result. If not specified, no Webhook will be sent, so the result must be retrieved using the Payment Status Check API.
`expiresAt`	STRING	—	Required	The expiration time of the payment link. Must be in ISO-8601 format and represent a future date and time. If the link is accessed after expiration, an expiration error will be returned.

Response Fields

Field	Type	Description
`linkUrl`	STRING	The URL of the generated payment link. Share this with the customer.
`linkId`	STRING	Link identifier. The merchant should retain this value; to expire the link, use the "Force Expiration" button in the payment link list in the console.

Example Request (JSON)

```
{
  "requestId": "REQ-20260304-0001",
  "storeId": "123",
  "serviceName": "MyShop",
```

```
  "itemName": "ITEM-GAME-001",
  "itemPrice": 12.34,
  "orderCurrencyCode": "USD",
  "returnUrl": "https://myshop.com/payment/returnUrl",
  "callbackUrl": "https://myshop.com/payment/callbackUrl",
  "expiresAt": "2026-08-31T00:00:00Z"
}
```

Example Response (JSON)

```
{
  "linkUrl": "https://unifipay.example.com/pay/links/3f1d0a549c1e4f6a8a5b2f",
  "linkId": "3f1d0a549c1e4f6a8a5b2f"
}
```

> 💡 **Note**: If you share the `linkUrl` from the response with the customer, the payment will proceed without any additional integration. Since `requestId` is an idempotent key, resubmitting a request with the same value will result in the duplicate issuance being rejected.

Error Responses

HTTP Status	Error Code	Description
400	`BAD_REQUEST`	Invalid input values (format error, missing required values, `expiresAt` set to a past date, etc.)
401	`UNAUTHORIZED`	Invalid HMAC
403	`FORBIDDEN`	Unauthorized IP or Path
503	`MAINTENANCE`	Under Maintenance
406	`PAYMENT_INVALID_REQUEST`	If the request is resubmitted with the same `requestId`, or if the `callbackUrl` is not on the allowlist
406	`PAYMENT_UNSUPPORTED_POLICY`	Unsupported order currency or amount policy (orderCurrencyCode, itemPrice)
406	`PAYMENT_STORE_INACTIVE`	When the requested `storeId` is not in the `ACTIVE` state
406	`PAYMENT_WALLET_NOT_REGISTERED`	If the settlement wallet address is not registered
500	`INTERNAL_SERVER_ERROR`	Internal server error

2.3 Payment Status Check API

Checks the current status of a created payment transaction.

```
GET /api/seller/v1/payment/{transactionId}
```

Response Fields

Field	Type	Description
`transactionId`	STRING	Transaction number generated by Unifi Pay.
`orderId`	STRING	ID information for the merchant transaction.
`storeId`	STRING	Merchant ID information provided by the requester.
`serviceName`	STRING	Service or merchant information to be displayed on the payment screen.
`status`	STRING	See the enum below
`failType`	STRING	Set when `status` is `FAILED`. See the enum below.
`items`	Array	List of products to be billed. Maximum of 1 entry
`items[].name`	STRING	Identification information for the product being sold.
`items[].price`	DOUBLE	Price of the product being sold.
`orderCurrencyCode`	STRING	Order currency code following ISO 4217. Currently fixed as `"USD"`.
`orderAmount`	DOUBLE	The amount of the order.
`payCurrencyCode`	STRING	Type of stablecoin used for payment.
`payAmount`	DOUBLE	Amount paid.
`blockchainTxId`	STRING	Blockchain transaction hash. Returned when the payment is finally successful (`CONFIRMED`) or fails (`FAILED`) after being propagated to the blockchain network.
`blockchainNetworkFee`	DOUBLE	Blockchain network fee. Returns the actual fee amount consumed at the time the payment was successful (`CONFIRMED`) or failed (`FAILED`) after propagation across the network. (Unifi covers the full amount; there is no cost to the user.)

`status` enum:

Value	Description
`CREATED`	Payment creation complete
`PENDING`	Payment in progress

Value	Description
`CONFIRMED`	Payment successful (closed)
`FAILED`	Payment failed (closed)
`CANCELED`	Payment canceled (closed)

`failType` enum (set when status is FAILED):

Value	Description
`WALLET_PURCHASE_BLOCKCHAIN_FAILED`	Failure during the blockchain withdrawal process in the unifi wallet
`WALLET_PURCHASE_INSUFFICIENT_BALANCE`	Insufficient balance in the user's wallet
`PAYMENT_PURCHASE_BLACKLISTED`	Failure because the user's wallet is on the payment blacklist
`UNKNOWN`	Unknown error during the payment process

Example Response


```
{
  "transactionId": "<uuid>",
  "orderId": "<orderId>",
  "storeId": "<storeId>",
  "serviceName": "<serviceName>",
  "items": [
    {
      "name": "ITEM-GAME-001",
      "price": 50000
    }
  ],
  "status": "CONFIRMED",
  "failType": null,
  "countryCode": "KOR",
  "payAmount": 49000,
  "payCurrencyCode": "USDT",
  "orderAmount": 50000,
  "orderCurrencyCode": "USD",
  "blockchainTxId":
"0x4fce0d72ff7f4b9f4ba0cf58d30119924bea3811d85a830c762e1d87b3e1ee17",
  "blockchainNetworkFee": 0.00253778
}
```

(Note: The response example actually includes a `countryCode` field that is not listed in the "Response Fields" table above. Translated verbatim from the original text.)

Status Information:

- **CONFIRMED**: The payment has been successfully completed and confirmed.
- **FAILED**: The payment has failed.

Error Responses

HTTP Status	Error Code	Description
400	`BAD_REQUEST`	Invalid input value
401	`UNAUTHORIZED`	If the HMAC is invalid
403	`FORBIDDEN`	If the IP address or path is not allowed
503	`MAINTENANCE`	During maintenance
406	`PAYMENT_TRANSACTION_NOT_FOUND`	Invalid transactionId
500	`INTERNAL_SERVER_ERROR`	Internal server error

2.4 Payment Webhook

Asynchronously receives confirmation of whether a created payment has been completed.

```
POST <가맹점 callbackUrl>
```

Request Body Fields

Field	Type	Required	Description
`transactionId`	STRING	Required	Transaction number generated by Unifi Pay.
`orderId`	STRING	Required	The order ID generated by Unifi Pay. For transactions made via a payment link, this is issued in the format `LINK-{linkId}-{random string}` and is determined when the customer completes the payment via the link.
`status`	STRING	Required	Payment processing result. `CONFIRMED` / `FAILED` / `CANCELED`.
`type`	STRING	Required	Transaction type. Since refunds are not supported via the API, this is always `PURCHASE`.

Example Request (JSON)

```
{
  "transactionId": "<uuid>",
  "orderId": "LINK-3f1d0a549c1e4f6a8a5b2f-a1b2c3",
  "status": "CONFIRMED",
  "type": "PURCHASE"
}
```

> 📡 **Important Response Note**: The merchant server must return an `HTTP 200 OK` response code after receiving the webhook.
>
> 💡 **Detailed Information**: The webhook transmits only the minimum required information. To retrieve payment details, call the 2.3 Payment Status Check API after receiving the webhook.

Retry Policy

If the merchant server returns a `5xx` error or does not respond, the Webhook is resent according to the policy below.

Order	Retry timing
1st retry	After 3 seconds
2nd retry	After 30 seconds
3rd retry	After 10 minutes
4th retry	After 1 hour
5th retry	After 3 hours

2.5 Refunds (Console Processing)

Refunds are not supported via the API. Since this process requires a wallet signature, it can only be executed through the console.

> **Procedure**: The merchant executes the refund by signing directly from their own Unifi Wallet via the **Refund Management** menu in the Unifi Pay Console.

Integration Notes

Item	Description
Refund Webhook	Not provided. Since refund processing results are not sent via Webhook, they must be retrieved using the Payment History API.
Refund history lookup	Use the 2.6 Payment History API described below, with `paymentType=REFUND`.
Link to original payment	The `originTransactionId` of the refund transaction points to the `transactionId` of the original payment.

> 💡 **Note**: For refund policies — including the refund wallet, refund amount criteria, address, and network conditions — refer to the User Guide "5. Refunds (Transfer Support Feature)" (`../guides/index.html#refund`).

2.6 Payment History API

Check payment and refund history.

```
GET /api/seller/v1/payment/settlement/transaction
```

Request Query Parameters

Parameter	Type	Required	Description
`from`	STRING	Conditional	Query start date (based on `capturedAt`, ISO 8601 UTC; e.g., `2026-03-01T00:00:00Z`). Required if `transactionId` or `settlementId` is not specified, and must be set to a date earlier than the `to` date.
`to`	STRING	Conditional	Query end date (based on `capturedAt`, ISO 8601 UTC; e.g., `2026-03-31T23:59:59Z`). Required if `transactionId` / `settlementId` is not specified; the query period is limited to a maximum of 90 days.
`paymentType`	STRING	Optional	Payment type filter (`PURCHASE` / `REFUND`). If not specified, all types are included.
`transactionId`	STRING	Optional	The transaction ID generated by Unifi Pay. Required if `from`, `to`, or `settlementId` are not specified.
`settlementId`	STRING	Optional	Unifi Pay payment statement ID information. Required if `from`, `to`, or `transactionId` are not specified.
`page`	INTEGER	Optional	Page number (starting from 0, default: 0)

Parameter	Type	Required	Description
`size`	INTEGER	Optional	Page size (default: 20)

Response Fields (200 OK)

Field Name	Type	Required	Description
`content`	List	Required	Payment data list
`content[].partnerCorpName`	STRING	Required	Corporate name (EN)
`content[].storeId`	STRING	Optional	Merchant ID information
`content[].orderId`	STRING	Required	ID information for the transaction
`content[].itemName`	STRING	Required	Product identification information
`content[].serviceName`	STRING	Optional	Service or merchant name (EN)
`content[].orderCountryCode`	STRING	Required	Buyer's country code
`content[].orderCurrencyCode`	STRING	Required	Order currency code in accordance with ISO 4217
`content[].orderAmount`	DOUBLE	Required	Order amount
`content[].createdAt`	INSTANT	Required	Order payment request date and time
`content[].transactionId`	STRING	Required	Transaction ID generated by Unifi Pay.
`content[].blockchainTxId`	STRING	Required	KaiaScan TxID
`content[].paymentType`	STRING	Required	Payment: `PURCHASE` / Refund: `REFUND`
`content[].originTransactionId`	STRING	Optional	Original transaction ID for refunds (required for `REFUND`; Null for `PURCHASE`)
`content[].status`	STRING	Required	Fixed to `CONFIRMED` (only `CONFIRMED` transactions are returned)
`content[].finalizedAt`	INSTANT	Required	Date and time of the order's final payment status
`content[].capturedAt`	INSTANT	Required	Order payment completion date and time
`content[].failType`	STRING	Optional	Reason for order payment failure
`content[].payCurrencyCode`	STRING	Required	Actual payment currency, USDT
`content[].payAmount`	DOUBLE	Required	Actual payment amount

Field Name	Type	Required	Description
`content[].paymentExchangeRate`	DOUBLE	Required	Exchange rate at the time of payment. Applies when the payment currency differs from the order currency; if they are the same, the value is 1. (Base currency: `payCurrencyCode`)
`content[].buyerWalletAddress`	STRING	Required	Buyer's payment wallet address
`content[].variableFeeRate`	DOUBLE	Optional	Settlement fee rate (%). Null if the Unifi Pay payment statement number has not yet been generated
`content[].settlementId`	STRING	Optional	Unifi Pay payment statement ID information. Null if not yet generated
`content[].settlementCurrencyCode`	STRING	Optional	Settlement currency code (USDT, JPYC). Null if settlement has not yet occurred
`content[].settlementExchangeRate`	DOUBLE	Optional	Exchange rate at the time of settlement. Null if settlement has not yet occurred
`totalElements`	LONG	Required	Total number of records
`totalPages`	INTEGER	Required	Total number of pages
`pageNumber`	INTEGER	Required	Current page number
`first`	BOOLEAN	Required	Whether it is the first page
`last`	BOOLEAN	Required	Whether it is the last page

Example Response — Success


```
{
  "content": [
    {
      "partnerCorpName": "ABC Corporation",
      "storeId": "store_001",
      "orderId": "ORD-20260301-0001",
      "itemName": "Premium Subscription",
      "serviceName": "ABC Service",
      "orderCountryCode": "KR",
      "orderCurrencyCode": "USD",
      "orderAmount": 50000,
      "createdAt": "2026-03-01T10:00:00Z",
```

```
      "transactionId": "txn_abc123def456",
      "blockchainTxId": "0xabcdef1234567890abcdef1234567890abcdef12",
      "paymentType": "PURCHASE",
      "originTransactionId": null,
      "status": "CONFIRMED",
      "finalizedAt": "2026-03-01T10:01:30Z",
      "capturedAt": "2026-03-01T10:01:25Z",
      "failType": null,
      "payCurrencyCode": "USDT",
      "payAmount": 10.87,
      "paymentExchangeRate": 1.0,
      "buyerWalletAddress": "0x1234567890abcdef1234567890abcdef12345678",
      "variableFeeRate": 1.5,
      "settlementId": "stl_xyz789",
      "settlementCurrencyCode": "USDT",
      "settlementExchangeRate": 1.0
    }
  ],
  "totalElements": 1,
  "totalPages": 1,
  "pageNumber": 0,
  "first": true,
  "last": true
}
```


⚠️ **Note on Discrepancy in Original Source**: The original HTML page contains two sets of JSON with different values: the JSON displayed on the screen above (as shown in the example, `orderCurrencyCode: "USD"`, `orderAmount: 50000`) and the JSON embedded in the copy button (`orderCurrencyCode: "KRW"`, `orderAmount: 15000.0`). This discrepancy originates from the source page itself, and it is not possible to determine which example is accurate based on this document alone. When actually integrating the service, we recommend verifying the exact behavior of the currency and amount fields with UnifiPay Technical Support (`pay-support@unifi.me`) before proceeding.

Example Response — Failure


```
{
  "code": "PAYMENT_INVALID_REQUEST",
  "message": "Date range must not exceed 90 days"
}
```

Error Type Codes

HTTP Status	Error Code	Description
400	`BAD_REQUEST`	Required parameter missing or type mismatch
404	`NOT_FOUND`	Partner information not found
406	`PAYMENT_INVALID_REQUEST`	Invalid request (date format error, required conditions not met, exceeded 90-day range, etc.)
500	`INTERNAL_SERVER_ERROR`	Internal server error

---

3. Test Guide

This guide explains how to test Unifi Pay payment integration in the Preview environment.

1. Unifi Pay testing can be conducted in the Preview environment.
2. Unifi Wallets are created and linked in the Preview environment using a Google account that meets Production standards. (Unifi Wallets created in the Preview environment are managed separately from those in the Production environment.) Preview Unifi Wallet: `https://app.unifi.me/`
3. Payments in the Preview environment are processed based on Kairos, the testnet environment of the Kaia Blockchain.
4. Follow the guide below to obtain test USDT on the Kairos testnet and then proceed with payment testing.

Test Environment Setup Procedure (rewritten text from the original SVG diagram, 3 steps within the "PREVIEW (Kairos Testnet)" section):

1. **STEP 1 — Create a Unifi Wallet**: Create and link a Unifi Wallet in the Preview environment
2. **STEP 2 — Obtain Test USDT**: Receive Kairos USDT from the Kaia Faucet
3. **STEP 3 — Conduct Payment Testing**: Test API integration in the Preview environment

Preview Environment Information:

Item	Value
Environment	Preview Environment / Kairos Testnet
Description	In the Preview environment, the Kairos Chain — the testnet for the Kaia blockchain — is integrated, and test funds can be obtained through the Kaia Faucet service.
Kaia Faucet	`https://www.kaia.io/ko/faucet`
Base URL	

Item	Value
	`https://app-api-pay.unifi.me`